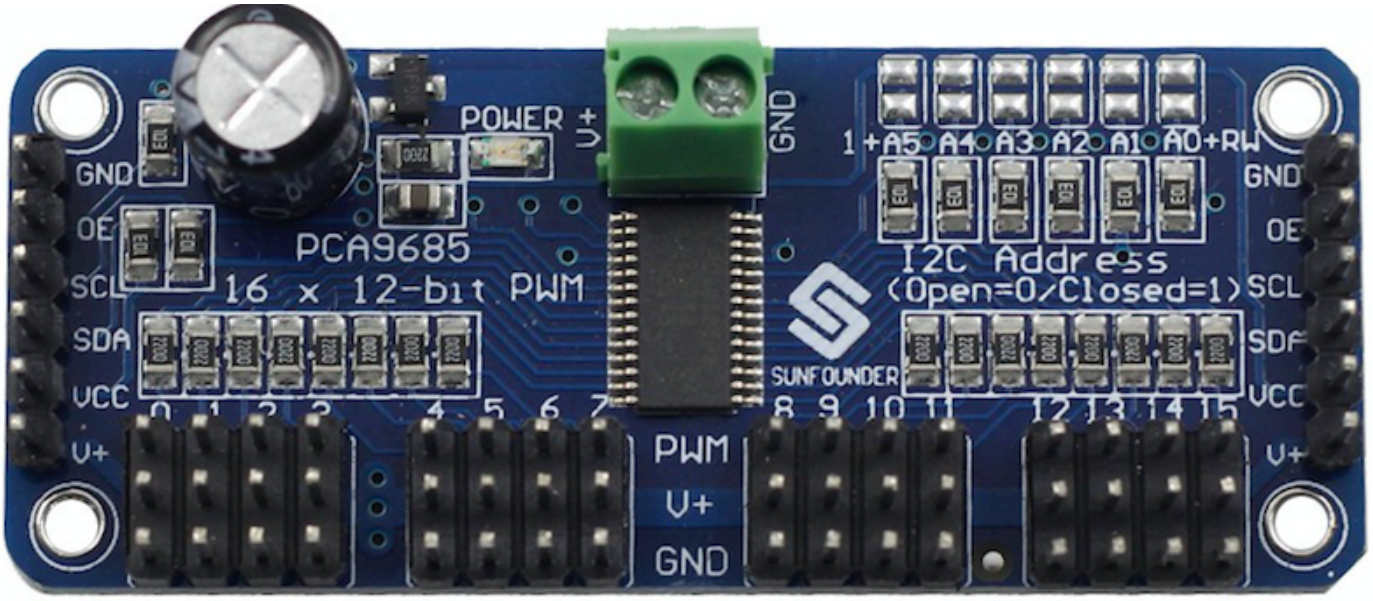
PCA9685 16路12位pwm信号发生器

目录(?) [+]

原文地址：http://nicekwell.net/blog/20161213/pca9685-16lu-12wei-pwmxin-hao-fa-sheng-qi.html

16路12位PWM信号发生器，可用于控制舵机、led、电机等设备，i2c通信，节省主机资源。



一、概述和硬件

1、概述

很常见的模块板子是这个样子，这个板子也比较便宜，十几块钱一个。
i2c通信，只需要几根i2c线就可以控制16路pwm，周期和占空比都可控。
可以多个模块级联。
可控制16路通道的四种工作模式：关、开、pwm、可变pwm。
精度是12位：

工作频率	时间分辨率	通常舵机500~2500us可分成份数	通常舵机500~2500us，旋转角180°的角度分辨率
50Hz	4.88us	410份	0.439°
60Hz	4us	492份	0.366°

驱动方式可以选择开漏输出或推挽输出。

2、硬件

1、电压

数字电路电压范围可接受3.3和5v电平。
此外还有一个v+引脚，这个引脚是给舵机供电用的，可以接稍微高一点的电压。

2、i2c地址

有6个地址控制脚，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，想要把哪一位置1就把那个引脚焊到一起。
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。

3、使能脚

模块有一个OE反使能脚，这个引脚低电平使能，不接的话模块内部默认已经接地使能了，所以正常使用可以不接。

二、寄存器功能

:-:|:-:|:-:

