第8章　电机驱动模块

杭州艾研信息技术有限公司

2014 年 11 月

申明

杭州艾研信息技术有限公司保留随时对其产品进行修正、改进和完善的权利，同时也保留在不作任何通告的情况下，终止其任何一款产品的供应的权利。用户在下订单前应及时获取相关信息的最新版本，并验证这些信息是当前的和完整的。

可通过如下方式获取最新信息、技术资料和技术支持：

技术支持电话：0571-86134572

技术支持邮箱：support@hpati.com

产品&资料下载中心：http://www.hpati.com/products/

互动论坛：http://www.hpati.com/bbs/forum.php

公司地址：浙江省杭州市西湖区留和路16号新峰商务楼B402

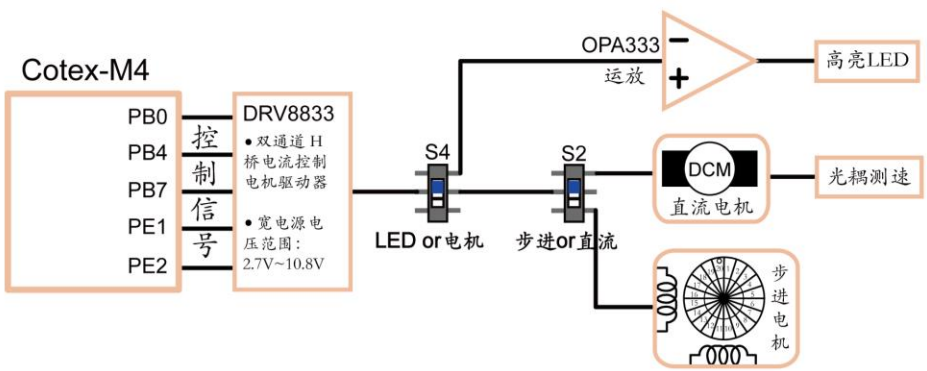
更多资讯请添加艾研信息官方微信（搜索公众号：艾研）或扫一扫下方二维码：

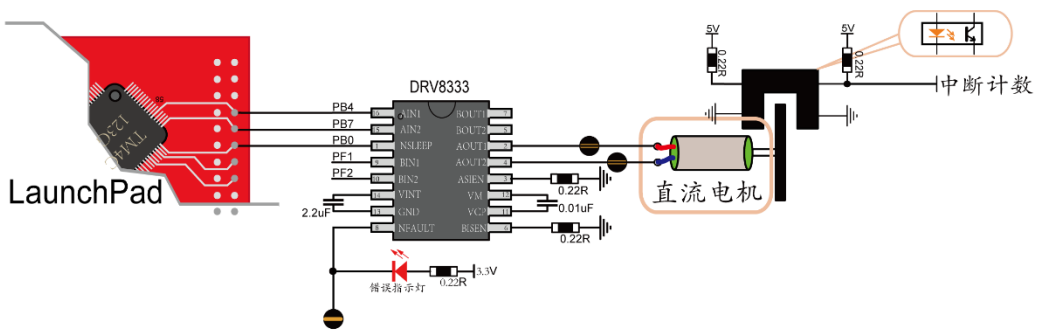
第8章　电机驱动模块

电机驱动模块介绍

实验简介

本模块采用一片双通道H桥电流控制电机驱动器DRV8833，它可以同时驱动两个直流电机或一个步进电机。在本实验中：1.开关S4打至右，S2打至右可驱动本模块右上方的步进电机 2.开关S4打至右，S2打至左可驱动本模块右下方的直流电机 3.开关S4打至左，P7短接时也可用于高亮LED模块的驱动。用户可通过软件改变DRV8833控制信号的占空比从而改变电机的转速或LED的亮度。




LaunchPad
DRV8333
PB4
PB7
PB0
PF1
PF2
AIN1
AIN2
NSLEEP
BIN1
BIN2
VINT
GND
NFAULT
BOUT1
BOUT2
AOUT1
AOUT2
ASEN
VM
VCP
BSEN
2.2uF
0.22R
0.01uF
0.22R
3.3V
0.22R
错误指示灯
5V
0.22R
5V
0.22R
直流电机
中断计数

