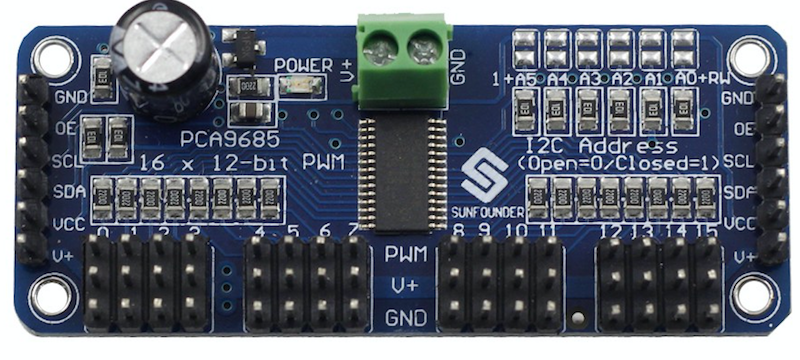
**PCA9685—i2c转pwm**



**一、引脚**

VCC：数字电源，3.3V或5V

V+：驱动电源，不高于10V

OE：芯片的使能引脚，低电平时使能芯片。模块中已经下拉保持低电平

SDA：I2C数据线

SCL：I2C时钟线

PWM：16路PWM输出端

GND：地

**二、I2C地址**

右上角有6个地址控制脚A5~A0，通过这些引脚可以控制设备的i2c地址。  
7位的I2C地址为：0x40 + A5:A0，A5到A0如果不做任何处理的话是0，此时地址为0x40。想要改变地址可以把对应位置引脚焊到一起，则为1。  
另外用i2cdetect检测出还有一个**0x70**地址一直存在，这是一个通用地址，可以给所有从机下达指令。

**三、寄存器**



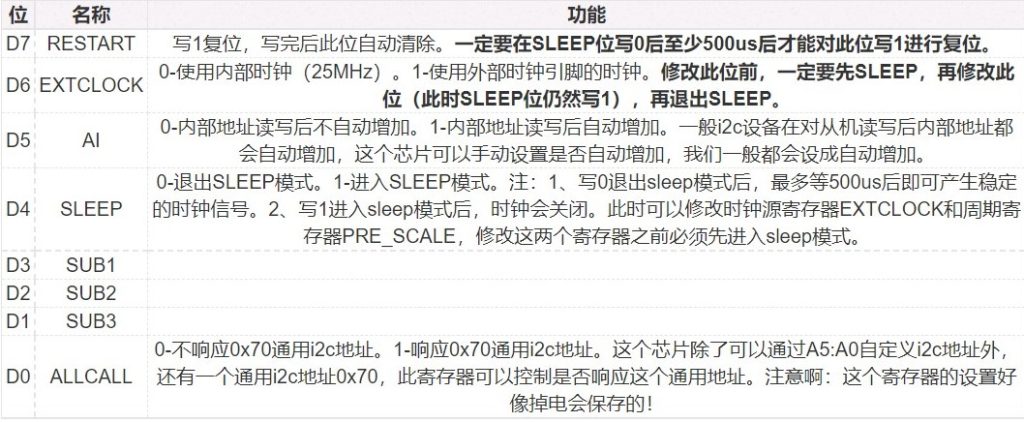
MODE1、MODE2是设置寄存器，用于初始化工作状态，一般只使用MODE1。

LED0\_ON\_L、LED0\_ON\_H、LED0\_OFF\_L、LED0\_OFF\_H是控制通道零的寄存器。一共有16个通道的PWM，每个通道都由四个寄存器控制，从地址0x06，按顺序往下排。

ALL\_LED\_ON\_L、ALL\_LED\_ON\_H、ALL\_LED\_OFF\_L、ALL\_LED\_OFF\_H这四个寄存器用来同时控制所有的通道。

