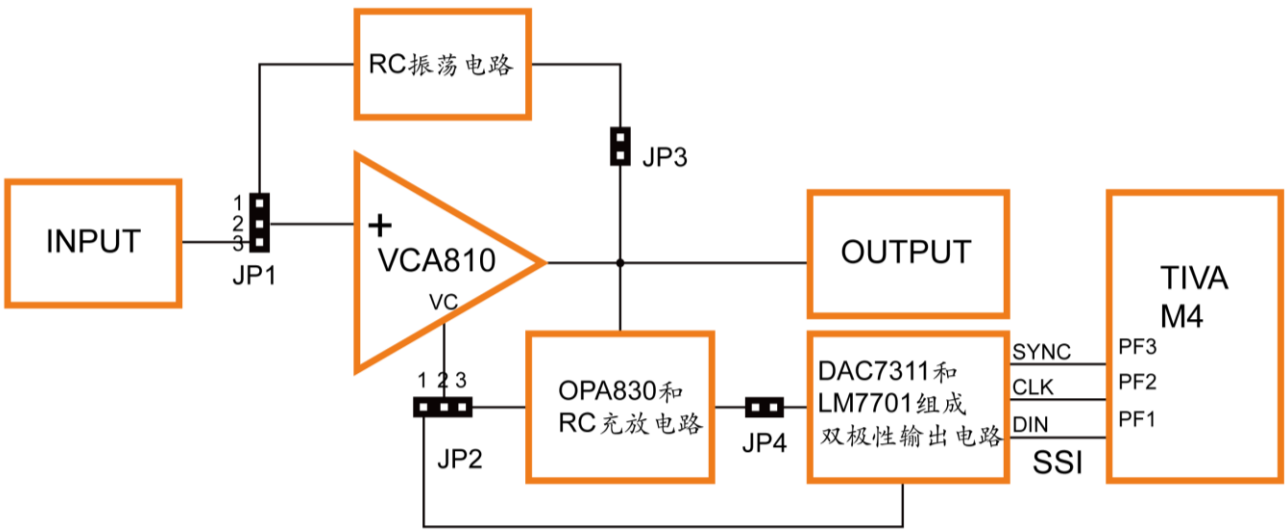
第九章、 宽带压控增益模块

高速压控增益模块介绍

实验简介

本本模块采用一片高增益可调节范围宽带压控放大器VCA810组成高速压控增益电路。其中，VCA810的增益控制是通过调节控制输入端VC的电压来实现的。不同的控制电压，就可以得到不同的增益值，从而获得不同增益的输出值。该控制电压可以直接从DAC双极性输出电路中得到，并可通过液晶模块的滚轮调节大小。另外通过跳冒的选择，可以提供多种实验电路：1、宽带压控增益放大与衰减；2、正反馈RC振荡器；3、自稳幅闭环振荡器。



实验程序使用按键 S3 选择当前程序工作在实验 B，实验 C 还是实验 D，这三个实验程序上唯一的区别是：在程序刚运行时三个实验的 DAC7311 的初始值不同，其实部分都是一致的；使用按键 S1 和 S2 完成 VC 端电压值的调节，同时改变 DAC7311 的工作值，在液晶上同步显示当前 VC 端电压值；通过 SSI 传输协议改变 DAC7311 的工作值；

