第6章 DC-DC 升压模块

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

第6章 DC-DC 升压模块

DC-DC Boost 实验模块介绍

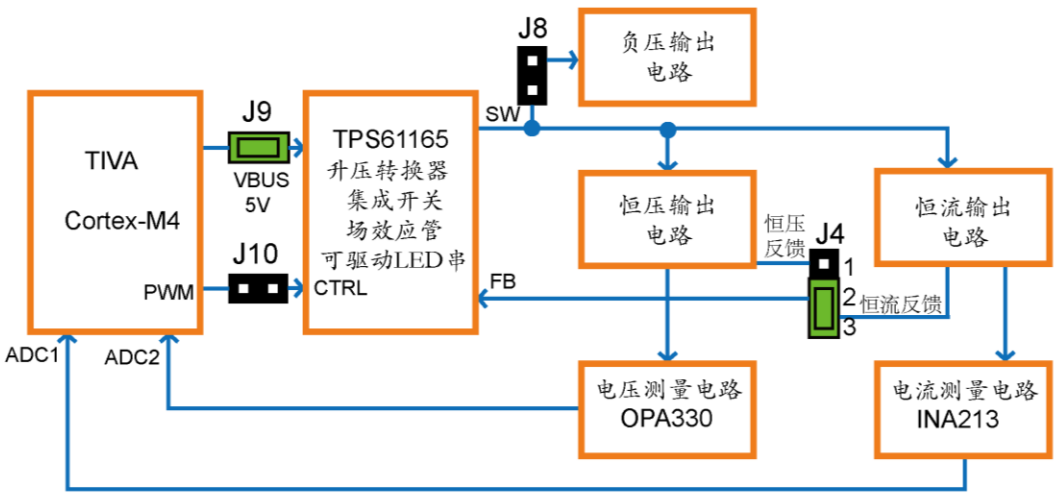
实验目的:

1、理解并掌握boost电源拓扑。
2、理解并掌握输出电压的调节方式。
3、理解带反馈和无反馈电路的差异。
4、LED恒流驱动的原理
5、理解并掌握各类电源参数测量方法：效率，纹波，软启动时间，电感电压与电流的变化。
6、理解电源设计的PCB布线的基本原则。

实验简介:

该实验以TPS61165为中心，通过跳线的变化连接，实现LED恒流驱动、恒压源输出及负压输出。通过TIVA产生PWM控制TPS61165的CTRL端，实现恒流源和恒压源输出大小的控制。

电流电压检测部分：电压测量电路，以OPA330为中心，起到电压跟随的作用，使TIVA能更准确地采到实际电压值。电流检测电路，以INA213为中心，实现高测电流检测。



原理框图

DC-DC Boost 模块布局及电路板跳线物理位置展示

■ PCB布局

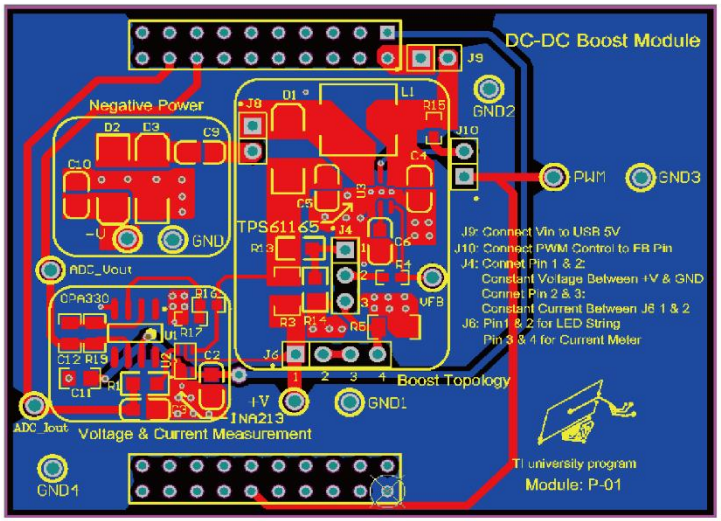


图6-1 PCB布局图

■ 实物图

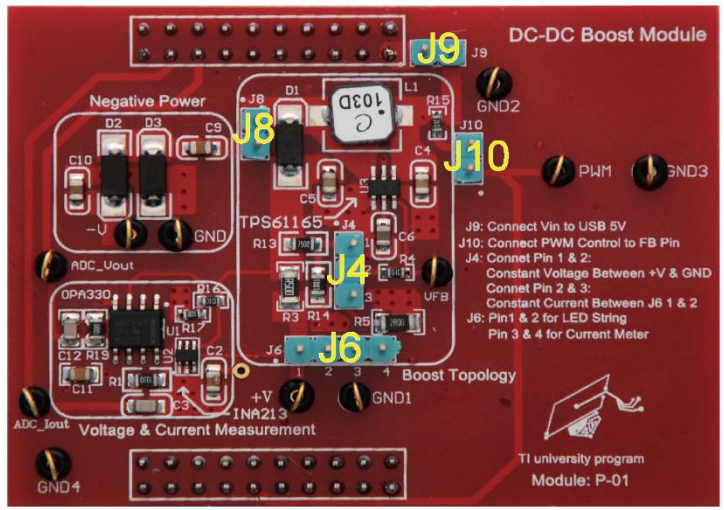


图6-2 实物图

设计考虑

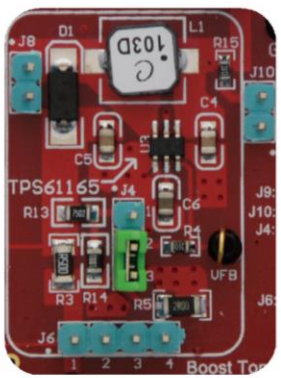
1、通过跳线选择恒压输出和恒流输出，帮助理解恒压和恒流输出在电路上的异同，和理解反馈电路中的放大器。

2、通过高侧电流检测，把放大器和电源结合在一起；

3、通过控制CTRL端口，实现输出电压/电流控制；

4、了解无反馈负压输出的原理和应用

恒流源驱动LED串



电路实物图

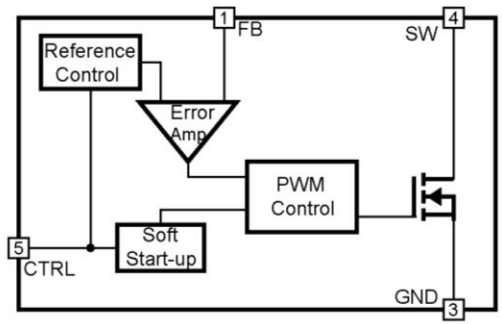
开关电源是用半导体器件作为开关，将一种形式的电源转换成另一种形式的电源。其特点是频率高、功耗低、工作效率高、体积小、输入范围宽，通过闭环系统调节，使输出电压保持稳定。

利用TPS61165搭建的电路是boost型开关电源电路。开关电源芯片TPS61165内部有一个MOSFET开关器件，它的开关频率达到1.2MHz。在MOSFET关断时，电源和电感L1给电容C5充电；当MOSFET导通时，相当于将L1的一端接地，L1释放能量。和boost型电路工作原理一样，通过电感L1和电容C5充放电，实现电压的转换。同时，该电路在FB端形成闭环通路，可以实现电压的稳定输出。

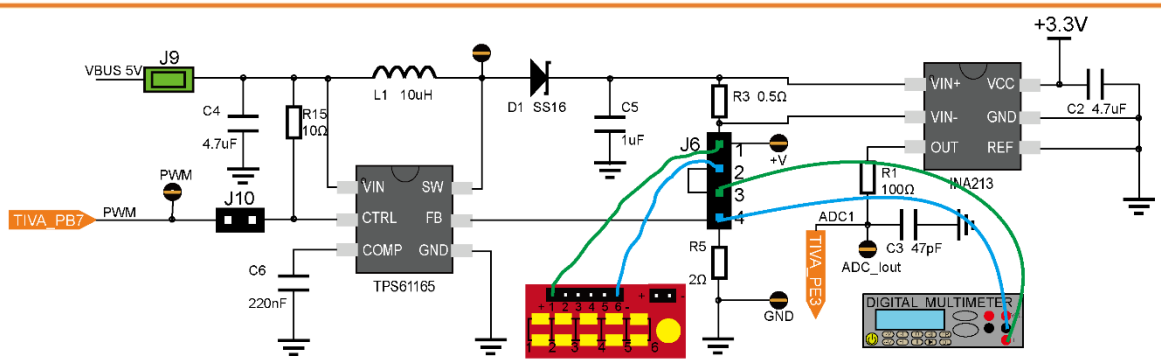
电压调节方式：

在本实验中通过PWM控制CTRL起到电压调节。在芯片的CTRL端直接送入一定占空比的PWM波形，改变芯片内部误差放大器的输入电压值，从而改变电源电压的大小。这种控制的方法简单容易实现，改变占空比即可调节电压的大小。

电流检测电路：此电路中采用高测电流测量。INA213正好是共模高，失调电压小，温度漂移特性好的运算放大器。



TPS61165内部原理图



恒流测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Boost(升压)模块，准备实验。

3、如图 1 所示，短接J9及J4的2,3。完成恒流测试的硬件配置。

4、如图 2 所示，连接负载及电流表，注意负载数不少于2对。

5、如图 2 所示，进行仪表配置。测J8观测开关节点电压波形，测+V，GND1观测输出电压的纹波，测ADC_Iout观测电流测量电路最后转化后的电压。

6、给TIVA上电，可以观察到LED点亮，在LCD模块及电流表上可以看到电流值大小。

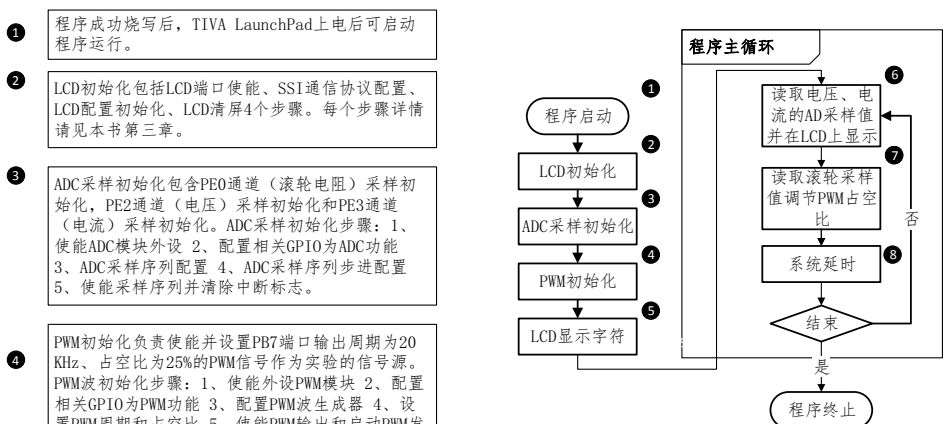
7、断电后，改变负载，即改变串入的灯数。观察液晶和电流表上显示的电流值。了解恒流情况。

注意

- 连接仪表及跳线时，断开电源。
- 如果没有电流表，可以如图1所示，将接口进行短接即可进行实验。

程序通过 PB7 口产生 PWM 波控制 tps61165 工作，完成 LED 串点亮以及电流控制，可通过 LCD 开发板上的滚轮调节 PWM 波的占空比进而实现电流的调节；通过 PE2 的 ADC 功能读取电压测量电路中的 OPA330 的输出电压，并根据电路原理图计算出测量点的电压值，显示在 LCD 上；通过 PE3 的 ADC 功能读取电流测量电路中的 INA213 的输出，并根据电路原理图以及 INA213 的放大倍数（50 倍）计算得出测量点的电流值，显示在 LCD 上。

软件流程图及关键代码分析



❶ 程序成功烧写后，TIVA LaunchPad上电后可启动程序运行。

❷ LCD初始化包括LCD端口使能、SSI通信协议配置、LCD配置初始化、LCD清屏4个步骤。每个步骤详情请见本书第三章。

❸ ADC采样初始化包含PE0通道（滚轮电阻）采样初始化，PE2通道（电压）采样初始化和PE3通道（电流）采样初始化。ADC采样初始化步骤：1、使能ADC模块外设 2、配置相关GPIO为ADC功能 3、ADC采样序列配置 4、ADC采样序列步进配置 5、使能采样序列并清除中断标志。

❹ PWM初始化负责使能并设置PB7端口输出周期为20 KHz、占空比为25%的PWM信号作为实验的信号源。PWM波初始化步骤：1、使能外设PWM模块 2、配置相关GPIO为PWM功能 3、配置PWM波生成器 4、设置PWM周期和占空比 5、使能PWM输出和启动PWM发生器的定时器。