PRE\_SCALE寄存器用于设置PWM的周期。

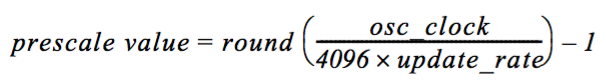
**3.1 MODE1寄存器**



重启位（RESTART）：  
如果PCA9685正在工作，用户在没有关闭PWM通道的情况下，决定让芯片进入SLEEP状态，那么在最后一个PWM周期结束后，RESTART位才会被置1，时钟停止后，各PWM寄存器(就是通道控制寄存器，每路4个）的内容会被保留。  
再切换到正常模式时，为了重启之前保留的PWM设置，可以按下面几步进行：  
1.读取MODE1寄存器内容。  
2.检查第7位（RESTART）是否是1.如果是，清空第4位（SLEEP）（写0），等待500us使时钟稳定。  
3.写MODE1的第7位写1。所有的PWM通道将会重新工作，且RESTART位会清空。  
备注：在向RESTART位写1之前，SLEEP位一定要至少持续500us为0。  
其他可以清空RESTART位的情况：  
1.上电。  
2.I2C软件重启命令。  
3.如果MODE2 OCH位为0，向任何PWM寄存器写数据后，I2C总线产生STOP。  
4.如果MODE2 OCH位为1，向任何通道的所有4个PWM寄存器都写一次数据。  
同时，如果用户在设置SLEEP位之前，人为的关闭所有PWM通道（有两个方法，最快的方法是向ALL\_LED\_OFF\_H寄存器的4位写1，另一个是向所有通道的LEDn\_OFF\_H寄存器的4位写1），RESTART位也会被清空。如果这样做，所有的PWM寄存器的内容都会作废，在重新启用之前需要设置。

**3.2PRE\_SCALE寄存器**

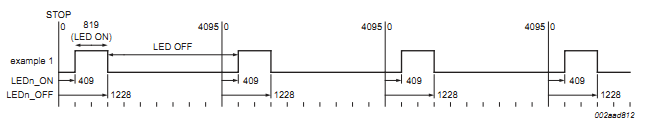
用于设置PWM的周期，计算公式为：



**osc\_clock**：选用的时钟频率，如使用内部25MHz时钟，即为25 000 000。  
**update\_rate**：PWM的频率，如设置PWM周期为20ms，则update\_rate=50Hz。  
**4096**：计数器是12位，故为4096。  
**round**：四舍五入  
**prescale value**：寄存器要填入的值

**3.3各个通道的ON和OFF寄存器**

LEDn\_ON和LEDn\_OFF寄存器控制PWM占空比。简单来说，在一个PWM周期内，LEDn\_ON控制的是什么时候开，LEDn\_OFF控制的是什么时候关。两者的取值范围都是从0到4095，这两个寄存器在工作时会和一个持续计数（0到4095）的计数器进行比较。当计数器的值等于LEDn\_ON的值时，开始输出低电平，当计数器的值等于LEDn\_OFF时开始输出低电平。



**四、示例**

以Jetson nano单片机，python语言为例：

class PCA9685:

def \_\_init\_\_(self, pca\_i2c):

self.i2c = pca\_i2c

self.address = 0x40 #设备地址

self.MODE1 = 0x00 #设置寄存器

self.LED0\_ON\_L = 0x06

self.LED0\_ON\_H = 0x07

self.LED0\_OFF\_L = 0x08

self.LED0\_OFF\_H = 0x09

self.PRE\_SCALE = 0xFE #控制周期寄存器

self.pca\_init() #初始化

#初始化

def pca\_init(self):

mode = self.i2c.read\_byte\_data(self.address, self.MODE1) #获取设置寄存器的值

newmode = (mode&0x70)|0x10 #休眠

self.i2c.write\_byte\_data(self.address, self.MODE1, newmode)

self.i2c.write\_byte\_data(self.address, self.PRE\_SCALE, 122) #设置频率为50HZ

mode &= 0xef #退出休眠

self.i2c.write\_byte\_data(self.address, self.MODE1, mode)

time.sleep(0.2)

self.i2c.write\_byte\_data(self.address, self.MODE1, mode | 0x80) #复位

#设置占空比

def setPWM(self, num, on, off): #num:要控制的通道

self.i2c.write\_byte\_data(self.address, self.LED0\_ON\_L+4\*num, on)

self.i2c.write\_byte\_data(self.address, self.LED0\_ON\_H+4\*num, on>>8)

self.i2c.write\_byte\_data(self.address, self.LED0\_OFF\_L+4\*num, off)

self.i2c.write\_byte\_data(self.address, self.LED0\_OFF\_H+4\*num, off>>8)

if \_\_name\_\_ =='\_\_main\_\_':

import smbus

import time

b = smbus.SMBus(1) #设定要用的总线

pca9685 = PCA9685(b)

pca9685.setPWM(0, 0, 290)

time.sleep(1)

pca9685.setPWM(0, 0, 304)