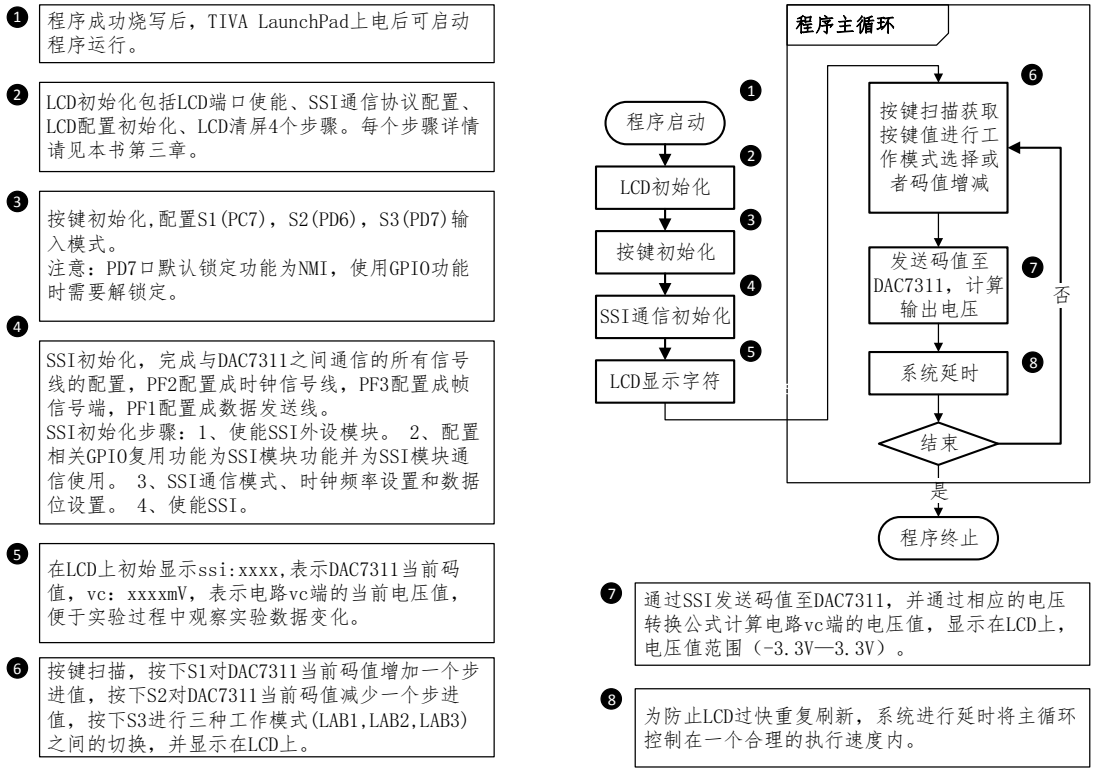


图 xx 程序流程图

LCD 显示部分程序可参考本书第三章。

按键功能

实验程序需要使用 LCD 开发板上的所有按键：S1、S2、S3。程序使用按键扫描完成端口状态的读取，在按键的初始化配置中需要注意的是 S3 的配置，因为 S3 连接在端口 PD7，而 PD7 口已经被锁定为 NMI（non-maskable interrupt，不可屏蔽中断）功能，所以在使用该端口时需要先解除锁定，使其能够配置成 GPIO 功能。解除锁定代码如下：

```
//解锁
HWREG(GPIO_PORTD_BASE+GPIO_O_LOCK)  |= GPIO_LOCK_KEY;
HWREG(GPIO_PORTD_BASE+GPIO_O_CR)    |= (1<<7);
HWREG(GPIO_PORTD_BASE+GPIO_O_DEN)   &=(~(1<<7));
HWREG(GPIO_PORTD_BASE+GPIO_O_PDR)   &= (~(1<<7));
HWREG(GPIO_PORTD_BASE+GPIO_O_PUR)   &= (~(1<<7));
```

```
HWREG(GPIO_PORTD_BASE+GPIO_O_AFSEL) &=(~(1<<7));
```

完成解锁后 PD7 口就可以跟 PC7，PD6 一样配置初始化使用。

按键使用端口的初始化程序代码如下：

```
 * @brief   对端口C、D进行按键初始化
 * @param   none
 * @return  none
 *      ________
 *             |
 *          PC7|<--Button1
 *  TIVA     PD6|<--Button2
 *          PD7|<--Button3
 *      ________|
 *  注：PD7口默认锁定功能为NMI，使用其GPIO功能时需要解锁定再配置成GPIO功能
 ************************************************************/
void Init_Key()
{

//-------------------------------------------------------------------
    //解锁
    HWREG(GPIO_PORTD_BASE+GPIO_O_LOCK)  |= GPIO_LOCK_KEY;
    HWREG(GPIO_PORTD_BASE+GPIO_O_CR)   |= (1<<7);
    HWREG(GPIO_PORTD_BASE+GPIO_O_DEN)  &=(~(1<<7));
    HWREG(GPIO_PORTD_BASE+GPIO_O_PDR)  &= (~(1<<7));
    HWREG(GPIO_PORTD_BASE+GPIO_O_PUR)  &= (~(1<<7));
    HWREG(GPIO_PORTD_BASE+GPIO_O_AFSEL) &=(~(1<<7));

//-------------------------------------------------------------------
    //初始化外设GPIO
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    // 设置PD为2MA，若上拉输出
    ROM_GPIOPadConfigSet(GPIO_PORTC_BASE,  GPIO_PIN_7,
                         GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
    ROM_GPIOPadConfigSet(GPIO_PORTD_BASE,  GPIO_PIN_6,
                         GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
    ROM_GPIOPadConfigSet(GPIO_PORTD_BASE,  GPIO_PIN_7,
```

```
                        GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);
    //设置GPIO输入模式
    ROM_GPIODirModeSet(GPIO_PORTC_BASE, GPIO_PIN_7,
                        GPIO_DIR_MODE_IN);
    ROM_GPIODirModeSet(GPIO_PORTD_BASE, GPIO_PIN_6,
                        GPIO_DIR_MODE_IN);
    ROM_GPIODirModeSet(GPIO_PORTD_BASE, GPIO_PIN_7,
                        GPIO_DIR_MODE_IN);
}
```