❺ 在LCD上显示LED_Current：xx mA，LED_Volgate：xx V等信息，便于实验过程中观察实验数据变化。

❻ 读取PE2（ADC_IN，电压）采样值，根据转换公式计算电压值进行LCD显示，读取PE3（CUR_ADC）采样值根据转换公式计算电流值进行显示。ADC采样值读取程序部分可参考本书第三章。

❼ 读取滚轮电阻采样值，计算新占空比，若当前占空比跟新计算得出的占空比相差1%则新占空比更新至系统使用。占空比控制在5%-95%之间。滚轮电阻获取ADC采样值的程序部分详见本书第三章。

❽ 为防止LCD过快重复刷新，系统进行延时将主循环控制在一个合理的执行速度内。

图 xx 程序流程图

此次实验程序中需要涉及 LCD 功能模块，ADC 功能模块，PWM 功能模块。其中 LCD 功能模块和 ADC 功能模块的实现可参考本书第三章内容。

PWM 功能模块实现

使用 PWM 波首先需要初始化相应功能模块，初始化过程如程序流程图中所述，PWM 设置中较为关键的是周期的设置，周期设置由 ROM_PWMGenPeriodSet 或者 PWMGenPeriodSet 设置完成，ROM 版本和非 ROM 版本的函数实现功能相同且参数也相同，函数原型为：

```
void PWMGenPeriodSet(uint32_t ui32Base, uint32_t ui32Gen,
                     uint32_t ui32Period);
```

其中 ui32Period 需要计算得到，该值表示 PWM 发生器输出的周期，使用时钟节拍来测量，可通过如下方式计算：

$$period = \frac{SysPWM\_Fre}{OutputPWM\_Fre}$$

$SysPWM\_Fre$ 为当前系统 PWM 时钟频率，该时钟频率通过函数 SysCtlPWMClockSet()设置。其函数原型为：

```
void SysCtlPWMClockSet(uint32_t ui32Config);
```

ui32Config 是 PWM 时钟的配置，它必须是下面的其中一个值：SYSCTL_PWMDIV_1、SYSCTL_PWMDIV_2、SYSCTL_PWMDIV_4、SYSCTL_PWMDIV_6、SYSCTL_PWMDIV_8、SYSCTL_PWMDIV_16、SYSCTL_PWMDIV_32、SYSCTL_PWMDIV_64。

这个函数提供给 PWM 模块的时钟频率作为一个处理器时钟的系数来设置，PWM 模块使用这个时钟来产生 PWM 信号，它的速率形成所有 PWM 信号的基础。处理器时钟有函数 SysCtlClockSet()设置，故 PWM 的时钟由 SysCtlClockSet()配置的系统时钟速率决定。例如：系统时钟设置为 12.5MHz，PWM 时钟设置为 SYS_PWMDIV_1，而 PWM 输出信号频率为 20KHz，可计算得出 PWM 输出信号的时钟节拍值为：

$$period = \frac{SysPWM\_Fre}{OutputPWM\_Fre} = \frac{12.5\times10^6/1}{20\times10^3} = \frac{12500}{20} = 625$$

程序代码如下：

```
 * @brief   初始化PWM
 * @param   none
 * @return  none
 *      ________
 *             |
 *      M4   PB7|-->M0PWM1
 *      ________|
 *
 * PERIOD_TIME计算：系统时钟使用12.5MHz（主函数中设置），需要产生20KHz的
 * PWM波，PWM周期就是（1/20000）,对于12.5MHz的时钟来说就是12500000*1/20000
 * 个时钟节拍，也就是约去1000后12500/20的值。
 ************************************************************************/
#define PERIOD_TIME        12500 / 20               //20KHz ,
void Init_PWM(void)
{的时钟，1分频
    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_1);

    //使能PWM0外设模块
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
    //使能PB端口
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    //配置GPIO复用功能为PWM模式
    ROM_GPIOPinConfigure(GPIO_PB7_M0PWM1);
    //配置PB7供PWM外设使用
```

```
    ROM_GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_7);
    //配置PWM发生器，PWM0模块的,PWM_GEN_0（发生器0）向下计数，立即更新参数
    ROM_PWMGenConfigure(PWM0_BASE, PWM_GEN_0,
           PWM_GEN_MODE_UP_DOWN | PWM_GEN_MODE_NO_SYNC);
    //PWM周期设置，20KHz
    ROM_PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, PERIOD_TIME);
    //设置占空比为25%的PWM1的脉宽
    ROM_PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1,PERIOD_TIME / 4);

    //使能输出
    ROM_PWMOutputState(PWM0_BASE, PWM_OUT_1_BIT, true);
    //启动发生器0的定时器
    ROM_PWMGenEnable(PWM0_BASE, PWM_GEN_0);
    //使能发生器模块计数器同步
    ROM_PWMSyncTimeBase(PWM0_BASE, PWM_GEN_0);
}
```

程序中通过 LCD 开发板上的滚轮电阻调节 PWM 波的占空比，而滚轮电阻数值是通过 Tiva M4 上的 ADC 模块采样得到，得到滚轮当前值后就可以计算出当前 PWM 波的占空比。其计算公式如下（占空比控制在 5%~95% ）：

$$Cur\_Duty = \frac{ADCWheel\_Value}{ADCMax} \times 90 + 5$$

式中 $ADCWheel\_Value$ 为当前滚轮电阻的采样值， $ADCMax$ 为滚轮电阻采用的 AD 模块中的最大值，Tiva M4 中的 AD 为 12-bit，故该值为 4096， $(ADCWheel\_Value / ADCMax)$ 的值为 0~1 之间，故 $Cur\_Duty \in [5,95]$ 。

根据计算公式可以得出如下程序：

```
//根据滚轮采样值计算占空比
uint32_t cur_Duty = 5 + (90 * pui32ADCWheel_Value) / ADCMAX;
uint32_t period = cur_Duty * PERIOD_TIME / 100;

//改变PWM波的占空比
ROM_PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, PERIOD_TIME);
ROM_PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, period);
```

PERIOD_TIME 也就是通过#define 宏定义的值，该值的表示的是 PWM 波的周期，计算得出的占空比乘以该值即表示这段时间内保持高电平输出，余下时间保持低电平输出。该控制功能由 TivaWare 库中的函数 ROM_PWMGenPeriodSet 和 ROM_PWMPulseWidthSet 配合完成。

实验的程序设计中需要在 LCD 上显示当前电流的电流值和电压值，电流值和电压值都是采用 ADC 采样获得，ADC 采样配置以及获取采样值跟本书第三章中的滚轮控制部分相似，故该部分可参考第三章完成。获取 ADC 采样值后需要根据实际电路计算得出实际值。

a. 电流

电流检测通过 Tiva M4 的 PE3 口的 ADC 功能采样电流测量电路中 INA213 的输出值。LCD 显示的是 INA213 输入端的电流值（电路原理图中的+IN、-IN 端，即通过 R_3 电阻的电流值。INA213 本身具有 50 倍的增益。电流计算公式如下：

$$Current = \frac{CUR\_ADC \times ADCRef}{ADCMax \times 50 \times R_3}$$

根据原理图可知 $ADCRef = 3.3V$，$R_3 = 0.5\Omega$，$ADCMax = 2^{12} = 4096$。

$$Current = \frac{CUR\_ADC \times 3.3}{4.96 \times 50 \times 0.5} \times 1000 = \frac{CUR\_ADC \times 3.3 \times 40}{4096} mA$$

在程序实现中将*Current*扩大 1000 倍，以方便 LCD 显示模块使用。程序代码如下

```
unsigned long CUR_ADC_Sample = (pui32CUR_ADC_Value *
                                                3300 * 40) / ADCMAX;
```

b. 电压

电压检测通过 Tiva M4 的 PE2 口的 ADC 功能采样电压测量电路中 OPA330 的输出值。而在 LCD 上显示的却是 R_{16} 和 R_{17} 之间的电压，OPA330 的输入端为 TO_BUF（可见原理图），根据分压可以计算出电压值。计算公式如下：

$$Volt = \frac{To\_Buf}{ADCMax} \times ADCRef \times \frac{R_{16} + R_{17}}{R_{17}}$$

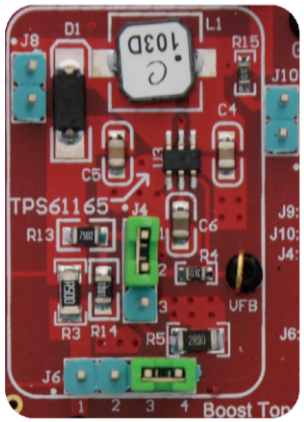
根据原理图可知 $ADCRef = 3.3V$，$ADCMax = 2^{12} = 4096$，$R_{16} = 10k$，$R_{17} = 2k$。

$$Volt = \frac{To\_Buf \times 3.3 \times 6}{4096} \times 1000mV$$

电压值计算成 mV 方便 LCD 显示模块使用，程序代码如下

```
unsigned long ADC_IN_Sample = (pui32ADC_IN_Value * 3300 * 6) / ADCMAX;
```

恒压源输出



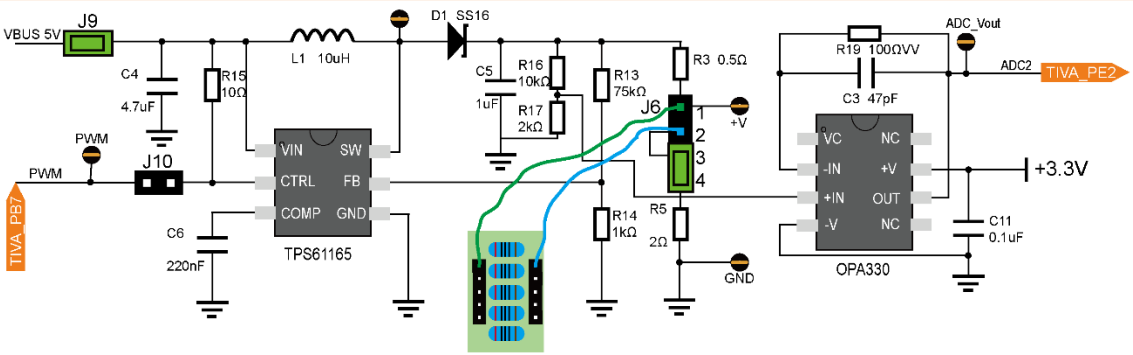
实物图

恒压源输出是开关电源能输出稳定的电压，且电压不会随着负载的变化而变化。需要注意的是在恒流控制点亮LED时，负载是LED串。但是，在恒压源输出时负载不能仅仅是LED串，这是因为二极管LED会使输出的电压发生钳位，导致电源芯片偏离了正常工作的轨道。因此，用功率电阻作为负载进行试验，测试恒压源的输出和调节恒压输出的大小。

恒压输出的实现是：在反馈端FB和地之间接入电阻R14，则R14两端的电压被控制在一个设定值。由于FB端是运放的反相输入端，故其输入/输出电流几乎为零，这样，流过R14的电流就可以计算出来。在R14与R1之间接入电阻R13，把电压抬高。因为FB端的电压恒定，流过R13和R14的电流就恒定，所以，在R13和R14两端产生电压也恒定，从而实现了恒压源的输出。

电压检测电路：

由于输出电压比较大，单片机难以直接采集数据，需要进行分压。但是分压后的信号，其输出阻抗比较大，若直接由单片机采样，可能造成信号的衰减，影响测量精度。为此，增加一级由OPA330运放搭建的电压跟随电路，以避免对ADC输入阻抗的影响，提高采样精度。在电压跟随电路中，在运放的反相输入端和输出之间并联一个电阻和一个电容，使电压跟随电路工作更加稳定。