直流电机的控制与测速

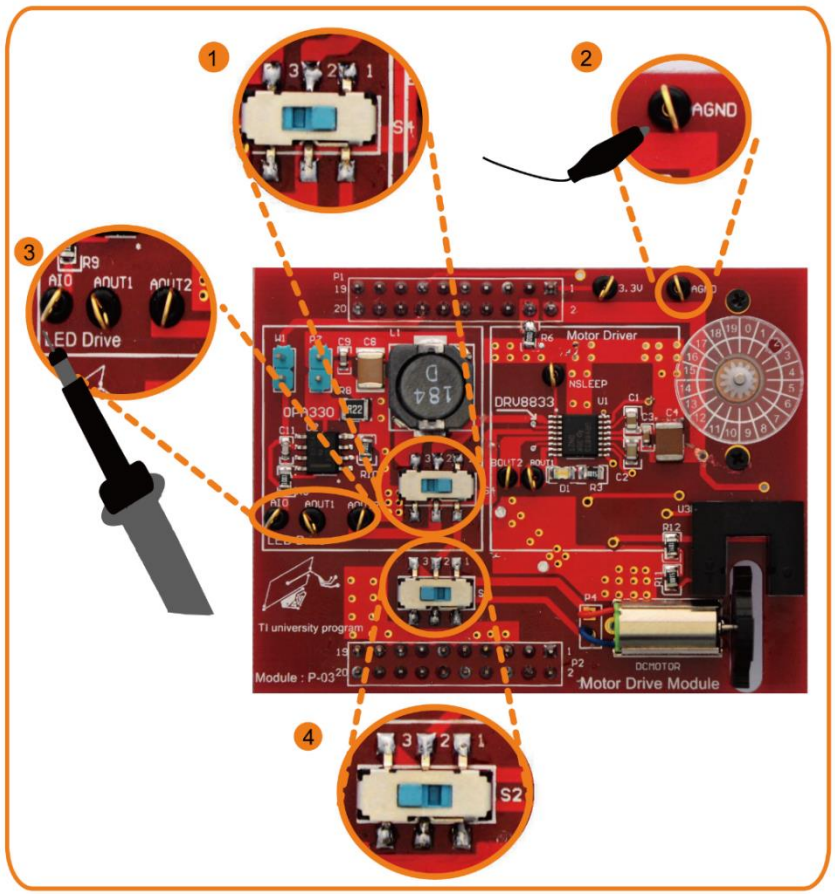
直流电机的控制与测速

1、理解原理图以后编写Launchpad代码，代码可参考网上资源。然后烧写代码。

2、在母板上TIVA、液晶、电机驱动模块连接完成，准备实验。

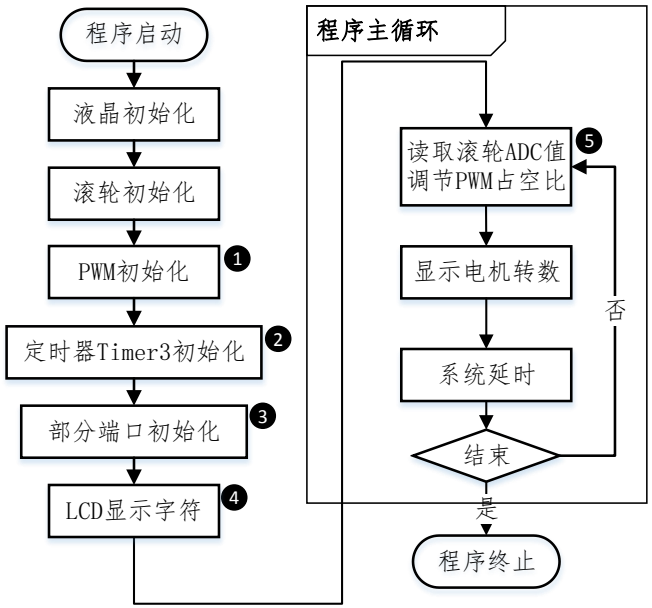
3、在电机驱动模块上完成开关的选择，如图 1 所示开关S4拨向右即连接了S4的1、2；如图 4 所示开关S2拨向左即连接了S2的3、2。