按键扫描程序直接使用 ROM_GPIOPinRead()进行读取按键所在端口的状态值，三个按键对应不同的功能：S3 选择实验项目，S1 增大 DAC7311 工作值使 VC 端电压增大，S2 减小 DAC7311 工作值使 VC 端电压减小。程序扫描三个端口即 PC7：S1 按键端口；PD6：S2 按键端口；PD7：S3 按键端口。

按键扫描程序代码如下：

```
/*************************************************************************
**
 * @brief   按键扫描函数
 * @param   none
 * @return  0x00            没有键按下
 *          0x01            按下PC7，S1
 *          0x02            按下PD6，S2
 *          0x03            按下PD7，S3
 *      ________
 *              |
 *          PC7|<--Button1
 *   TIVA   PD6|<--Button2
 *          PD7|<--Button3
 *      ________|
 *
 *
*************************************************************************
*/
unsigned char scan_key(void)
{
    if (ROM_GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_7) == 0x00)
    {
```

```
        // 延时约10ms，消除按键抖动
        ROM_SysCtlDelay(10*(ROM_SysCtlClockGet() / 3000)); KEY抬起
        while (ROM_GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_7) == 0x00);
        // 延时约10ms，消除松键抖动
        ROM_SysCtlDelay(10*(ROM_SysCtlClockGet() / 3000));
        return 0x01;
    }ROM_GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_6) == 0x00)
    {约10ms，消除按键抖动
        ROM_SysCtlDelay(10*(ROM_SysCtlClockGet() / 3000));
        // 等待KEY抬起
        while (ROM_GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_6) == 0x00);
        // 延时约10ms，消除松键抖动
        ROM_SysCtlDelay(10*(ROM_SysCtlClockGet() / 3000));
        return 0x02;
    }
    if (ROM_GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_7) == 0x00)
    {
        // 延时约10ms，消除按键抖动
        ROM_SysCtlDelay(10*(ROM_SysCtlClockGet() / 3000));
        // 等待KEY抬起
        while (ROM_GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_7) == 0x00);
        // 延时约10ms，消除松键抖动
        ROM_SysCtlDelay(10*(ROM_SysCtlClockGet() / 3000));
        return 0x03;
    }
   return 0;
}
```

按键扫描函数的返回值表示实验中按下了不同按键，根据该值就可以完成工作模式转换或者是 VC 端电压的增减。按键响应程序代码如下：

```
key_val = scan_key();          //键扫描
if(key_val)
{
    switch(key_val)
    {
        //按下S1（按键1），增加
        case 0x01:
```

```
            //......
            break;
        //按下S2（按键2），减小
        case 0x02:
            //......
            break;
        //按下S3（按键3），工作模式的切换（实验之间的切换）
        //切换工作模式主要改变的是dac7311工作的默认码值（VC端）
        //开始工作的电压。
        case 0x03:
            switch(Key3_PressCount)
            {
                //工作模式1：带宽压控增益放大与衰减
                case 1:
                    //......
                    break;
                //工作模式2：正反馈RC震荡器
                case 2:
                    //......
                    break;
                //工作模式3：自稳幅闭环振荡器
                case 3:
                    //......
                    break;
                default:
                    break;
                }
                Key3_PressCount++;
                if(Key3_PressCount > 3)
                {
                    Key3_PressCount = 1;
                }
            default: break;
        }
    }
}
```

SSI 通信功能

DAC7311 与 Tiva M4 之间通过 SSI（SPI）通信完成。DAC7311 是 12-bit 的 DAC，其内部有一个 16-bit 的寄存器。寄存器格式如下：

Bit	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
Data	PD1	PD2	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X

寄存器低两位为无效位，B2~B13 为数据位，高两位 B14（PD2），B15（PD1）为控制选择位表示 DAC7311 不同的工作模式：一种正常工作模式，三种 power-down 工作模式。在实验程序只使用正常工作模式，这两位都配置为 0。