恒压测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Boost(升压)模块，准备实验。

3、如图 1 所示，短接J9、J4的1,2及J6的3,4。完成恒压测试的硬件配置。

4、连接负载，如图 1 所示，注意负载功率及阻值大小。

5、如图 2 所示，进行仪表配置。测J8观测开关节点电压波形，测+V，GND1观测输出电压的纹波，测ADC_Vout观测电压测量电路最后转化后的电压。

6、给TIVA上电，在LCD模块上可以看到测量得到的电压值。

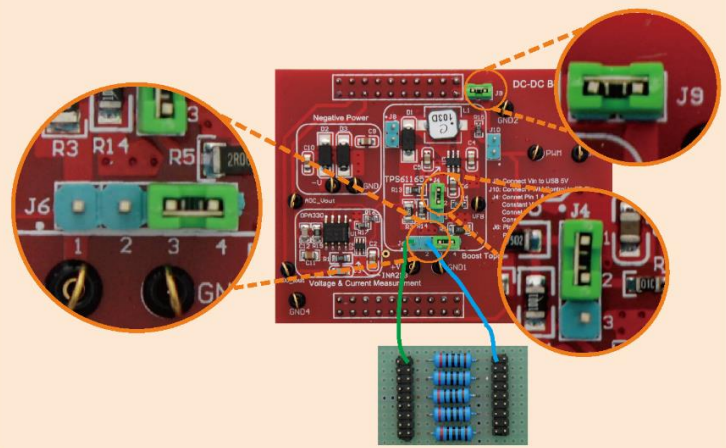
7、变换负载的大小，注意受USB供电功率限制，负载的大小不得低于100Ω。

注意

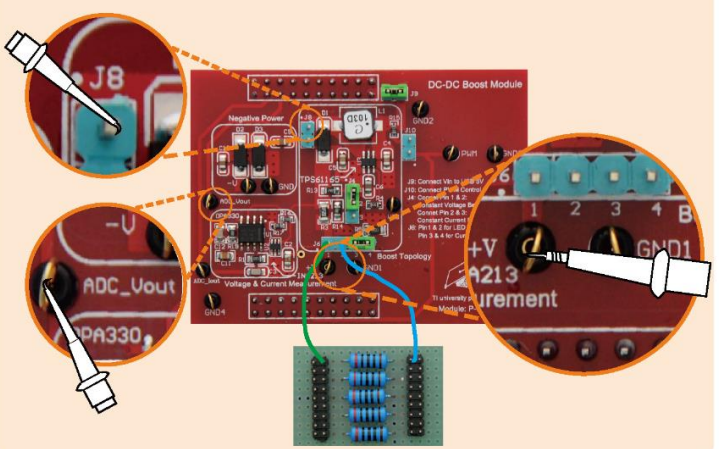
- 连接仪表及跳线时，断开电源。
- 输出电压测量过程中，测纹波需要将示波器切换到交流模式，而测电压值则为直流模式。
- 实验套件中没有提供电阻负载，实验中的负载需要自己制作。这里的负载是阻值100Ω、功率1W的电阻。通过电阻的串并联改变负载大小。

恒压测试主要步骤

1



2



电压调节测试主要步骤

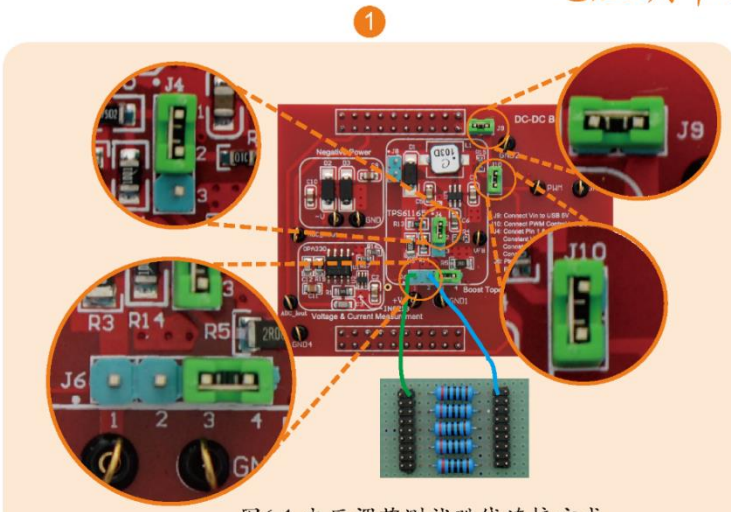


图6-1 电压调节测试跳线连接方式

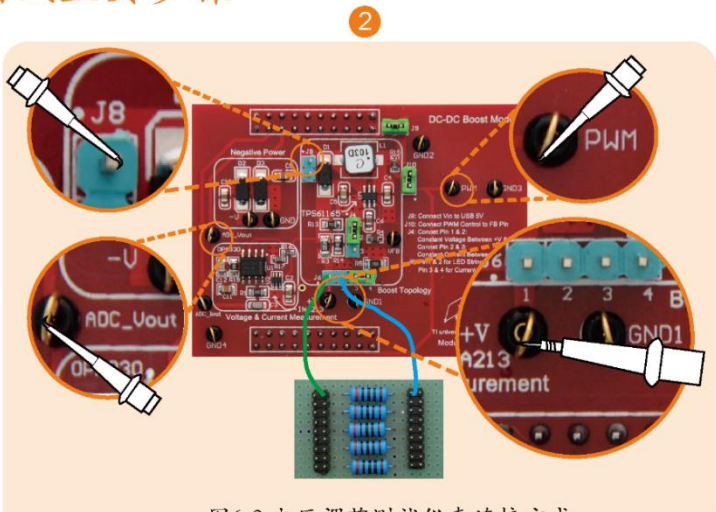


图6-2 电压调节测试仪表连接方式

电压调节测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Boost(升压)模块，准备实验。

3、如图1所示，短接J9、J4的1,2、J6的3,4及J10。完成电压调节测试的硬件配置。

4、连接负载，如图1所示，注意负载功率及阻值大小。

5、如图2所示，进行仪表配置。测J8观测开关节点电压波形，测+V，GND1观测输出电压的纹波，测ADC_Vout观测电压测量电路最后转化后的电压。测PWM观测TIVA输出的PWM波形。

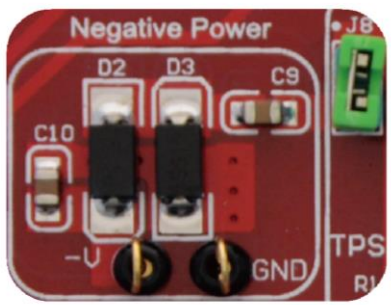
6、给TIVA上电，在LCD上可以看到测量得到的电压值。

7、拨动滚轮，记录PWM的占空比和对应的输出电压值，可以发现他们之间的控制关系。

注意

- 连接仪表及跳线时，断开电源。
- 输出电压测量过程中，测纹波需要将示波器切换到交流模式，而测电压值则为直流模式。
- 实验套件中没有提供电阻负载，实验中的负载需要自己制作。这里的负载是阻值100Ω、功率1W的电阻。通过电阻的串并联改变负载大小。

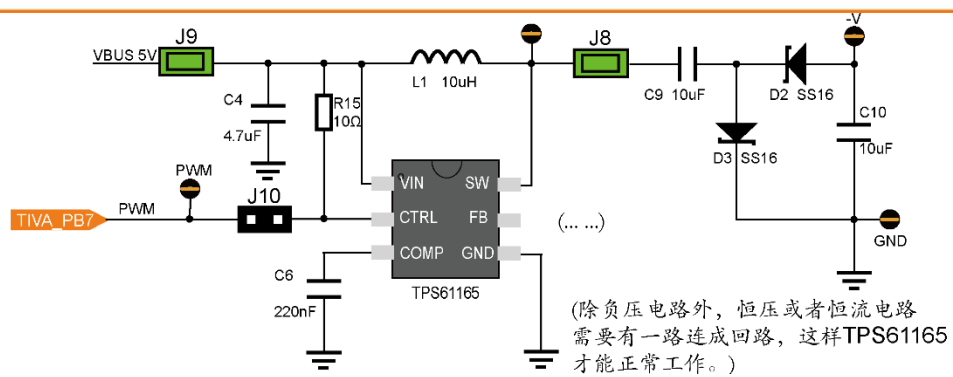
负压生成

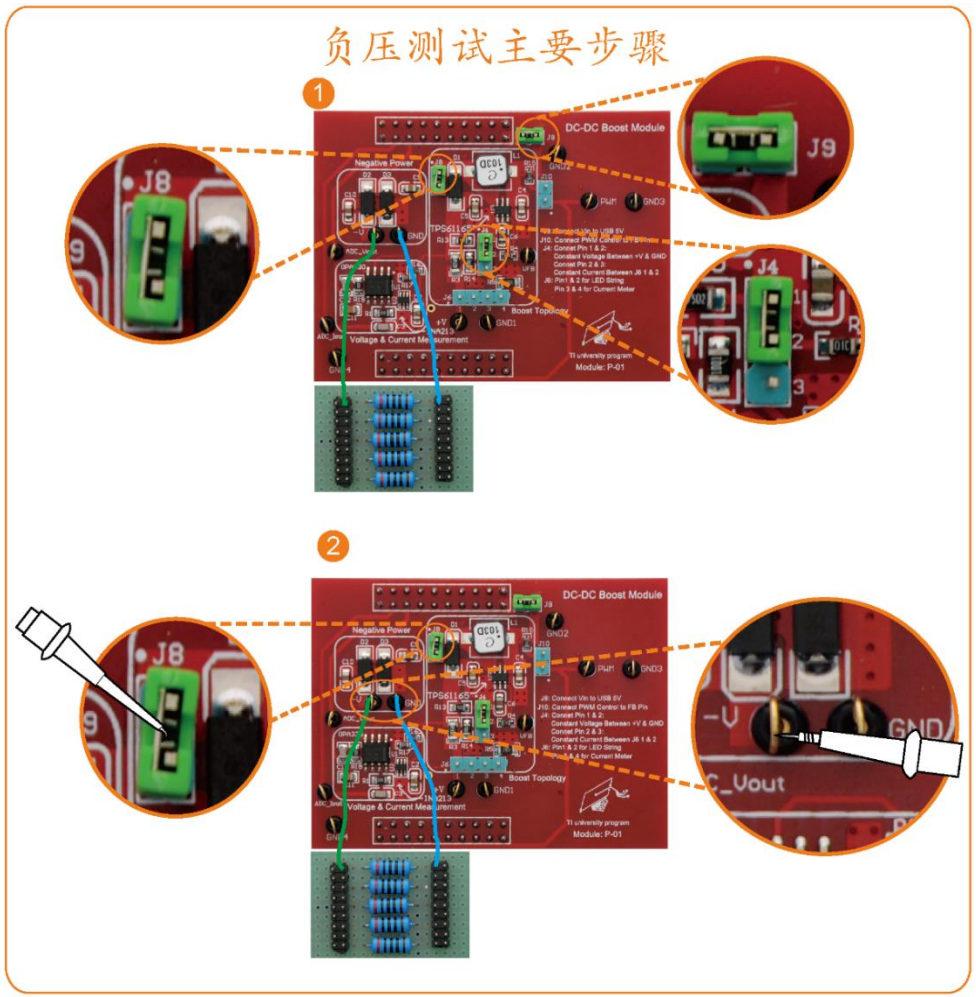


实物图

负压生成电路借用二倍压电路实现负压产生。通过MOSFET的导通和关断，在SW端不断产生高电平和低电平。当MOSFET关断时，在SW端产生高电平，电流经过C9、D3到地，对C9充电。当MOSFET导通时，电流从地，经过C10、D2、C9。此时，C9放电，C10充电。C10的上端为负，下端为正；而正端接地，所以，由C10的上端输出负压。

在以TPS61165为平台，通过外扩电路实现负压产生的条件是：利用该芯片搭建开关电源电路要形成闭环回馈电路。因此在恒流控制电路和恒压源输出电路时均能产生负压，因为恒压时电路接法简单点，下面按照恒压源输出电路的接法研究负压产生电路。





负压测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Boost(升压)模块，准备实验。

3、如图1所示，短接J9、J4的1,2及J8。完成负压测试的硬件配置。

4、连接负载，如图1所示，注意负载的功率和阻值大小。

5、如图2所示，进行仪表配置。测J8观测开关节点电压波形，测-V，GND1观测输出电压的纹波。

6、给TIVA上电，观察输出电压的电压值、纹波波形及开关节点波形。

7、变换负载的大小，受USB供电功率限制，负载的大小不得低于100Ω。记录不同负载下输出电压的电压值和纹波，比较负压输出和正压输出之间的区别，即有反馈和无反馈之间的差异。