4、打开电源，调节液晶模块的滚轮，可看见直流电机的转速改变，同时用示波器观察图 3 所示的AOUT1、AOUT2、AIO等测量点的波形。

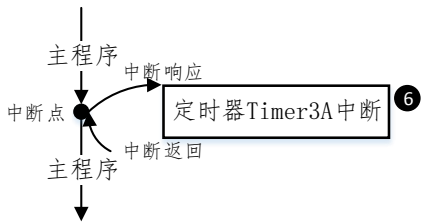


注意

连接仪表及跳线时断开电源。



图x、直流电机模块流程图



图x、直流电机模块中断响应

1. 初始化TIVA LaunchPad的PB4口输出一个频率20KHz的PWM信号用于驱动直流空心杯电机。

2. 电机运转后，电机轴上的带齿叶轮片会不断的穿过一对光耦对管。产生的高低电平变换的信号，由TIVA LaunchPad上的PB2口外部中断获得。通过计算中断响应周期可以测算出电机的转速并显示在LCD上。

3. 驱动电机除了需要PB4的PWM信号外，还需要一个配合PB7的置高信号，以及PB0的使能信号。这里将两个端口直接置高电平。

4. 液晶上显示“SPEED：xxx (r/s)”表示直流电机的旋转速度。

5. 滚轮的位置通过电压ADC采样传输到TIVA中，根据滚轮的ADC采样值调节PWM信号的占空比，可以改变电机的旋转速度。

6. 定时器Timer3A中断初始化为4个上升沿信号触发一次即正好电机叶轮转动了一圈。在前后两次中断时都读取一次SysTickValue，两次差值便是叶轮旋转一周的周期值。根据TIVA主频可获得电机的转速。

如上图所示，直流电机的驱动程序需要液晶、滚轮、Tiva 的 PWM 输出、定时器等多个模块的共同配合完成。其中液晶用于显示电机转数、滚轮用来调节 PWM 的占空比从而控制电机的转速、PWM 输出用于驱动直流电机旋转、而定时器则是用来检测电机的旋转数度。

关键代码分析

PWM 驱动信号的初始化

```
/*************************************************************************
 * 初始化PWM获取一路脉宽调制信号
 *      ————————|
//             |
 *      M4   PB4|-->M0PWM2          ---------Channel 1
//      _________|
*************************************************************************/
#define PERIOD_TIME         12500 / 60      // 20K Hz   //DC_motor
                                        // 60K Hz DC Source
// 定义最大最小周期时间   频率在200~1000之间
#define MAX_PERIOD          PERIOD_TIME * 90 / 100
#define MIN_PERIOD          PERIOD_TIME * 12 /  100
void Init_PWM()
{
    // 设置PWM时钟和系统时钟一致
    SysCtlPWMClockSet(SYSCTL_PWMDIV_1);

     // 使能PWM外设
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);

    // 使能外设端口
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

    //设置对应管脚的PWM信号功能
    GPIOPinConfigure(GPIO_PB4_M0PWM2);

    //设置PWM信号端口
    GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_4);

    //PWM生成器配置
    PWMGenConfigure(PWM0_BASE, PWM_GEN_1, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);

    //设置PWM信号周期
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, PERIOD_TIME);
```

```
    //设置PWM信号占空比
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2, PERIOD_TIME / 10);

    // 使能PWM输出端口
    PWMOutputState(PWM0_BASE, PWM_OUT_2_BIT, true);

    // 使能PWM生成器
    PWMGenEnable(PWM0_BASE, PWM_GEN_1);

    // 使能PWm生成器模块的及时功能.
    PWMSyncTimeBase(PWM0_BASE, PWM_GEN_1);

}
```

定时器 Timer3 初始化

```
/*************************************************************************
 * @brief    初始化Timer3A为边沿触发减计数中断。PB2作为外部中断源获取中断信号。
 *           捕获采用统计两路光耦信号之间的时间差来折算电机叶轮旋转数度
 * @param   null
 * @return  null

*************************************************************************/
void Init_Timer()
{
       // 启用Timer4模块
       SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER3);

       // 启用GPIO_M作为脉冲捕捉脚
       SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

       // 配置GPIO脚为使用Timer4捕捉模式
       GPIOPinConfigure(GPIO_PB2_T3CCP0);
       GPIOPinTypeTimer(GPIO_PORTB_BASE, GPIO_PIN_2);

       // 为管脚配置弱上拉模式
       GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_2,
GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

       // 配置使用Timer4的TimerA模块为边沿触发减计数模式
       TimerConfigure(TIMER3_BASE, TIMER_CFG_SPLIT_PAIR |
TIMER_CFG_A_CAP_COUNT);

       // 使用下降沿触发
       TimerControlEvent(TIMER3_BASE, TIMER_A, TIMER_EVENT_NEG_EDGE);
```

```
        // 设置计数范围为0x4~0X0
        TimerLoadSet(TIMER3_BASE, TIMER_A, 0x4);
        TimerMatchSet(TIMER3_BASE, TIMER_A, 0x0);

        // 注册中断处理函数以响应触发事件
        TimerIntRegister(TIMER3_BASE, TIMER_A, Timer3AIntHandler);

        // 系统总中断开
        IntMasterEnable();

        // 时钟中断允许，中断事件为Capture模式中边沿触发，计数到达预设值
        TimerIntEnable(TIMER3_BASE, TIMER_CAPA_MATCH);

        // NVIC中允许定时器A模块中断
        IntEnable(INT_TIMER3A);

        // 启动捕捉模块
        TimerEnable(TIMER3_BASE, TIMER_A);
}
```