DAC7311 由 SYNC，SCLK，DIN 三线控制。SYNC 为信号选择线，相当于 Tiva M4 的 SSIFss 信号线。SCLK 为时钟信号线，相当于 Tiva M4 的 SSIClk 信号线。DIN 为数据线，相当于 Tiva M4 的 SSITx 信号线。SSI（SPI）通信程序设置如下：

```
/*************************************************************************
**
 *  @brief  SSI初始化设置
 *  @param  none
 *  @return none
 *
 *      ________________
 *                      |
 *          PF2(SSI1Clk)|-->SPICLK  时钟信号端
 *  TIVA    PF3(SSI1Fss)|-->SYNC    帧信号端
 *          PF1(SSI1Tx) |-->SDIN    SSI数据发送端（LM4F120->DAC7811）
 *      ________________|
 *
*************************************************************************
*/
void ssi_en()
{
    //使能SSI1外设
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI1);
    //使能SSI1使用的GPIOF外设
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //配置PF2复用功能为SSI1CLK，时钟线
    ROM_GPIOPinConfigure(GPIO_PF2_SSI1CLK);
    //配置PF3复用功能为SSI1FSS，片选线
```

```
    ROM_GPIOPinConfigure(GPIO_PF3_SSI1FSS);
    //配置PF1复用功能为SSI1TX，数据发送线
    ROM_GPIOPinConfigure(GPIO_PF1_SSI1TX);

    //配置PF1，PF2，PF3供SSI1使用
    ROM_GPIOPinTypeSSI(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 |
                       GPIO_PIN_3);
    //端口模式:1M,16位数据
    ROM_SSIConfigSetExpClk(SSI1_BASE, ROM_SysCtlClockGet(),
                    SSI_FRF_TI, SSI_MODE_MASTER, 1000000, 16);
    ROM_SSIEnable(SSI1_BASE);
}
```

DAC7311 寄存器的低两位无效，所以在 SSI 通信程序实现中需要将发送的数据左移两位，然后再通过 SSI 进行传输。程序代码实现如下：

```
/*************************************************************************
********
 *  @brief  发送数据到dac7311
 *  @param  val    取值范围0~4095
 *  @return 0   参数不正确;
 *          1   传输成功;
 ************************************************************/
unsigned char ssi_send_2_dac7311(unsigned long val)
{
    if(val > 4095) return 0;          //判断参数正确与否
    val = val << 2;                   //左移两位，DAC7311内部寄存器低两位无
效
    ROM_SSIDataPut(SSI1_BASE, val); //发送数据
    while(ROM_SSIBusy(SSI1_BASE));  //等待发送完成
    return 1;
}
```

实验程序需要在 LCD 上显示当前 VC 端的电压值，VC 端的电压可以通过 DAC7311 的输出电压换算得到。

DAC7311 的输出电压计算公式：

$$V_{out} = V_{DD} \times \frac{D}{2^n}$$

式中V_{out}是 DAC7311 的输出电压，V_{DD}是 DAC7311 的输入电压，D是 Tiva M4 通过 SSI 传输给 DAC7311 的工作值，$D \in [0, 4095]$，n是 DAC7311 的 DA 位数，该值为 12。

电路中$V_{DD} = 3.3V$，故 DAC7311 的输出电压为$0 \sim 3.3V$，该电压通过 LMP7701 构成的双极性输出电路将电压值转换为$-3.3V \sim 3.3V$，即为 VC 端的电压。在程序设计中不涉及负值的处理，故可以将$-3.3V \sim 3.3V$在程序中提升为$0 \sim 6.6V$。程序中可以通过如下方式计算出 VC 端的电压。

$$\Delta V = 2 \times V_{DD} \times \frac{D}{2^n} \qquad \Delta V \in [0, 6.6V]$$

故 VC 端电压为：

$$VC = \begin{cases} 3.3 - \Delta V & 0 \le \Delta V \le 3.3 \\ \Delta V - 3.3 & 3.3 < \Delta V \le 6.6 \end{cases}$$

程序中扩大 1000 倍计算，显示成 mV 档。程序代码如下：

```
//计算VC端的电压值
VC_Value = (dac7311_val*3300*2) / 4096;
//根据3300（中间值，进行正负显示处理）
int show_val;
if(VC_Value > 3300)
{
    show_val = VC_Value - 3300;
}
else
{
    show_val = 3300 - VC_Value;
}
```

