

目录

第一章 方案理论分析	1
1.1 电容测量方案理论分析.....	1
1.1.1 直接法 (I-V) 测量电容	1
1.1.2 自动平衡电桥测量电容原理.....	2
1.1.3 电容测量影响因素分析	5
1.2 电感测量方案理论分析.....	5
1.2.1 谐振法测量电感.....	5
1.2.2 电感测量的其他影响因素	8
1.2.3 其他测量法	9
第二章 测量方案介绍	11
2.1 硬件电路搭建	11
2.1.1 供电模块.....	11
2.1.2 运算放大器模块.....	16
2.1.3 电容测量电路	16
2.2 软件工程搭建与程序编写	17
2.2.1 C2000 微控制器配置	17
2.2.2 主函数程序流程	19
2.2.3 中断函数流程	20
第三章 系统调试与测量结果	22
3.1 硬件调试问题与解决方案	22
3.1.1 电容测量电路中采样电阻切档	22
3.1.2 电感测量电路中谐振电容的选择	22
3.1.3 电感测量电路中杂散电容电阻对测量的影响	22
3.2 软件调试问题与解决方案	22
3.2.1 C2000 片内 RAM 与 FLASH 空间分配问题	22
3.3 测试结果	23
第四章 总结与展望	24
附录 A 串联电路中电容 D 值测量原理分析	25
附录 B 主程序代码	27

第一章 方案理论分析

1.1 电容测量方案理论分析

1.1.1 直接法 (I-V) 测量电容

为得到电容的容值与损耗正切角 (D) 值, 向待测电路中通入正弦激励信号, 采集电容两端的电压和电流, 再经由 C2000 微处理器分析处理。测量方案如图1-1所示。

图1-1中, C_x 为待测电容, R_C 为电容等效串联电阻, R_S 为采样电阻。设信号发生器的输出激励信号为 U_a 电容两端的压差 U_C 为:

$$U_C = U_a - U_b = A\angle\theta \quad (1-1)$$

通过电容的电流 I_C 为

$$I_C = \frac{U_b}{R_S} \quad (1-2)$$

通过以上的测量值, 可以得到待测电容的阻抗为:

$$Z_x = \frac{1}{j\omega C} + R_C = \frac{U_C}{I_C} = \frac{U_a - U_b}{U_b} R_S \quad (1-3)$$

其中, 损耗正切角为待测原件阻抗的实部与虚部之比:

$$D = \tan\delta = \frac{\Re(Z_x)}{\Im(Z_x)} = \frac{R_C}{\frac{1}{\omega C_x}} = \omega R_C C_x \quad (1-4)$$