关于中断函数的初始化和调用参考第三章中按键中断的实现部分，这里不再重复说明。初始化中的计算范围定在 4~0，即四次边沿触发即响应中断函数。是由于直流电机的叶轮上有四个对称的缺口，而在电机模块 PCB 中的光耦对管正好能捕获叶轮上的缺口产生一组信号。这组信号通过 PB2 获得，并触发中断函数的响应。

```
/*****************************************************************
 * @brief   光耦信号触发中断函数响应，通过主频和主频计数差值换算出电机的转速
 * @param   null
 * @return  null
 *****************************************************************/
// 记录前次和当前的系统计数值用于鉴别中断响应状况
uint32_t old_tick , cur_tick= 0;
// 计算前后两次中断的时间间隔，用于计算电机叶轮的频率
uint32_t tick_delay = 0;
// 记录前次和当前的电机叶轮的频率
float cur_frequency , old_frequency= 0.0;
void Timer3AIntHandler(void)
{
    unsigned long ulstatus;

   // 读取中断标志位
   ulstatus = TimerIntStatus(TIMER3_BASE, TIMER_CAPA_MATCH);

   if(ulstatus == TIMER_CAPA_MATCH)
```

```
    {
        //  获取当前系统时钟值
        cur_tick = ROM_SysTickValueGet();

        // 清除中断标志位
        TimerIntClear(TIMER3_BASE, ulstatus);

        // 因为减计数会自动停止，所以需要重新启用计数模块
        TimerEnable(TIMER3_BASE, TIMER_A);

        // 得到响应两次中断之间的系统时间计数差值
        if(old_tick > cur_tick)
            tick_delay = old_tick - cur_tick;

        //保留本次中断结束时的系统计数值，用于下次统计时间差的依据
        old_tick = cur_tick;
    }
}
```

得到了两次中断之间的系统计数值，变可以根据主频来换算出电机的运转频率，如下所示：

```
    // 系统主频 除以两次中断的系统计数值 = 频率
    cur_frequency = (1.0*TIVA_MAIN_FREQUENCY) / tick_delay;
```