程序代码中的 VC_Value 即为计算公式中的ΔV，show_val 计算得出的都是正值，而根据程序数据跟 VC 端真实电压的对应关系可知当 VC_Value<3300（$\Delta V < 3.3$）时 VC 端电压为负值，LCD 上显示负值；VC_Value>3300（$\Delta V > 3.3$）时 VC 端电压为正值 LCD 上显示正值。程序代码如下：

```
for(i = 0; i < 4; ++i)
{
    if(VC_Value < 3300)
    {
        //在LCD上显示负号
        LCD_Draw_Char('-', LINE_TWO, 50);
    }
    else
```

```
    {
        //在LCD上清除负号
        LCD_Draw_Char(' ', LINE_TWO, 50);
    }
    //在LCD上显示VC端电压值
    LCD_Draw_Char('0' + data1[i], LINE_TWO, 60 + 8 * i);
}
```

不同的实验 VC 端以及 DAC7311 具有各自的初始默认值，VC 端的电压是通过 DAC7311 工作值计算获得。默认值在按键响应程序中设置。

A 宽带压控增益放大和衰减

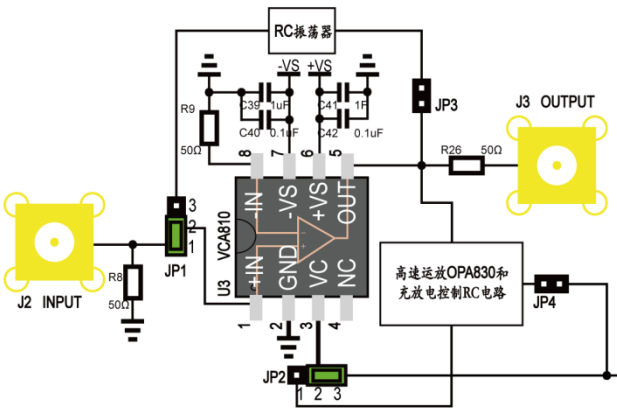
该实验初始默认值 VC 端电压需要为负压，保持在-1.13V。根据计算公式换算，在程序实现中保持 DAC7311 的工作值为 1350，计算得出的 VC 端的电压为-1.125V。

B 正反馈 RC 振荡器

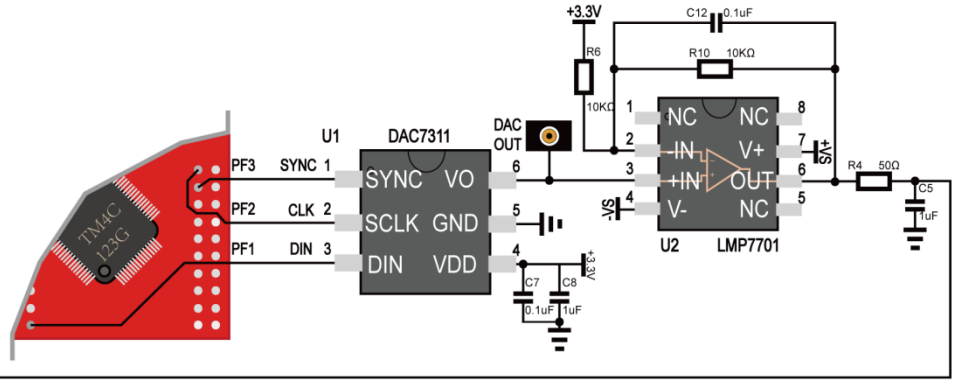
该实验初始默认值为 0 即可，此时 DAC7311 的工作值为 2048。

C 自稳幅闭环振荡器

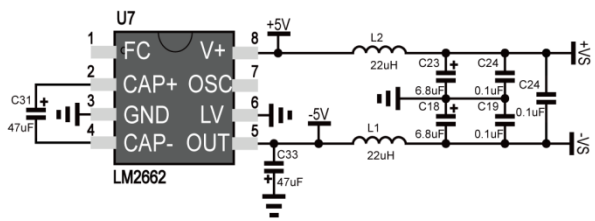
该实验初始默认值 VC 端电压需要为正压，保持在 1.17V。根据计算公式换算，在程序实现中保持 DAC7311 的工作值为 2775，计算得出 VC 端电压为 1.171V。



压控增益放大电路



双极性输出电路



电平转换及EMI滤波电路

宽带压控增益放大与衰减实验

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、高速压控增益模块连接完成，准备实验。

3、在高速压控增益模块上完成带宽压控增益放大与衰减实验的跳线连接，如图所示，短接图 1 JP1的1.2，以及JP2的2.3。

4、用实验套件里的两根高频连接线分别接在图 3 的J2和图 2 的J3上，其中J2连接到信号发生器，J3连接到示波器上。再万用表的红表笔接在跳线帽图 4 JP2上，黑表笔接地，用于测量VC的电压值大小。