内部地址(hex)	名称	功能
00	MODE1	设置寄存器1
01	MODE2	设置寄存器2
02	SUBADR1	i2c-bus subaddress1
03	SUBADR2	i2c-bus subaddress2
04	SUBADR3	i2c-bus subaddress3
05	ALLCALLADR	
06	LED0_ON_L	
07	LED0_ON_H	
08	LED0_OFF_L	
09	LED0_OFF_H	
…	…	…
0x06 + 4*X	LEDX_ON_L	
0x06 + 4*X + 1	LEDX_ON_H	
0x06 + 4*X + 2	LEDX_OFF_L	
0x06 + 4*X + 3	LEDX_OFF_H	
…	…	… 上面共16路通道
FA	ALL_LED_ON_L	
FB	ALL_LED_ON_H	
FC	ALL_LED_OFF_L	
FD	ALL_LED_OFF_H	
FE	PRE_SCALE	控制周期的寄存器
FF	TestMode	

MODE1寄存器

位	名称	功能
D7	RESTART	写1复位，写完后此位自动清除。**一定要在SLEEP位写0后至少500us后才能对此位写1进行复位。**
D6	EXTCLOCK	0-使用内部时钟（25MHz）。1-使用外部时钟引脚的时钟。**修改此位前，一定要先SLEEP，再修改此位（此时SLEEP位仍然写1），再退出SLEEP。**
D5	AI	0-内部地址读写后不自动增加。1-内部地址读写后自动增加。一般i2c设备在对从机读写后内部地址都会自动增加，这个芯片可以手动设置是否自动增加，我们一般都会设成自动增加。
D4	SLEEP	0-退出SLEEP模式。1-进入SLEEP模式。注：1、写0退出sleep模式后，最多等500us后即可产生稳定的时钟信号。2、写1进入sleep模式后，时钟会关闭。此时可以修改时钟源寄存器EXTCLOCK和周期寄存器PRE_SCALE，修改这两个寄存器之前必须先进入sleep模式。
D3	SUB1	
D2	SUB2	
D1	SUB3	
D0	ALLCALL	0-不响应0x70通用i2c地址。1-响应0x70通用i2c地址。这个芯片除了可以通过A5:A0自定义i2c地址外，还有一个通用i2c地址0x70，此寄存器可以控制是否响应这个通用地址。注意啊：这个寄存器的设置好像掉电会保存的！

各个通道的ON和OFF寄存器

总共16个通道，每个通道都有 LEDX_ON_L、LEDX_ON_H、LEDX_OFF_L、LEDX_OFF_H 四个寄存器。

系统中有一个12位的计数ACK，ACK根据PRE_SCALE寄存器设置的周期进行增加，没增加一次就会和上述四个寄存器对比：

当发现 ACK == LEDX_ON_H[3:0]:LEDX_ON_L 时，X通道输出高电平；

当发现 ACK == LEDX_OFF_H[3:0]:LEDX_OFF_L 时，X通道输出低电平。

PRE_SCALE寄存器

这个寄存器是用来设置周期的，具体原理可以不用管，只要记住这个公式：

$$prescale\ value = round\left(\frac{osc\_clock}{4096 \times update\_rate}\right) - 1$$

其中osc_clock是时钟，根据上面的寄存器设置选择是内部25MHz时钟还是外部时钟；update_rate是频率，比如周期是20ms，那么频率就是50。 注意：**实际应用中发现有误差，需要加入校准，要把udpate_rate乘以0.915。** 包括从网上下载的arduino驱动中也加入了此校准。

三、驱动

树莓派wiringPi平台

这里是基于树莓派wiringPi提供的i2c通信接口基础上实现的驱动，在其他平台上的驱动方法类似，只要把这里的i2c接口换成其他平台的通信接口即可。
本驱动周期固定为20ms不可变，如需修改也非常容易。