剩下的只要通过 LCD 的驱动函数将换算得到的频率值显示在 LCD 即可。

数据测试与分析

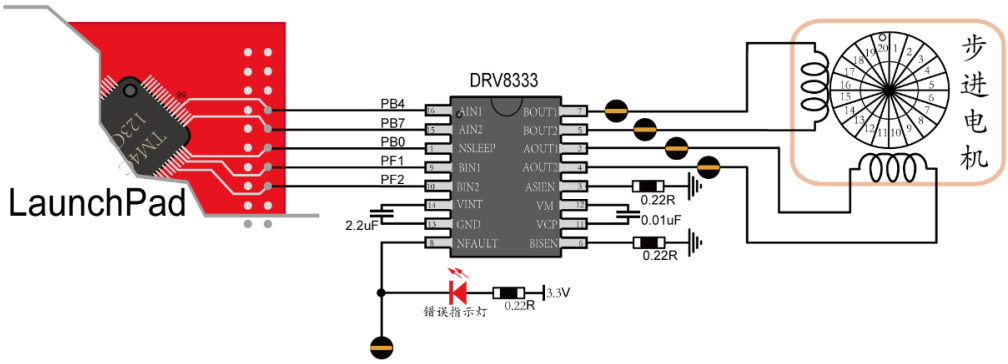
A. 步进电机的控制

实验内容

双极永磁步进电机的转动控制，不同的通电控制方式可以产生不同的步距角（单拍和双拍）；步进电机的反转

资源准备

等效原理图及电路分析



步进电机的控制

步进电机的控制

1、理解原理图以后编写Launchpad代码，代码可参考网上资源。然后烧写代码。

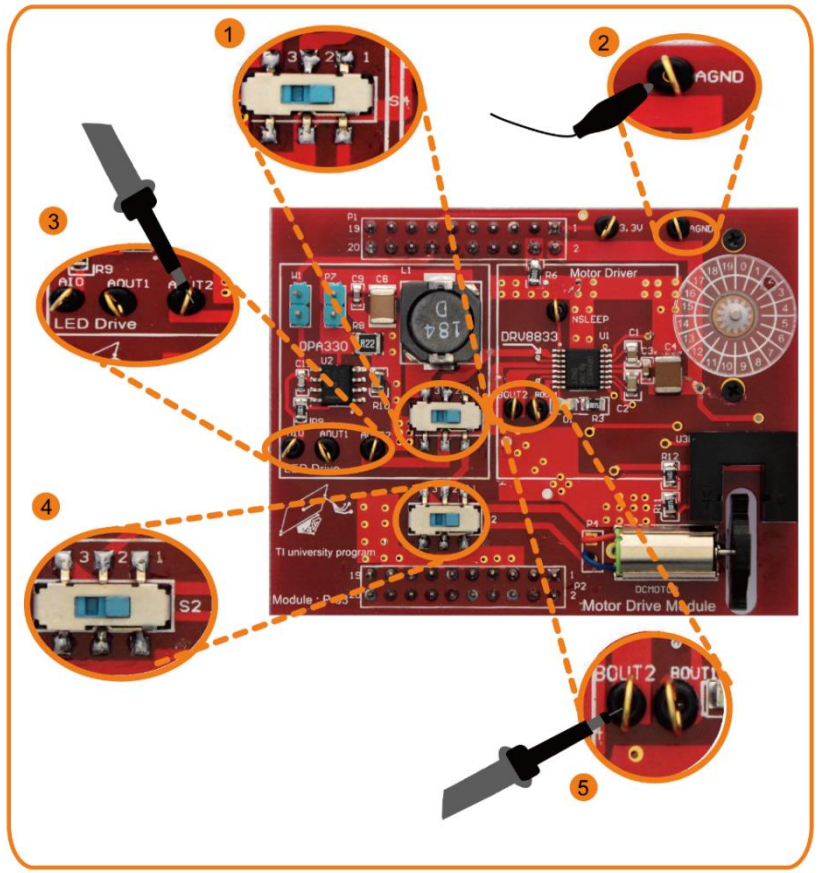
2、在母板上TIVA、液晶、电机驱动模块连接完成，准备实验。

3、在电机驱动模块上完成开关的选择，如图 1 所示开关S4拨向右；图 4 所示开关S2拨向右；

4、打开电源，调节液晶模块的滚轮，可看见直流电机的转速改变，同时用示波器观察图 3 所示的AOUT1、AOUT2、AIO等测量点的波形。

注意

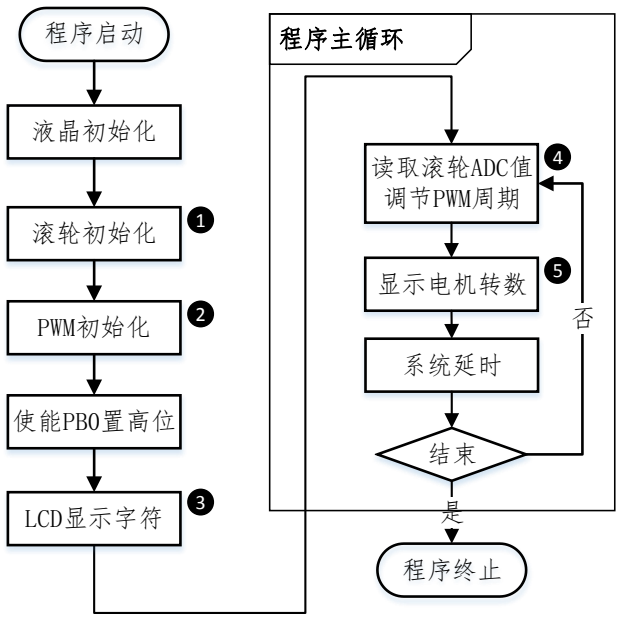
连接仪表及跳线时断开电源。



软件流程图及关键代码分析

软件流程图

步进电机的驱动程序和直流电机的驱动最大的差距在于直流电机只需一路 PWM，而步进需要四路 PWM 信号。软件流程如下图所示：



图x、步进电机模块流程图

1. 滚轮ADC采样值通过PE0端口读取，将PE0的外设设置为ADC。获取滚轮的ADC值后可以调节PWM的周期大小。（配置详细内容参看第三章相关内容）

2. 步进电机的驱动需要四组信号配合才能完成。初始化TIVA LaunchPad的PB7、PB4、PF1、PF2分别输出一组PWM方波。方波时序如下所示：

   PB7　1100110011001100
   PB4　0011001100110011
   PF1　0100010001000100
   PF2　0001000100010001

   他们的周期一致，其中PB7、PB4占空比50%而PF1、PF2占空比为25%。而且每组信号间的相位偏移都为25%。

3. 液晶上显示“SPEED：xxx　(r/s)”表示直流电机的旋转速度。

4. 滚轮的位置通过电压ADC采样传输到TIVA中，根据滚轮的ADC采样值调节驱动交流电机的PWM信号的周期，可以改变电机的旋转速度。

5. PWM信号执行一个周期即正好步进电机转动了一圈。所以电机的转数可以通过PWM信号的周期转化为电机的实际转速。

关键代码分析

1、降低主频：

```
    // 系统时钟设置
    SysCtlClockSet(SYSCTL_SYSDIV_64 | SYSCTL_USE_PLL |
SYSCTL_OSC_MAIN |
                   SYSCTL_XTAL_16MHZ);
```

由于步进电机的驱动PWM信号频率一般都在几十赫兹的水平上，而Tiva的常用工作频率在12.5MHz。所以在使用计时器做PWM波的生成时，我们采用系统时钟分频的方式降低Tiva主频来产生频率较低的PWM波形。

2、四路 PWM 波的初始化：

```
/*****************************************************************************
 * @brief   初始化PWM获取四组脉宽调制信号用于步进电机的驱动
 *      _________|
//      TIVA    |
 *      M4   PB7|-->M0PWM1         ---------Channel 1
//      M4   PB4|-->M0PWM2         ---------Channel 2
 *      M4   PF1|-->M1PWM5         ---------Channel 3
 *      M4   PF2|-->M1PWM6         ---------Channel 4
//      _________|
*****************************************************************************/
#define PERIOD_TIME        0x1FFFF
void Init_PWM()
{
    // 设置PWM时钟和系统时钟一致
    SysCtlPWMClockSet(SYSCTL_PWMDIV_1);

    // 使能PWM外设
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);

    // 使能外设端口