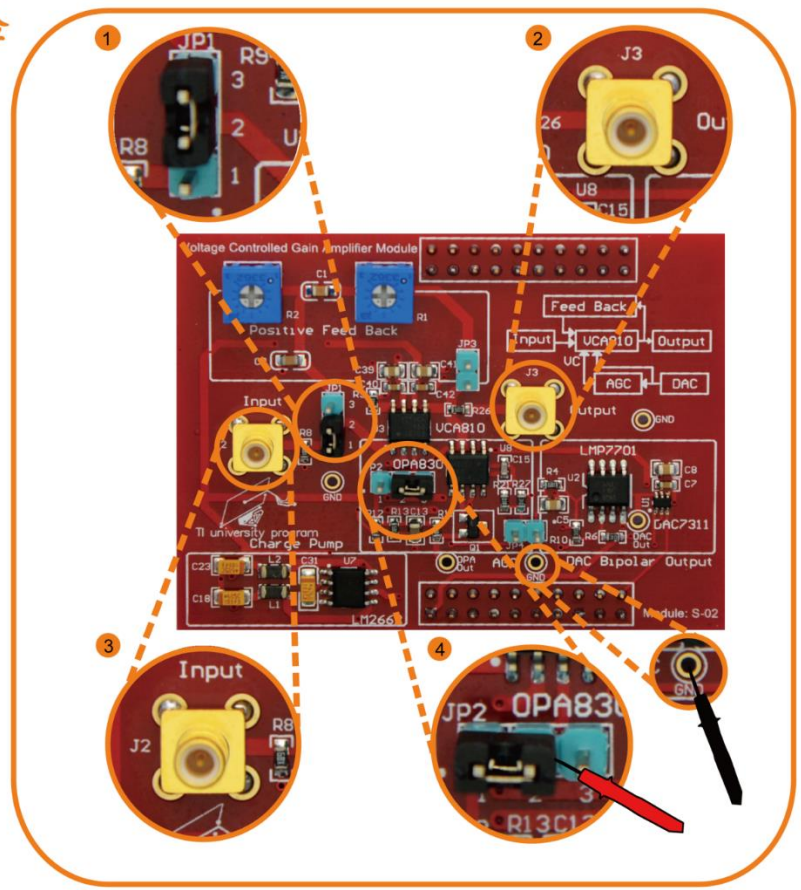
5、用信号发生器产生一个信号例如（10MHz 0.05Vpp），打开TIVA开关，通过液晶模块上的S1和S2按钮来调节VC的电压值（注意VC值要小于0），同时观察示波器上的输出信号，并记录，再计算增益与VC值的关系，查看增益线性度。

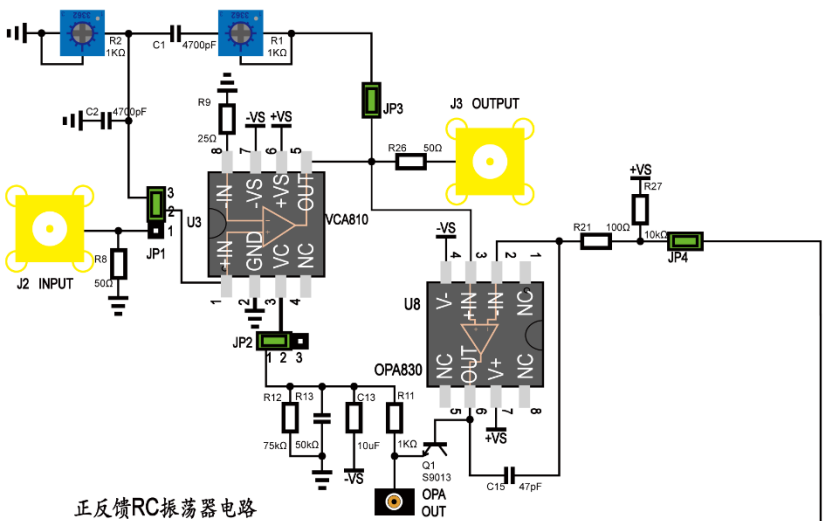
6、改变输入信号的幅值，例如（10MHz 2.5Vpp）重复步骤4，查看增益的线性度。