注意

- 连接仪表及跳线时，断开电源。
- 试验箱不提供负载，需自己配备，本实验中使用100Ω、1W电阻，通过串并联实现不同负载变化。

第7章 DC-DC 降压模块

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B306

第7章　DC-DC 降压模块

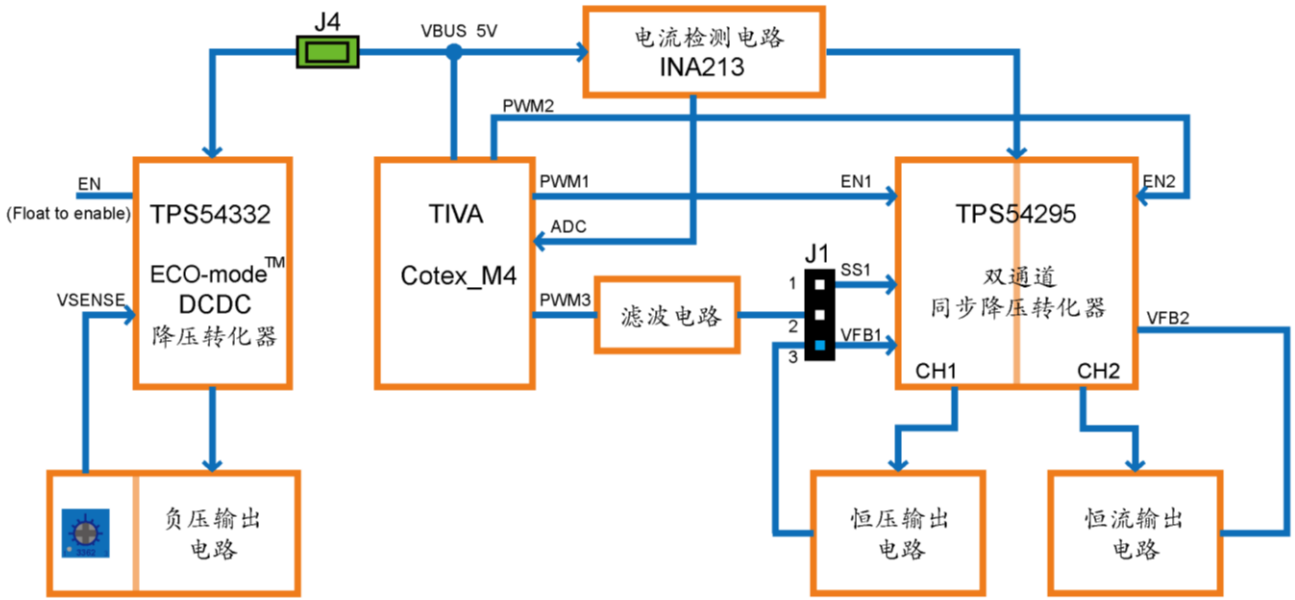
DC-DC Buck 实验模块介绍

实验目的:

1、理解并掌握buck; buck-boost电源拓扑。
2、理解并掌握输出电压的调节方式。
3、理解同步与非同步电路的差异。
4、LED恒流驱动的原理。
5、理解并掌握各类电源参数测量方法：效率，纹波，软启动时间，电感电压与电流的变化。
6、理解电源设计的PCB布线的基本原则。

实验简介:

本模块采用了一片双通道同步降压芯片TPS54295和一片单通道非同步降压芯片TPS54332实现buck拓扑的恒压、恒流和负压生成，并提供多种输出调节模式：1、PWM调节使能端；2、PWM滤波后调节反馈端；3、PWM滤波后调节软启动（SS）端；4、调节反馈电阻。其中，负压的产生采用了buck拓扑到 拓扑的转变，为了评估电源的效率，加入电流检测电路。



DC-DC Buck 模块布局及电路板跳线物理位置展示

■ PCB布局

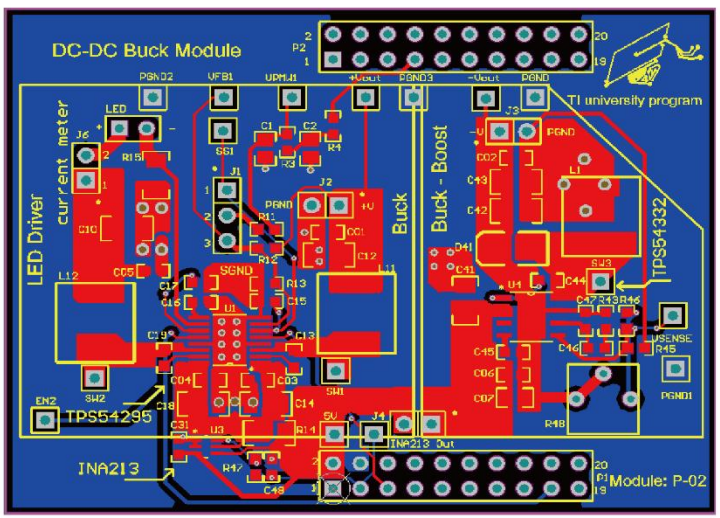


图6-1 PCB布局图

■ 实物图

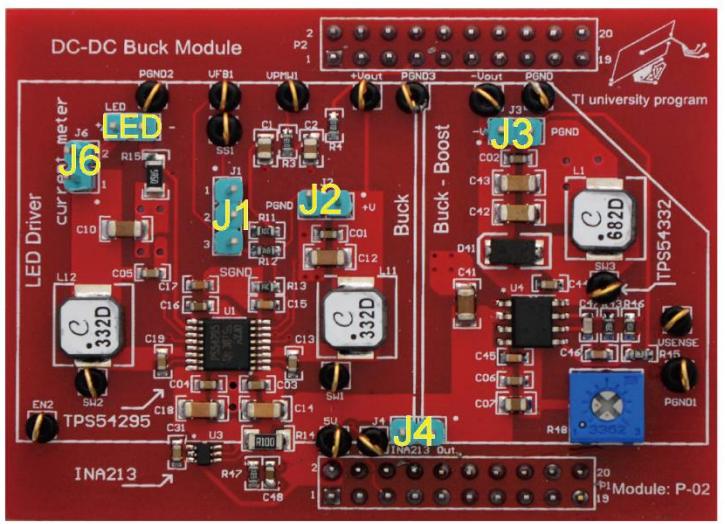
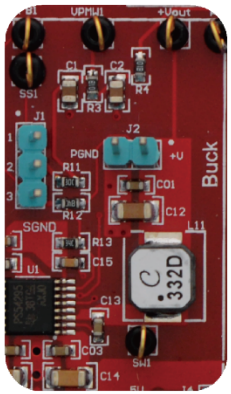


图6-2 实物图

PCB布局布线介绍

1、使用双路降压模块，一路恒流输出，一路恒压输出，理解恒压恒流电路的异同。
2、突出利用SS，FB和EN来调整输出电压/电流的方法，理解电源内部的反馈，软启动等特点。
3、使用Buck电源来实现Buck-Boost反向输出，帮助学生理解Buck和Buck-Boost的异同。
4、通过把同步和非同步电源放在一个板卡上，理解同步和非同步，并可测试关键结点的波形。

恒压源输出

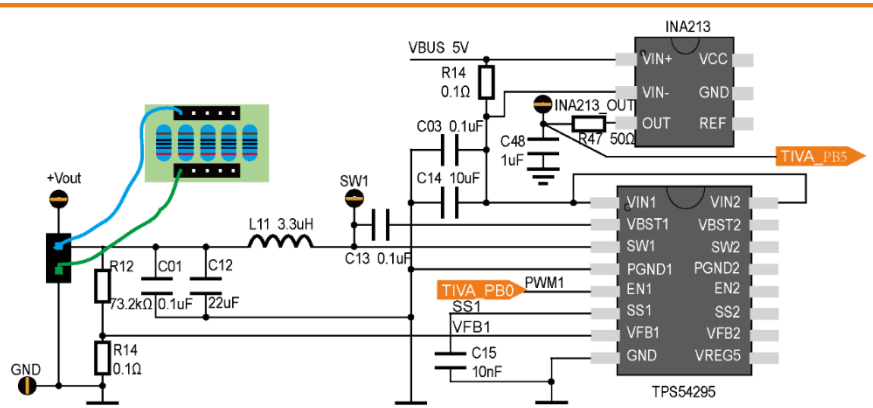


实物图

恒压源输出是开关电源能输出稳定的电压，且电压不会随着负载的变化而变化。该电路实现+5V输入，+3.3V输出的功能。如下为原理图，左侧为实物图。需要注意的是在恒流控制点亮LED时，负载是1W的LED灯珠。但是，在恒压源输出时负载不能仅仅是LED灯珠，这是因为二极管LED会使输出的电压发生钳位，导致电源芯片偏离了正常工作的轨道。因此，用功率电阻作为负载进行试验，S测试恒压源的输出和调节恒压输出的大小。

恒压输出的实现是：在反馈端FB和地之间接入电阻R14，则R14两端的电压被控制在一个设定值。由于FB端是运放的反相输入端，故其输入/输出电流几乎为零，这样，流过R14的电流就可以计算出来。在R14与L11之间接入电阻R12，把电压抬高。因为FB端的电压恒定，流过R12和R14的电流就恒定，所以，在R12和R14两端产生电压也恒定，从而实现了恒压源的输出。

电流检测电路：此电路中采用高测电流测量。INA213正好是共模高，失调电压小，温度漂移特性好的运算放大器。



恒压测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Buck(降压)模块，准备实验。

3、连接负载，如右图所示。注意负载功率和阻值大小。

4、在降压模块上完成恒压测试的仪表连接，如右图所示。其中测 1 观测反馈电压值；测 2 观测输出电压，通过调节示波器可以测量得到电压值及纹波情况；测 3 观测开关节点的波形；测 4 观测输入电流检测电路输出的电压值。

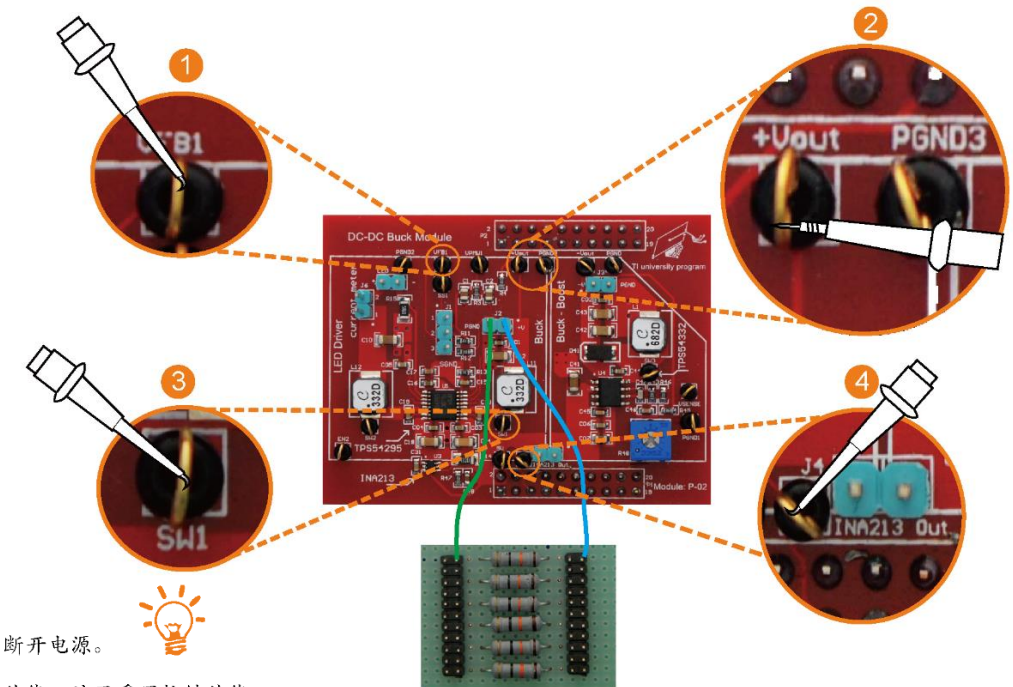
5、给TIVA上电，按LCD模块中按键S1(按键S1是通道1是否打开的切换键)，将TPS54295的通道1使能，进入工作状态，在LCD上可以看到输入电流的大小。

6、断电后，改变负载，通过示波器观察电路恒压的情况，及纹波的变化。

注意

- 连接仪表及跳线时，断开电源。
- 电路进入工作前需要使能，这里采用按键使能。请勿忘记。
- 实验套件中没有提供电阻负载，实验中的负载需要自己制作。这里的负载是阻值13Ω、功率1W的电阻。通过电阻的串并联改变负载大小。

恒压测试主要步骤



TPS54295电压调节方式介绍

DC-DC Buck模块提供了多种电压调节的方式。这里主要以TPS54295为中心，介绍三种电压调节方式，分别是使能端调节、软启动端调节及反馈端调节。

如右下图所示，为TPS54295部分内部简单原理图。图中主要指出该芯片内部5个重要的逻辑结构，即使能逻辑结构、软启动逻辑结构、比较器、控制结构及MOSFET。

使能端调节：

使能逻辑和反馈端（VFB）是一个线与逻辑，当EN输入为高电平，则整个芯片在电路连接完整的情况下是可以正常工作的，相反当EN输入为低电平，则反馈端电压被拉低，整个芯片不工作。