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //设置对应管脚的PWM信号功能
    GPIOPinConfigure(GPIO_PB7_M0PWM1);
    GPIOPinConfigure(GPIO_PB4_M0PWM2);
```

```
    GPIOPinConfigure(GPIO_PF1_M1PWM5);
    GPIOPinConfigure(GPIO_PF2_M1PWM6);

    //设置PWM信号端口
    GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_4 | GPIO_PIN_7);
    GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2);

    //PWM生成器配置
    PWMGenConfigure(PWM0_BASE, PWM_GEN_0, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);
    PWMGenConfigure(PWM0_BASE, PWM_GEN_1, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);
    PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);
    PWMGenConfigure(PWM1_BASE, PWM_GEN_3, PWM_GEN_MODE_UP_DOWN |
PWM_GEN_MODE_NO_SYNC);

    //设置PWM信号周期
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, PERIOD_TIME);
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, PERIOD_TIME);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, PERIOD_TIME);
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, PERIOD_TIME);

    //设置PWM信号占空比
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, PERIOD_TIME / 2);
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2, PERIOD_TIME / 2);
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, PERIOD_TIME / 4);
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, PERIOD_TIME / 4);

    // 使能PWM输出端口
    PWMOutputState(PWM0_BASE,  PWM_OUT_1_BIT | PWM_OUT_2_BIT , true);
    PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT | PWM_OUT_6_BIT, true);

    // 使能PWM生成器
    PWMGenEnable(PWM0_BASE, PWM_GEN_0);
    PWMGenEnable(PWM0_BASE, PWM_GEN_1);
    PWMGenEnable(PWM1_BASE, PWM_GEN_2);
    PWMGenEnable(PWM1_BASE, PWM_GEN_3);

    // 延时同步四路PWM信号
    PWMSyncTimeBase(PWM0_BASE, PWM_GEN_1_BIT);
    SysCtlDelay((PERIOD_TIME / 2) );
    PWMSyncTimeBase(PWM0_BASE, PWM_GEN_0_BIT);
    SysCtlDelay((PERIOD_TIME * 7/ 8));
```

```
    PWMSyncTimeBase(PWM1_BASE, PWM_GEN_3_BIT);
    SysCtlDelay((PERIOD_TIME / 2));
    PWMSyncTimeBase(PWM1_BASE, PWM_GEN_2_BIT);
}
```