7、保持VC的值固定，例如（VC=-1.13V），保持输入信号的幅值，例如（Vin=1.5Vpp），改变输入信号的频率，测试增益，查看增益频率特性。

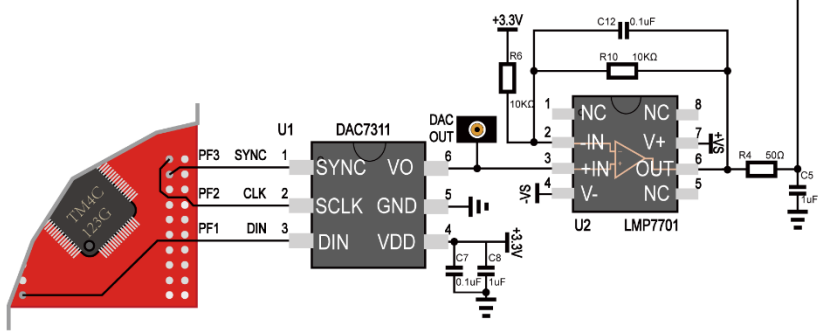
注意

连接仪表及跳线时断开电源。

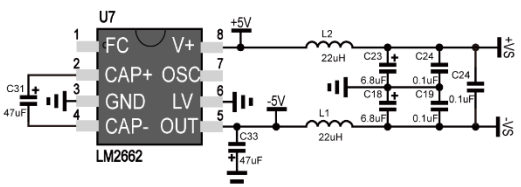




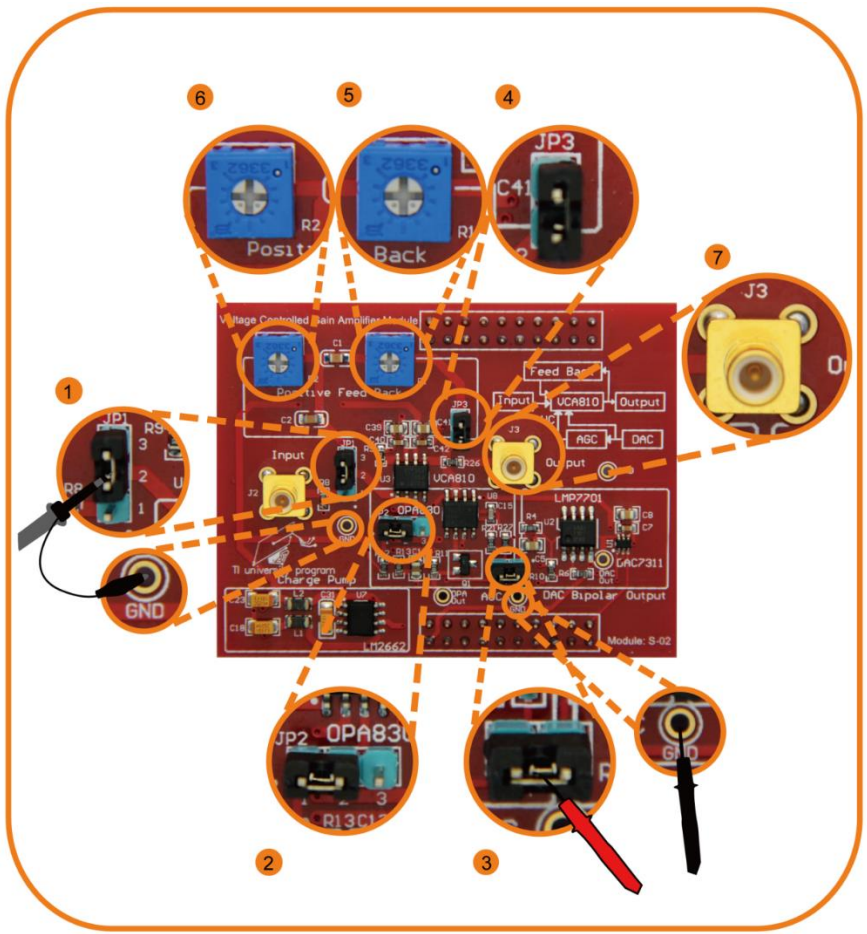
正反馈RC振荡器电路



双极性输出电路



电平转换及EMI滤波电路



正反馈RC振荡器实验

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、高速压控增益模块连接完成，准备实验。

3、在模块上完成正反馈RC振荡器的跳线连接，如图中的1.2.3.4所示，短接JP1的2.3，JP2的1.2,JP3以及JP4。

4、如图中1所示，用示波器测量RC振荡器产生的频率。如图中3所示，用万用表测量幅度控制电压的大小。如图中7所示，用高频连接线连接J3至示波器，观察输出

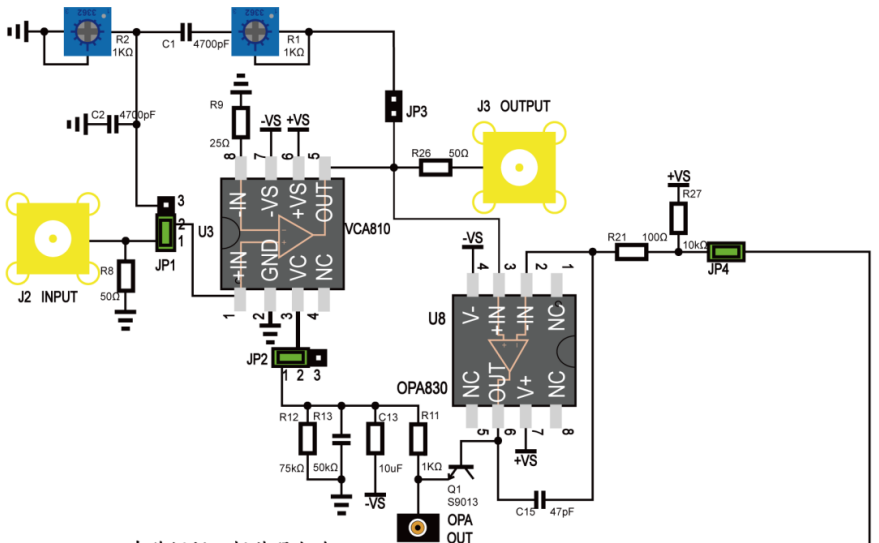
5、打开TIVA开关，通过调节滑动变阻器R1和R2的值来改变RC振荡器的输出频率。（注意同角度的调节滑变），再通过示波器可观察振荡器输出波形。

6、保持RC振荡器的输出频率不变，通过液晶模块上的S1.S2按钮来改变幅度控制电压的大小（JP4）。并测量输出幅度，查看振荡器幅度控制的线性度。