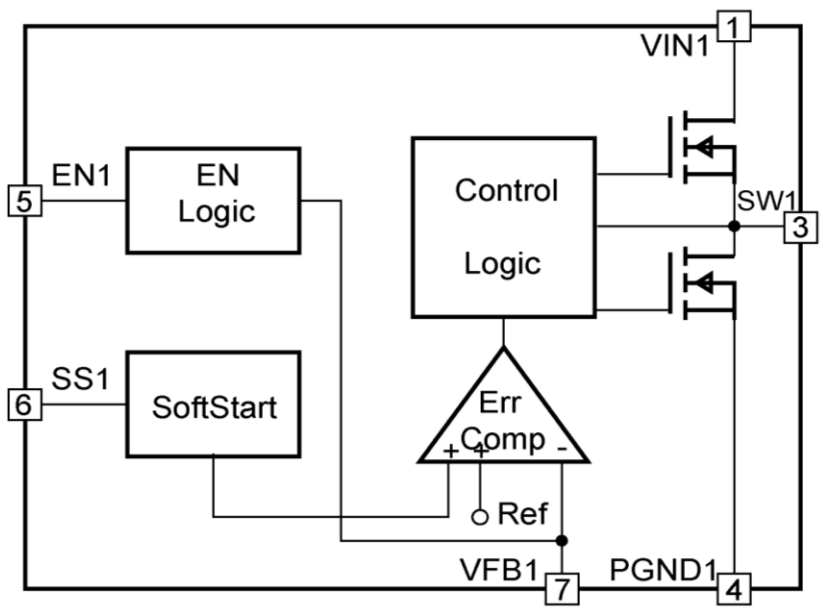
软启动调节：

电压由零慢慢提升到额定电压，使芯片能平滑地开启工作状态，不产生大的电流冲击。这就是软启动。

在TPS54295中，当芯片处于正常工作状态时，软启动处的电容呈现电压充满状态，这时对反馈端不产生影响。但当软启动处的电容充电不完全时，且电容的电压低于反馈电压，软启动的电压就会影响反馈端电压，使输出调整。

反馈端调节：

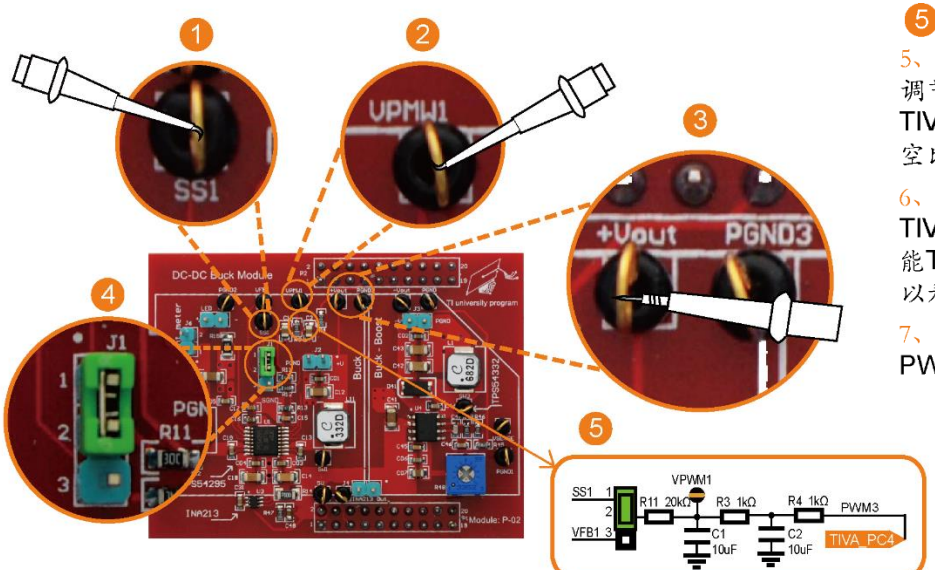
反馈端电压的调节是比较直接的。芯片会通过比较器输出的正负和其自身的逻辑控制来有效地调节开关的占空比，使输出达到稳定状态。



TPS54295内部简单原理图

电压调节测试主要步骤

——SS控制电压输出



SS控制电压输出测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Buck(降压)模块，准备实验。

3、在降压模块上完成电压控制测试，如左图所示为SS控制实现电压调节。

4、SS控制电压调节的测试点连接，测 1 观测SS1的电压值；测 2 观测PWM3滤波后的电压值，测 3 观测输出电压的电压值和纹波情况；5 是滤波电路。

5、给TIVA上电，观察VPWM1的值通过滚轮调节使VPWM1的最小（或者将表笔连接在TIVA的PWM3输出端口上，调节滚轮，使占空比最小。）。

6、关闭TIVA,如 4 所示短接J1的1,2。再次给TIVA上电，按一下LCD模块上的按键S1，使能TPS54295的EN1端口，缓慢调节滚轮，可以看到输出电压随滚轮转动发生变化。

7、记录PWM3的占空比和输出电压值，了解PWM3和输出电压之间的控制关系。

注意

- 连接仪表及跳线时，断开电源。
- 输出电压测量过程中，测纹波需要将示波器切换到交流模式，而测电压值则为直流模式。
- 如果想了解电源的性能，可以接负载进行测试。负载可以选择阻值13Ω功率1W电阻，通过串并联实现负载变化。

VFB电压调节测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Buck(降压)模块，准备实验。

3、在降压模块上完成电压控制测试，如右图所示为VFB控制实现电压调节。

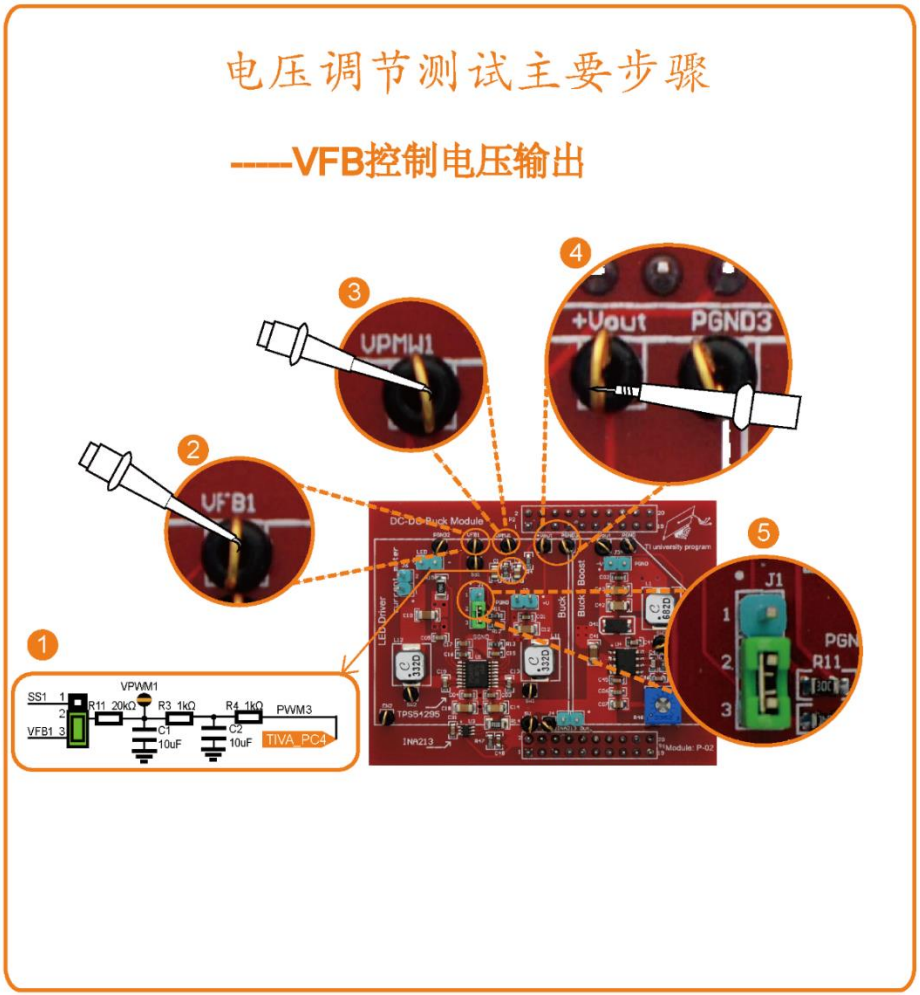
4、如右图所示，完成跳线连接，即5中显示，短接J1的2,3。

5、在降压模块上，完成仪表连接，如图所示，测 2 观测反馈电压值，测 3 观测PWM3滤波后的电压值，测4观测输出电压情况。 1是滤波电路的原理图。

6、给TIVA上电，按LCD模块上的按键S1，使能TPS54295的EN1端口，拨动滚轮，记录TIVA输出的PWM3的占空比和输出电压值。了解PWM3可电压之间的控制关系。

注意

- 连接仪表及跳线时，断开电源。
- 输出电压测量过程中，测纹波需要将示波器切换到交流模式，而测电压值则为直流模式。
- 如果想了解电源的性能，可以接负载进行测试。负载可以选择阻值13Ω功率1W的电阻，通过串并联实现负载变化。



TPS54295 提供两路输出：一路正压输出，一路高亮 LED 驱动。两个功能由 Tiva M4 分别控制。正压输出：通过 PC4 产生一路 PWM 波，PWM 波输入至 J1 跳线帽处，可跳过掉线将 PWM 波连接至 TPS54295 的 SS1 端或者 VFB1 端；通过 PB0 控制 TPS54295 的 EN1 端，PB0 配置成 GPIO 功能，其输出电平控制 EN1 端的使能或者禁能；PB0 输出电平状态由 LCD 开发板上的按键 S2 控制。高亮 LED 驱动：通过 PA6 产生一路 PWM 波控制 TPS54295 的 EN2 端，用于驱动 LED。这两路 PWM 波的占空比可以通过 LCD 开发板上的滚轮控制，同时 LCD 会显示当前测量点的电流值。

软件流程图及关键代码分析

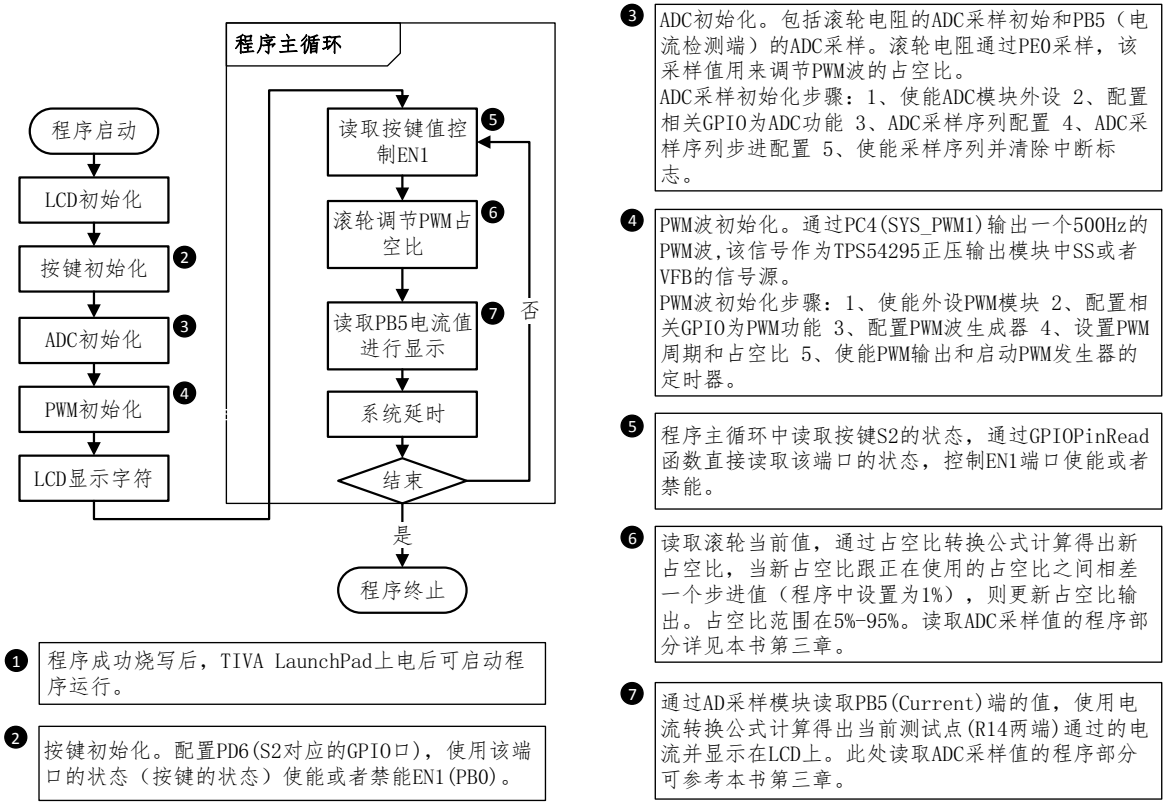


图 xx 程序流程图

此次实验程序涉及 LCD 显示，ADC 采样，按键控制以及 PWM 波功能。LCD 显示和 ADC 采样可参考本书第三章。

PWM 波功能实现

PWM 波实现中，其输出信息的周期是需要计算的，该计算公式在第六章中已经给出，本章将直接使用该公式，不在详细说明。其计算公式如下：

$$period = \frac{SysPWM\_Fre}{OutputPWM\_Fre}$$

程序中的时钟节拍值 PERIOD_TIME=25000，其表示的是 500Hz（系统主频 12.5Mhz，PWM 时钟频率配置为 SYSCTL_PWMDIV_1，也就是 1 分频）,可有上述公式直接计算得出。

$$period = \frac{SysPWM\_Fre}{OutputPWM\_Fre} = \frac{12.5\times10^6/1}{500} = 25000$$