完成了PWM的初始化后，步进电机便能转动了。

3、步进电机转速滚轮调节：

在程序的出循环中假如滚轮调节PWM周期的逻辑可调节电机的转动速度。主要代码如下所示：

```
/*************************************************************************
while(1)
    {
        ADCProcessorTrigger(ADC_BASE, SequenceNum);

        // 等待完成取样转换
        while(!ADCIntStatus(ADC_BASE, SequenceNum, false))
        {
        }

        // 清楚ADC中断标志位
        ADCIntClear(ADC_BASE, SequenceNum);

        // 读取ADC采样值
        ADCSequenceDataGet(ADC_BASE, SequenceNum, pui32ADC0Value);

        // 当前周期转化公式
        cur_Period = MIN_PERIOD + ((MAX_PERIOD - MIN_PERIOD) *
pui32ADC0Value[0]) / 4096;

        // 记录ADC的变化率大小
        uint32_t temp = 0;

        if(cur_Period > old_Period)
        {
            temp = cur_Period - old_Period;
        }else{
            temp = old_Period - cur_Period;
        }

        // ADC 实现16级的有极变化，
        if(temp > 0xFFF)
        {
            // 调整周期
```

```
        PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, cur_Period);
        PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, cur_Period);
        PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, cur_Period);
        PWMGenPeriodSet(PWM1_BASE, PWM_GEN_3, cur_Period);

       // 调整占空比
        PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, cur_Period / 2);
        PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2, cur_Period / 2);
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, cur_Period / 4);
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_6, cur_Period / 4);

        // 延时调整
        PWMSyncTimeBase(PWM0_BASE, PWM_GEN_1_BIT);
        SysCtlDelay((cur_Period / 2) );
        PWMSyncTimeBase(PWM0_BASE, PWM_GEN_0_BIT);
        SysCtlDelay((cur_Period * 7/ 8));
        PWMSyncTimeBase(PWM1_BASE, PWM_GEN_3_BIT);
        SysCtlDelay((cur_Period / 2));
        PWMSyncTimeBase(PWM1_BASE, PWM_GEN_2_BIT);

        //计算电机转数
        speed = ROM_SysCtlClockGet() / cur_Period;

        old_Period = cur_Period;
    }
```