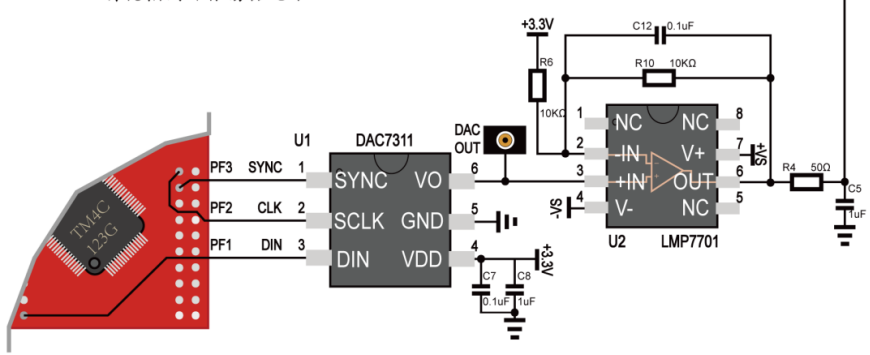
7、保持幅度控制电压大小不变，变化振荡频率，测试输出幅度，查看稳幅振荡情况。

注意

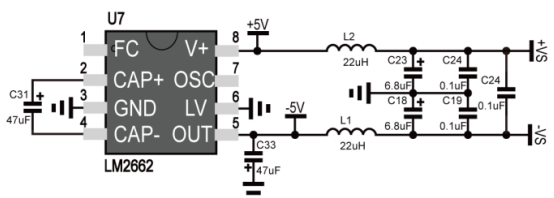
连接仪表及跳线时断开电源。



自稳幅闭环振荡器电路



双极性输出电路



电平转换及EMI滤波电路

自稳幅闭环振荡器实验

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、高速压控增益模块连接完成，准备实验。

3、在模块上完成自稳幅闭环振荡器的跳线连接，如图中的 1 . 2 . 4 所示，短接JP1的1.2，JP2的1.2,以及JP4。

4、用两根高频连接线分别接在图 5 的J2和图 3 J3上，其中J2连接到信号发生器，J3连接到示波器上。

5、打开TIVA开关，用液晶模块上的S1.S2按钮来调节幅度控制电压的大小（JP4）,测量方式如图中的 2 。保持其电压为1.17V左右。

6、用信号发生器产生一个频率固定的输入信号，例如（100KHz），再改变输入信号的幅度，测试AGC的幅度稳定能力。（需测量输出端的电压值和VC的电压值）

7、使幅度控制电压为0.033V左右（JP4）。重复实验步骤5。

8、变化输入信号频率，测试AGC的幅度稳定能力。

注意

连接仪表及跳线时断开电源。