程序代码如下

```
/*************************************************************************
 *  @berif  初始化PWM
 *  @param  none
 *  @return none
 *      ________
 *              |
 *      M4   PC4|-->M0PWM6(SYS_PWM1)
 *      ________| 25000       //500Hz
void Init_PWM()
{
    //设置PWM时钟，1分频
```

```
    SysCtlPWMClockSet(SYSCTL_PWMDIV_1);
    //使能外设PWM0模块
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
    //使能PWM0使用的外设GPIOC
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    //配置PC4复用功能是PWM
    GPIOPinConfigure(GPIO_PC4_M0PWM6);
    //配置PC4为PWM0模块使用
    GPIOPinTypePWM(GPIO_PORTC_BASE, GPIO_PIN_4);
    //配置PWM1模块，PWM1生成器1，向下计数并且立即更新参数
    PWMGenConfigure(PWM1_BASE, PWM_GEN_1, PWM_GEN_MODE_UP_DOWN | PWM_GEN_MODE_NO_SYNC);
    //设置PA6（M1PWM1）产生的PWM周期
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_1, PERIOD_TIME);
    //设置PC4（MOPWM6）的占空比25%
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_6, PERIOD_TIME / 4);
    //使能输出
    PWMOutputState(PWM0_BASE, PWM_OUT_6_BIT, true);
    //启动PWM0发生器3的定时器
    PWMGenEnable(PWM0_BASE, PWM_GEN_3);
    //使能发生器模块计数器同步
    PWMSyncTimeBase(PWM0_BASE, PWM_GEN_3);
}
```

PWM 输出信号的占空比调节部分程序代码如下

```
//根据滚轮采样值计算占空比
```

```
uint32_t cur_Duty = 5 + (90 * pui32ADCWheel_Value) / ADCMAX;
uint32_t period = cur_Duty * PERIOD_TIME / 100;

//改变PWM波的占空比
ROM_PWMGenPeriodSet(PWM0_BASE, PWM_GEN_3, PERIOD_TIME);
ROM_PWMPulseWidthSet(PWM0_BASE, PWM_OUT_6, period);
```

占空比计算公式在第六章已给出。

按键控制功能

按键使用 LCD 开发板上的 S2 完成，该按键连接至 PD6。按键初始化代码如下：

```
 * @brief   对端口C、D进行按键初始化
 * @param   none
 * @return  none
 *      ________
 *              |
 *  TIVA     PD6|<--Button2
 *      ________|
 ************************************************************/
void Init_Key()
{
    //初始化外设GPIO
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    // 设置PD为2MA，弱上拉输出
    ROM_GPIOPadConfigSet(GPIO_PORTD_BASE,  GPIO_PIN_6,
```

```
                    GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
    //设置GPIO输入模式
    ROM_GPIODirModeSet(GPIO_PORTD_BASE, GPIO_PIN_6, GPIO_DIR_MODE_IN);

}
```

对于按键状态读取采用键扫描完成，程序代码如下：

```
// 监控PortD端口的变化
uint32_t readData = GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_6);
```

按键控制的是 TPS54295 的 EN1 端，过在读取了按键状态后就可控制 EN1 端的状态。控制程序代码如下：

```
// 响应按键2(S2)
if(!(readData & GPIO_PIN_6))
{
    if(status1 == 0)
        status1 = 1;
    else
        status1 = 0;
    if(status1)
    {
        // PB0置高（使能EN_1）
        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0);
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0);
    }
    else
    {
```

```
        //PB0置低（禁能EN_1）
        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0);
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0);
    }
}
```

实验的程序设计中

实验的程序设计中需要在 LCD 上显示检测点的电流值。电流值是采用 ADC 采样获得，ADC 采样配置以及获取采样值跟本书第三章中的滚轮控制部分相似，故该部分可参考第三章完成。获取 ADC 采样值后需要根据实际电路计算得出实际值。程序通过 PB5 口采样 INA213 的输出端获取采样值，而 LCD 上显示的是 INA213 输入端流经的电流值，即流经 R_{14} 的电流值。INA213 具有 50 倍增益，计算公式如下：

$$Current = \frac{Cur\_ADC \times ADCRef}{ADCMax \times 50 \times R_{14}} = \frac{Cur\_ADC \times 3.3}{4096 \times 50 \times 0.1}$$

程序中将该值扩大 1000 倍使用，故该公式简化为：

$$Current = \frac{Cur\_ADC \times 3.3 \times 1000}{4096 \times 50 \times 0.1} = \frac{Cur\_ADC \times 660}{4096} mA$$、

程序代码如下：

```
unsigned long Current_Sample = (pui32ADC_CurrentValue * 660) / 4096;
```

pui32ADC_CurrentValue 是 PB5 端口的 ADC 采样值。该值获取可参考本书第三章滚轮值的获取部分。

恒流高亮 LED 驱动

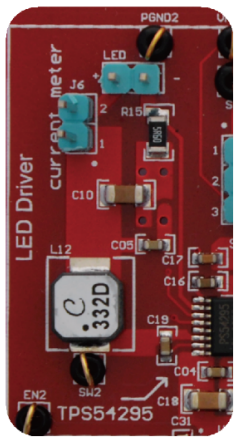
实验内容

以单个大功率（1W）LED 为负载，通过调节恒流输出的电流大小，观察 LED 的亮度的变化和恒流电流的大小

资源准备

等效原理图及电路分析

恒流高亮LED驱动

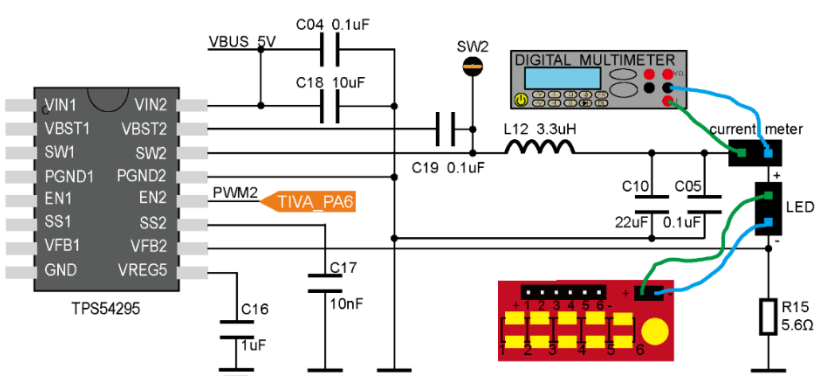


实物图

利用TPS54295第二通道搭建的电路是buck型开关电源电路，原理图如下图所示，实物图如左图所示。开关电源芯片TPS54295内部有MOSFET开关器件，它的开关频率达到700kHz。在开关导通时，电源给电感L12和电容C10充电；当开关关断时，相当于将L12的一端接地，L12释放能量。和buck型电路工作原理一样，通过电感L12和电容C10充放电，实现电压的转换。同时，该电路在FB端形成闭环通路，可以实现电压的稳定输出。

电流调节方式:

在本实验中通过PWM控制EN2起到平均电流的调节。由于使用LED做实验，一定的频闪对于肉眼来说是不可见的。故可以通过调节EN2来调整LED的亮度。



恒流高亮LED驱动测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Buck(降压)模块，准备实验。

3、在降压模块上完成恒流高亮LED驱动测试的仪表连接，如右图所示。其中 1 中一个是电流测量的接口，J6的1接红表笔，J6的2接黑表笔；另一个的大功率LED的接口，连接时注意正负极。测 2 观测TPS54295的EN2端口处波形。测 3 观测输入电流检测电路输出的电压值。

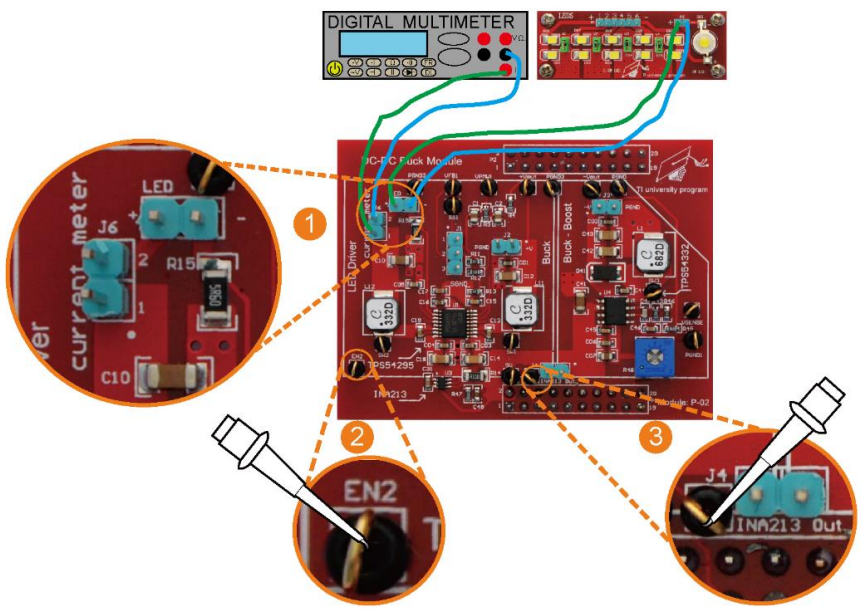
4、连接负载，如右图所示。此处使用的是试验箱中的LED板，连接在电路中的是1W的LED，连接过程中注意正负极。

5、给TIVA上电，可以观察到LED点亮，拨动滚轮，可以看到LED的亮度发生变化，此处是通过控制EN2端实现调节作用，过程中可以记录EN2处波形的占空比和电流表显示的电流值，了解他们之间的控制关系。

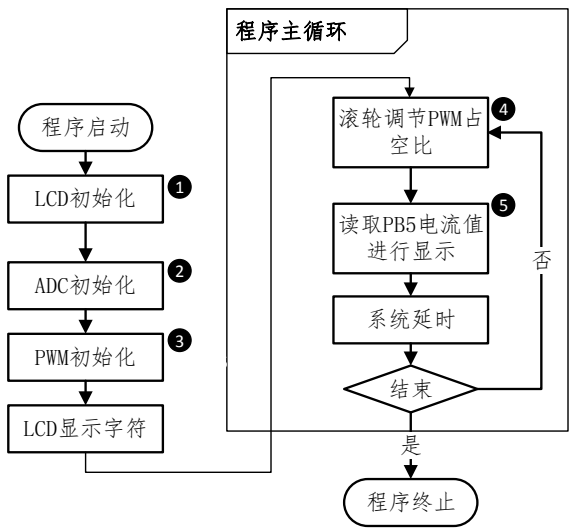
注意

- 连接仪表及跳线时，断开电源。
- 负载和电流表连接过程中注意正负极。

恒流高亮LED驱动测试主要步骤



软件流程图及关键代码分析



❶ LCD初始化包括LCD端口使能、SSI通信协议配置、LCD配置初始化、LCD清屏4个步骤。每个步骤详情请见本书第三章。

❷ ADC初始化。包括滚轮电阻的ADC采样初始和PB5（电流检测端）的ADC采样。滚轮电阻通过PE0采样，该采样值用来调节PWM波的占空比。PB5采样得到的是当前电源的电流。
ADC采样初始化步骤：1、使能ADC模块外设 2、配置相关GPIO为ADC功能 3、ADC采样序列配置 4、ADC采样序列步进配置 5、使能采样序列并清除中断标志。

❸ PWM波初始化。通过PA6(EN_2)输出一个频率为500Hz的PWM波，该PWM波输出至TPS54295中的EN2。其占空比可以通过滚轮进行调节。
PWM波初始化步骤：1、使能外设PWM模块 2、配置相关GPIO为PWM功能 3、配置PWM波生成器 4、设置PWM周期和占空比 5、使能PWM输出和启动PWM发生器的定时器。

❹ 读取滚轮当前值，通过占空比转换公式计算得出新占空比，当新占空比跟正在使用的占空比之间相差一个步进值（程序中设置为1%），则更新占空比输出。占空比范围在5%-95%。通过更改占空比可以调节负载LED的亮度。滚轮当前值的读取详见本书第三章。