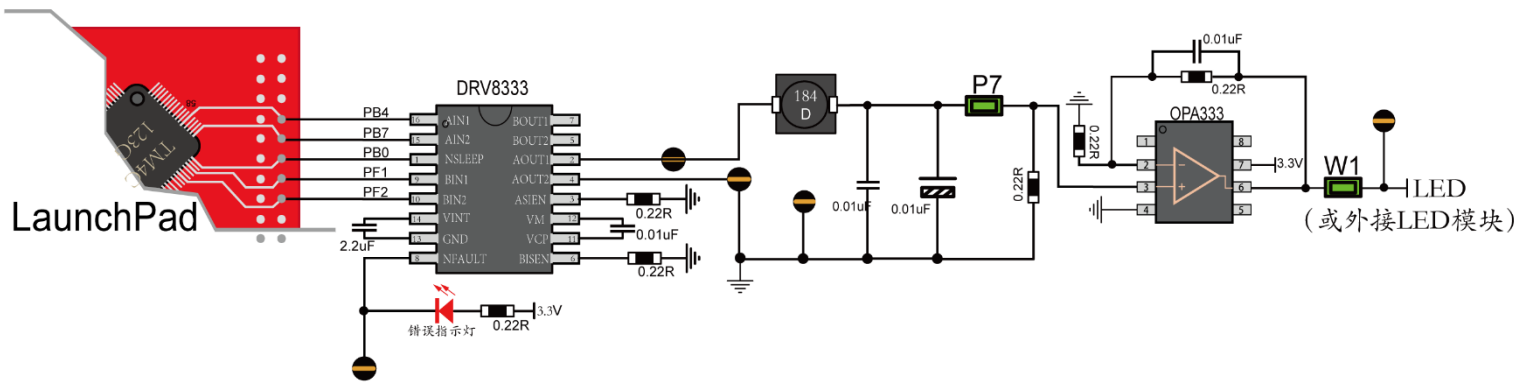
B. 高亮 LED 的驱动与电流检测

实验内容

应用电机控制芯片内部的 H 桥式电路，辅以外围电流检测电路和单片机反馈控制可以形成一个数字电源。应用这个电源组成一个高亮 LED 的驱动电路

资源准备

等效原理图及电路分析



高亮LED的驱动与电流检测

高亮LED的驱动与电流检测

1、理解原理图以后编写Launchpad代码，代码可参考网上资源。然后烧写代码。

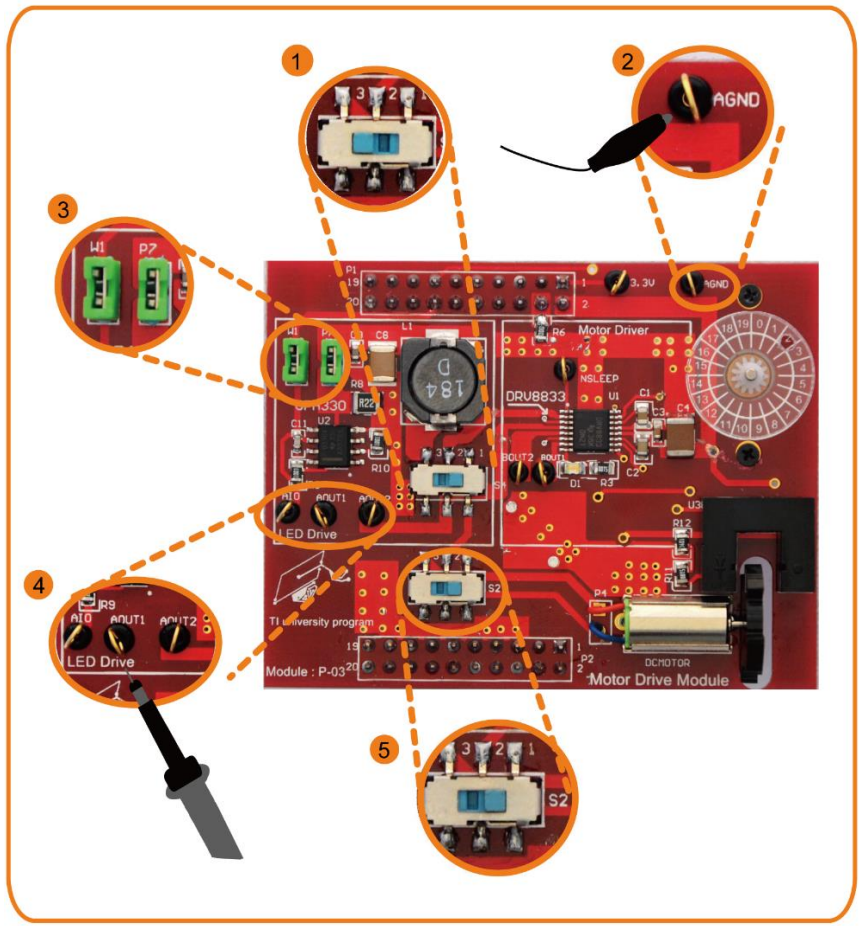
2、在母板上TIVA、液晶、电机驱动模块连接完成，准备实验。

3、在电机驱动模块上完成开关的选择，如图 1 所示开关S4拨向左；图 5 所示开关S2任意；图 3 跳线P7短接；图 3 跳线W1可短接可外接高亮LED模块。

4、打开电源，调节液晶模块的滚轮，可看见直流电机的转速改变，同时用示波器观察图 4 所示的AOUT1、AOUT2、AIO等测量点的波形。

注意

连接仪表及跳线时断开电源。



软件流程图及关键代码分析

本实验和直流电机控制与测速使用同样的程序，软件流程和关键代码分析就不再赘述。