pca9685_wiringpi.h文件：

```
/*
  这个驱动是在树莓派的wiringPi基础上的，基于wiringPi对i2c的接口函数。
  此驱动的使用方法是：
  1、先用 pca9685_init(从机地址) 初始化，得到一个设备描述符（int型），这个设备描述符代表这个pca9685芯片，因为可能多个pca9685级联，通过这个设备描述符来区分它们。
     它的
  2、调用 pca9685_setmk
 */

#ifndef PCA9685_WIRINGPI_H
#define PCA9685_WIRINGPI_H
#include <wiringPi.h>

int pca9685_init(unsigned char addr);   // addr是7位的i2c从机地址，返回的是linux标准的设备描述符，调用它的地方视作pca9685的设备描述符
                                        //因为可以多个pca9685级联，通过设备描述符区别它们
                                        //此驱动仅作为驱动舵机使用，周期固定死位20ms，不允许外部设置
void pca9685_setmk(int fd, int num, int mk);    //设置指定通道的脉宽。fd是在pca9685_init时获得的设备描述符，num是通道号（从0开始），mk是脉宽单位是us。周期已经固定为20ms了

#endif
```

pca9685_wiringpi.c文件：

```c
#include "pca9685_wiringpi.h"

#define PCA9685_SUBADR1 0x2
#define PCA9685_SUBADR2 0x3
#define PCA9685_SUBADR3 0x4

#define PCA9685_MODE1 0x0
#define PCA9685_PRESCALE 0xFE

#define LED0_ON_L 0x6
#define LED0_ON_H 0x7
#define LED0_OFF_L 0x8
#define LED0_OFF_H 0x9

#define ALLLED_ON_L 0xFA
#define ALLLED_ON_H 0xFB
#define ALLLED_OFF_L 0xFC
#define ALLLED_OFF_H 0xFD

int pca9685_init(unsigned char addr)    // addr是7位的i2c从机地址，返回的是linux标准的设备描述符，调用它的地方视作pca9685的设备描述符
                                        //因为可以多个pca9685级联，通过设备描述符区别它们
                                        //此驱动仅作为驱动舵机使用，周期固定死位20ms，不允许外部设置
{
    int pca9685;
    pca9685 = wiringPiI2CSetup(addr);

    {   //初始化pca9685芯片
        double T = 20000;       //周期，单位是us
        unsigned char prescale;
        double osc_clock = 25000000;
        unsigned char oldmode, newmode;
        T /= 0.915;     //不知道为什么，会有所偏差，这里校准一下就ok了，从网上找的arduino代码也进行了校准。
        T /= 1000000;   //把T转换成秒
        prescale = (unsigned char)(osc_clock/4096*T - 1);
//      printf("prescale = 0x%x", prescale);
        oldmode = wiringPiI2CReadReg8(pca9685, PCA9685_MODE1);
        newmode = (oldmode&0x7f) | 0x10;        //准备进入sleep，设置时钟前必须先进入sleep模式
        wiringPiI2CWriteReg8(pca9685, PCA9685_MODE1, newmode);
        wiringPiI2CWriteReg8(pca9685, PCA9685_PRESCALE, prescale);
        oldmode &= 0xef;        //清除sleep位
        wiringPiI2CWriteReg8(pca9685, PCA9685_MODE1, oldmode);
        delay(0.005);
        wiringPiI2CWriteReg8(pca9685, PCA9685_MODE1, oldmode | 0xa1);
    }

    return pca9685;
```

```
}

void pca9685_setmk(int fd, int num, int mk)     //设置指定通道的脉宽。fd是在pca9685_init时获得的设备描述符，num是通道号（从0开始），mk是脉宽单位是us。周期已经固定为20ms了
{
    unsigned int ON, OFF;
    ON = 0;     //每次周期一开始就输出高电平
    OFF = (unsigned int)((((double)mk)/20000 * 4096)*1.0067114);
        //最后的1.0067114是校准用的
//    printf("off = 0x%x", OFF);

    wiringPiI2CWriteReg16(fd, LED0_ON_L+4*num, ON);
    wiringPiI2CWriteReg16(fd, LED0_OFF_L+4*num, OFF);
}
```

5
7
5
8

关于驱动在树莓派上的速度：

树莓派设置的i2c波特率	设置16路通道所用时间
100000	
1000000(1M)	2067us
2000000(2M)	1300us

四、使用流程

1、确定i2c地址

通过焊接A5~A0确定模块的i2c地址，如果不做任何焊接，默认地址是0x40。

2、连接数字电路电源。

3、连接两根i2c线。

4、连接v+引脚，给舵机供电电源。

5、把驱动合入到工程，即可使用。

更多内容，欢迎访问作者博客：http://nicekwell.net/