❺ 通过AD采样模块读取PB5(Current)端的值，使用电流转换公式计算得出当前测试点(R14两端)通过的电流并显示在LCD上。AD采样值的读取可参考本书第三章。

图 xx 程序流程图

此次实验程序涉及 LCD 显示，ADC 采样，以及 PWM 波功能。LCD 显示和 ADC 采样可参考本书第三章。

PWM 功能实现

实验中通过 PA6 产生一路 PWM 信号直接控制 TPS54295 的 EN2 端，该实验产生的 PWM 信号跟本章实验 B 中的产生的 PWM 信号一致，唯一的不同是产生 PWM 信号的端口不同。

程序代码如下：

```
/*************************************************************************
 *  @berif  初始化PWM获取两组反向的脉宽调制信号
 *  @param  none
 *  @return none
 *      ________
 *              |
 *      M4   PA6|-->M1PWM2(EN_2)
 *      ________|
 * ************************************************************************/
#define PERIOD_TIME        25000      //500Hz
    //设置PWM时钟，1分频
    SysCtlPWMClockSet(SYSCTL_PWMDIV_1);
    //使能外设PWM1模块
```

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    //使能PWM1使用的外设GPIOA
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    //配置PA6复用功能为PWM
    GPIOPinConfigure(GPIO_PA6_M1PWM2);
    //配置PA6为PWM1模式使用
    GPIOPinTypePWM(GPIO_PORTA_BASE, GPIO_PIN_6);
    //配置PWM1模块，PWM1生成器1，向下计数并且立即更新参数
    PWMGenConfigure(PWM1_BASE, PWM_GEN_1,
                    PWM_GEN_MODE_UP_DOWN | PWM_GEN_MODE_NO_SYNC);
    //设置PA6（M1PWM1）产生的PWM周期
    PWMGenPeriodSet(PWM1_BASE, PWM_GEN_1, PERIOD_TIME);
    //设置PA6（M1PWM1）的占空比25%
    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_2, PERIOD_TIME / 4);
    //使能输出
    PWMOutputState(PWM1_BASE, PWM_OUT_2_BIT, true);
    //启动PWM1发生器1的定时器
    PWMGenEnable(PWM1_BASE, PWM_GEN_1);
    //使能发生器模块计数器同步
    PWMSyncTimeBase(PWM1_BASE, PWM_GEN_1);
}
```

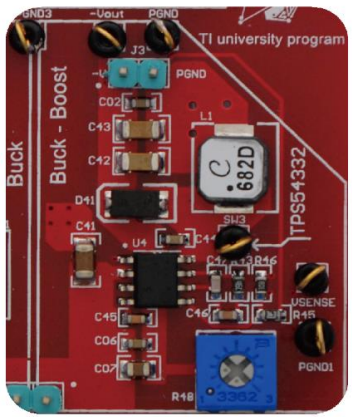
PWM 输出信号的占空比调节部分程序代码如下

```
//根据滚轮采样值计算占空比
uint32_t cur_Duty = 5 + (90 * pui32ADCWheel_Value) / ADCMAX;
uint32_t period = cur_Duty * PERIOD_TIME / 100;

//改变PWM波的占空比
ROM_PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, PERIOD_TIME);
ROM_PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2, period);
```

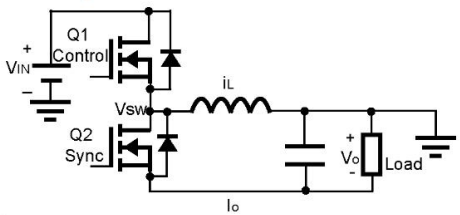
而早 LCD 上显示的内容跟使用 B 也是一致的，计算公式和实现代码可参见使用 B。

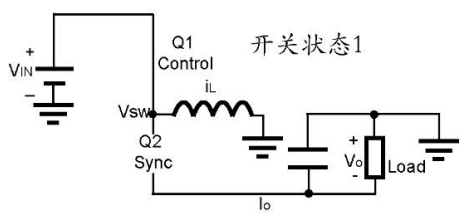
通过拓扑变换实现负压生成

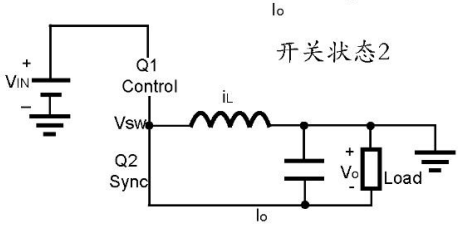


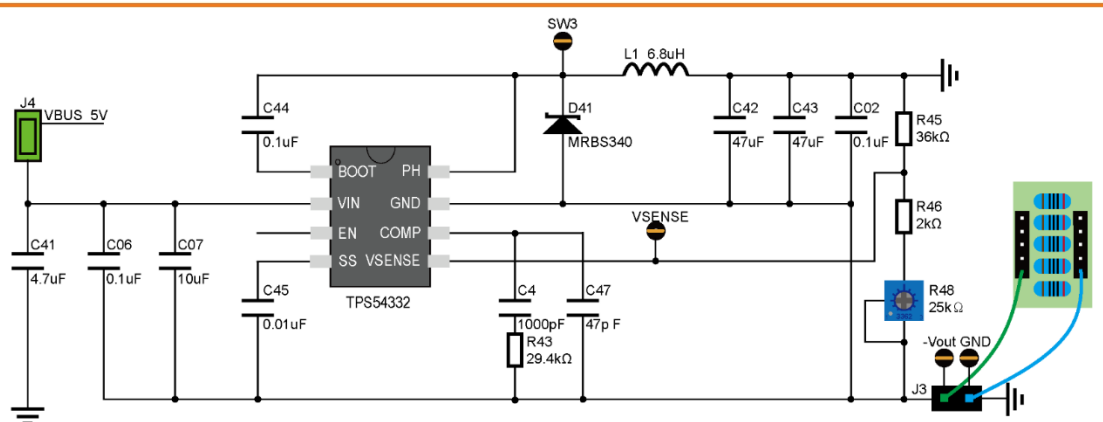
实物图

该实验的原理图如下所示，实物图如左侧所示。buck到buck-boost的转变主要就是电路中两个开关的切换。如图开关状态1和开关状态2所示。通过开关状态变化实现负压输出的范围可以大于输出电压也可以小于输入电压。









通过拓扑变换实现负压生成测试实验

1、理解原理图，编写Launchpad代码（参考代码见网上资源）。

2、在母板上连接TIVA、LCD模块、DC-DC Buck (降压)模块，准备实验。

3、在降压模块上完成通过拓扑变换实现负压生成测试的跳线及仪表连接，如右图所示。其中 1 是负压输出连接负载的接口； 2 是负压输出的测试点； 3 是开关电源的开关节点； 4 是反馈电压的测试点； 5 是地的测试点； 6 是可调电阻，如果想改变输出电压值可以可以改变滑变阻值，起到电压调节作用； 7 是电路中唯一一个跳线连接，即短接J4完成电路连接。

4、连接负载，如右图所示。注意负载的功率和阻值大小。

5、给TIVA上电，先将电压调节到-15V输出，然后

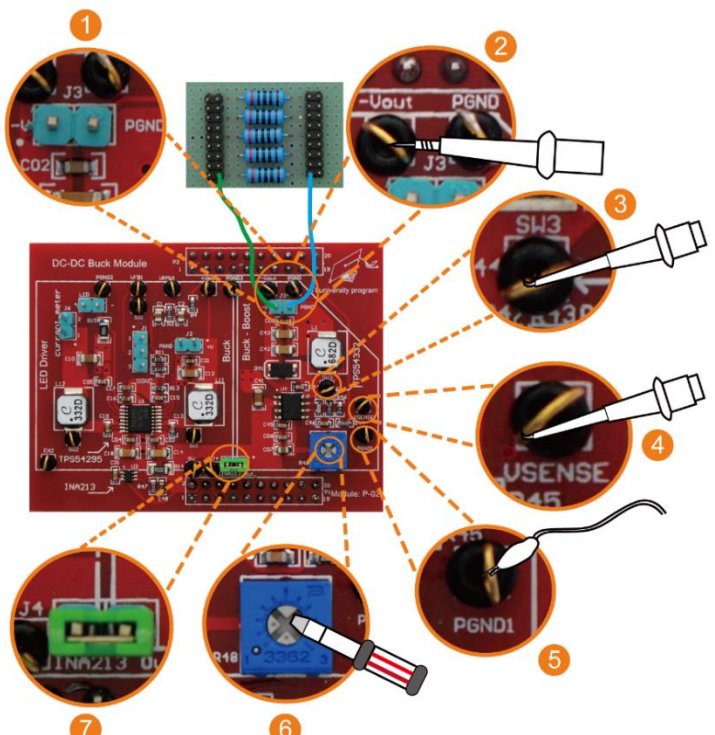
6

注意

- 连接仪表及跳线时，断开电源。
- 实验套件中没有提供电阻负载，实验中的负载需要自己制作。这里的负载是阻值100Ω、功率1W的电阻。通过电阻的串并联改变负载大小。

- 实验套件中没有提供电阻负载，实验中的负载需要自己制作。这里的负载是阻值100Ω、功率1W的电阻。通过电阻的串并联改变负载大小。

通过拓扑变换实现负压生成测试主要步骤



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

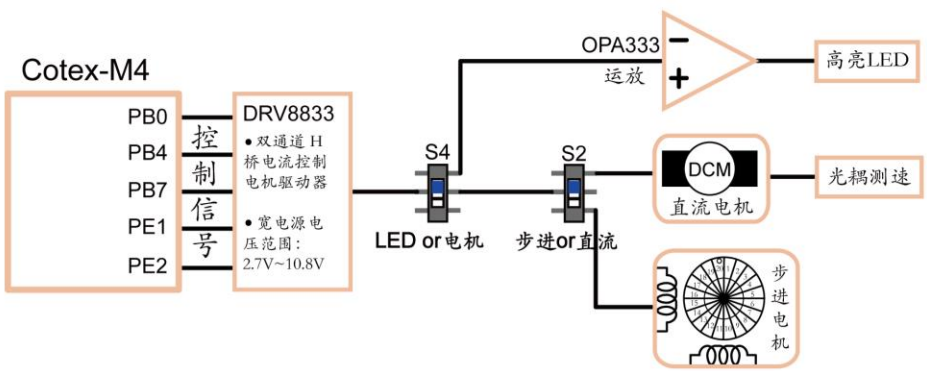
公司地址：浙江省杭州市西湖区留和路16号新峰商务楼B306

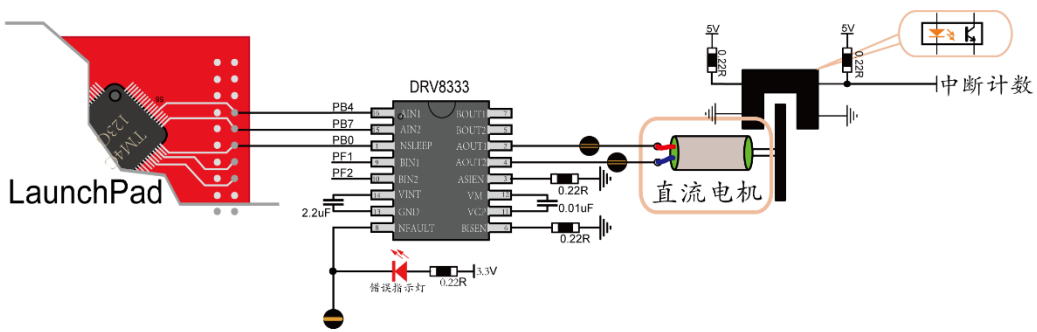
第8章　电机驱动模块

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

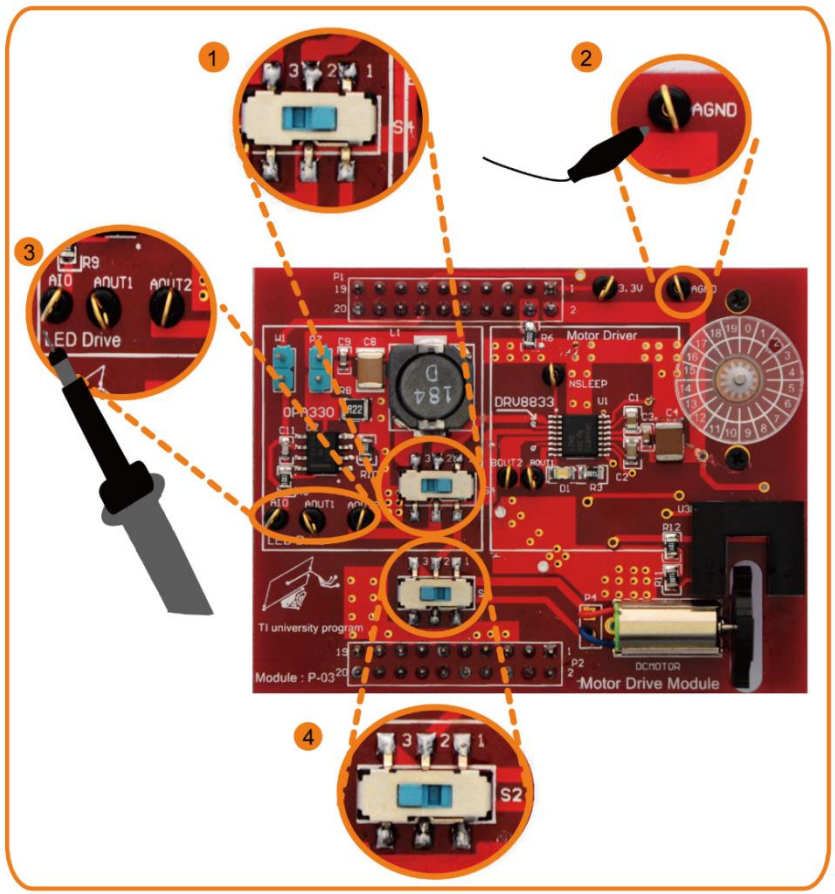
直流电机的控制与测速

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。

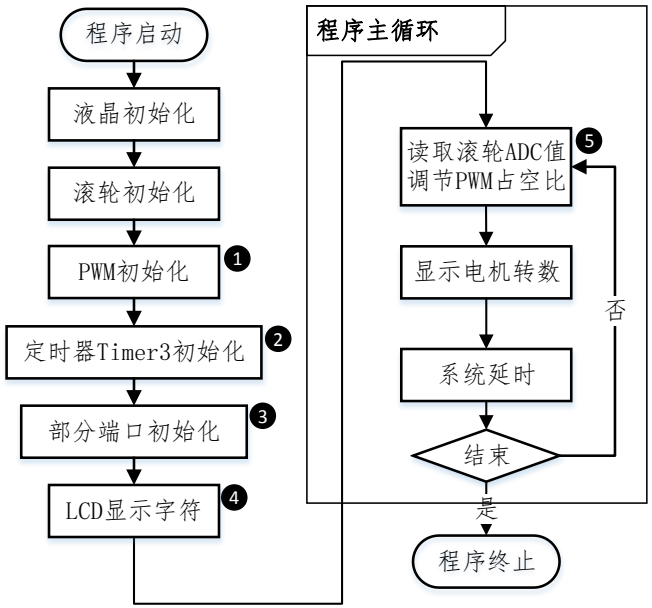
3、在电机驱动模块上完成开关的选择，如图 1 所示开关S4拨向右即连接了S4的1、2；如图 4 所示开关S2拨向左即连接了S2的3、2。

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。

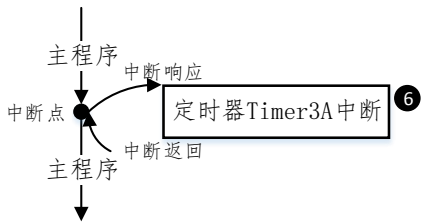


注意

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

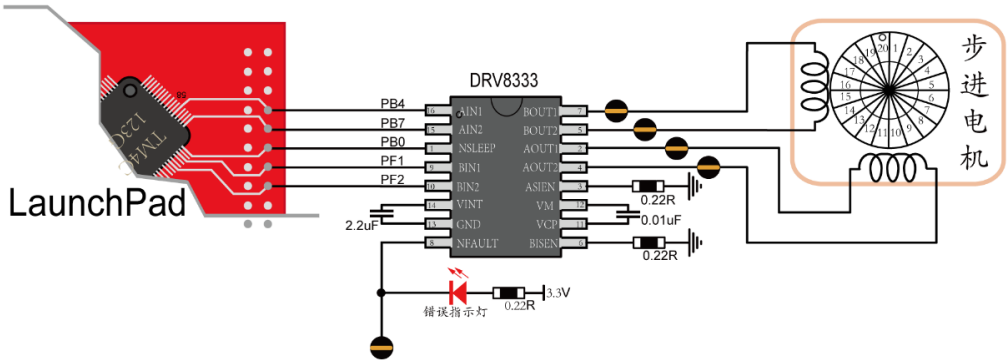
A. 步进电机的控制

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。

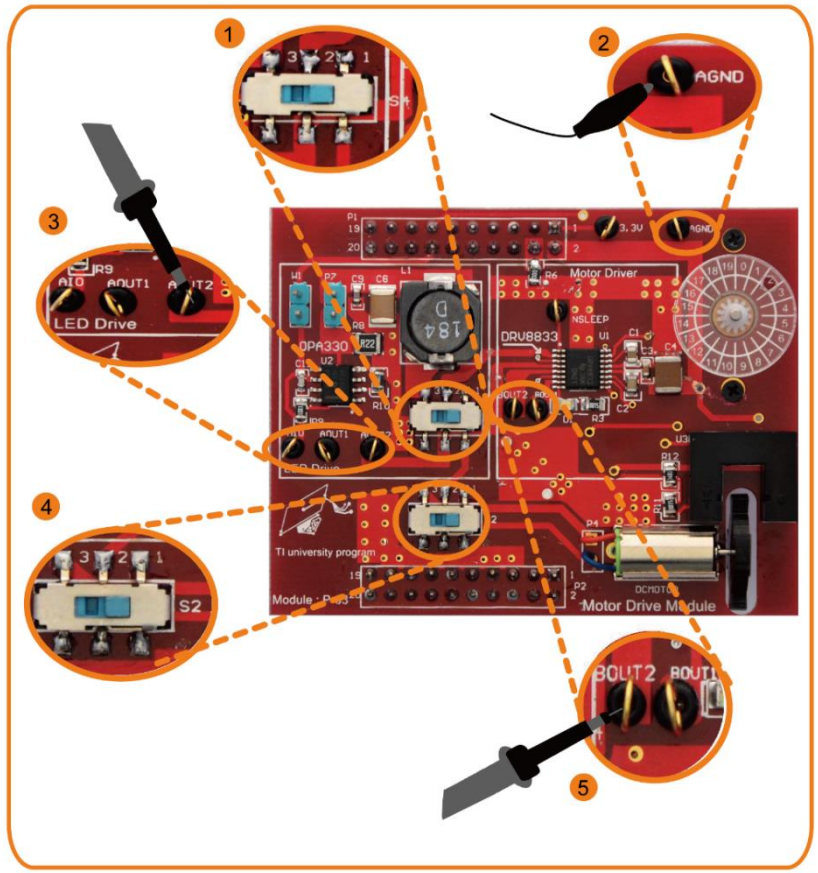
2、在母板上TIVA、液晶、电机驱动模块连接完成，准备实验。

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

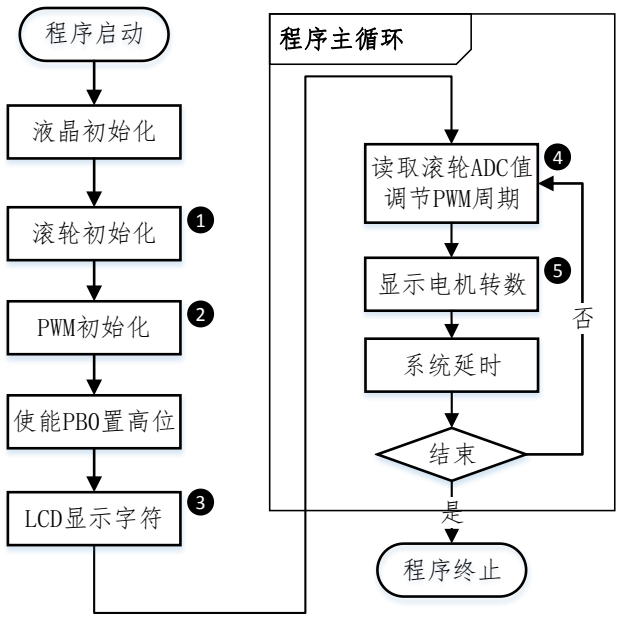
连接仪表及跳线时断开电源。



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;
    }
```

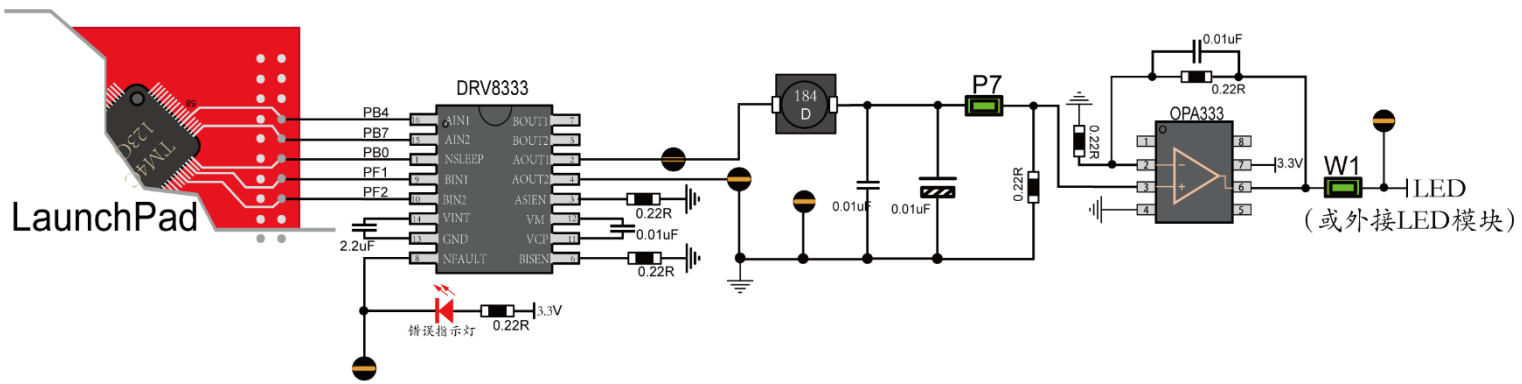
B. 高亮 LED 的驱动与电流检测

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。

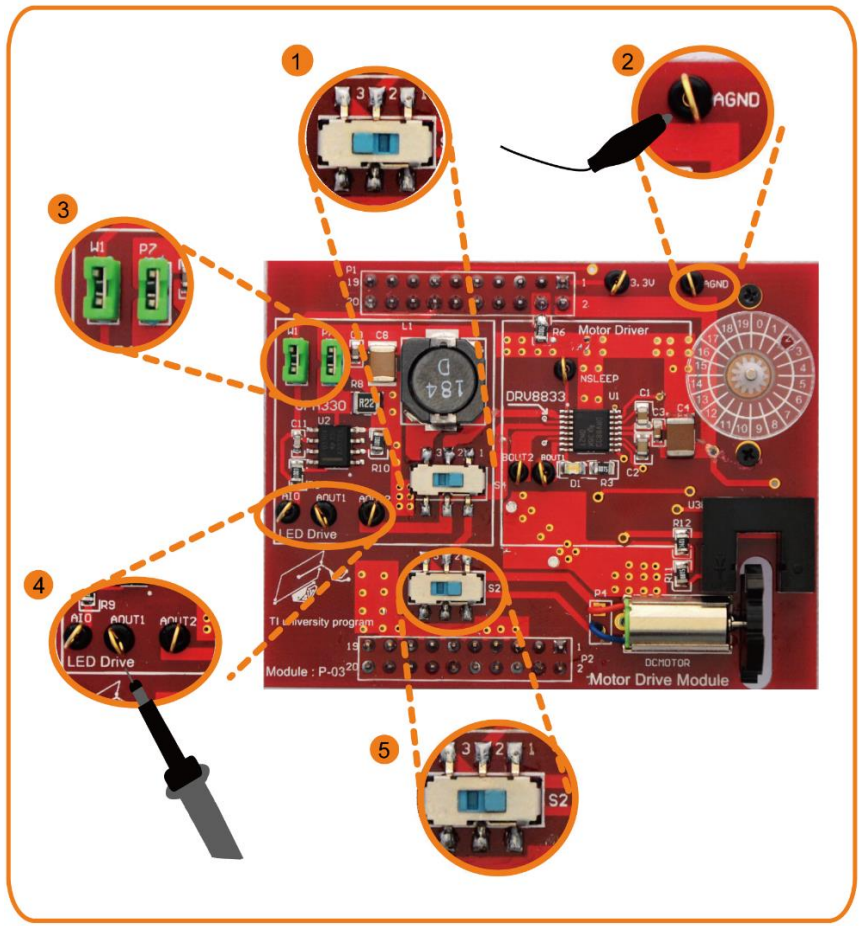
2、在母板上TIVA、液晶、电机驱动模块连接完成，准备实验。

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。