从电路的阻抗模型分析也可知晓 D 值测量的原理。推导过程见附录 A。

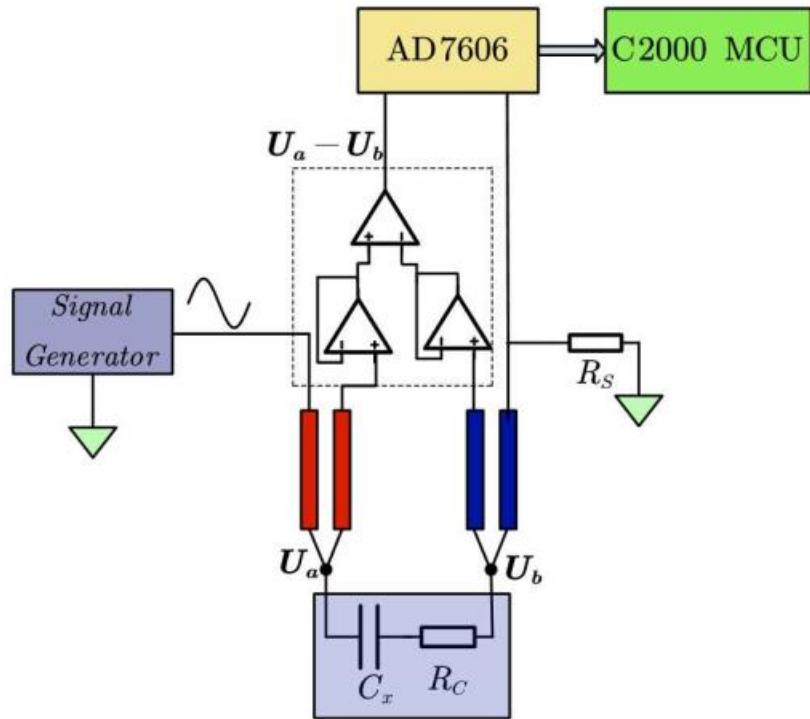


图 1-1 电容测量电路示意图

1.1.2 自动平衡电桥测量电容原理

与前者不同，大多数 LCR 表将难以采集与分析的交流激励信号转换成了易测的直流信号。这种方法被称作“自动平衡电桥法”。测量原理如图 1-2 所示：

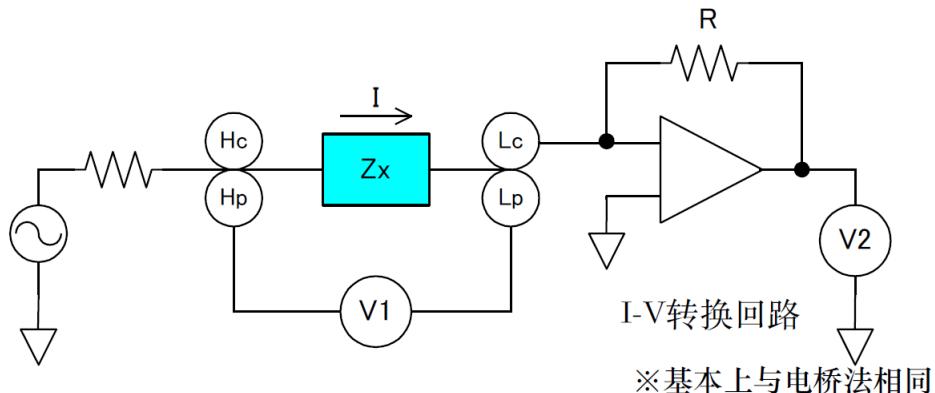


图 1-2 自动平衡电桥法测量原理

V1 电表用于测量待测原件两端的电压，V2 电表用于测量通过跨阻放大器转换成电压信号的电流信息。

$$Z_x = \frac{V_1}{I} = V_1 \frac{R}{V_2} \quad (1-5)$$

对原始测量电路稍加变换，得到图1-3的电路测量示意图，我们不难发现自动平衡电桥法的本质也是利用了电桥平衡的原理。

$$\frac{Z_x}{R} = \frac{V_1}{V_2} \quad Z_x = V_1 \frac{R}{V_2} \quad (1-6)$$

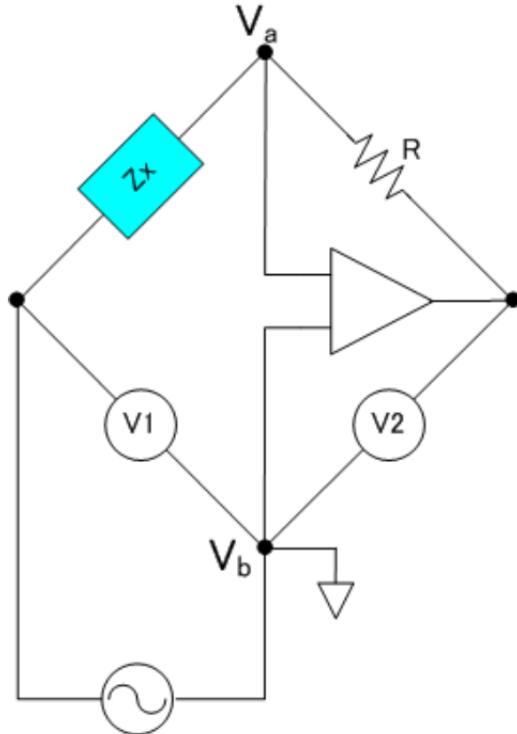


图 1-3 自动平衡电桥与电桥平衡电路的关系

低频情况下，自动平衡电桥表现较为良好。但是对于高频，尤其是电感测量的应用场景中，需要复杂的 I-V 转换电路，由此衍生出了 RF-IV 法，测量原理如图1-4所示。高频 LCR 仪就是采用这类方法制作出的。

Low Z type

$$\frac{V_1}{V_2} = \frac{1}{1 + \frac{2R}{Z_x}} = \frac{Z_x}{Z_x + 100} \quad (1-7)$$

High Z type

$$\frac{V_2}{V_1} = \frac{R}{2Z_x + R} = \frac{25}{Z_x + 25}$$

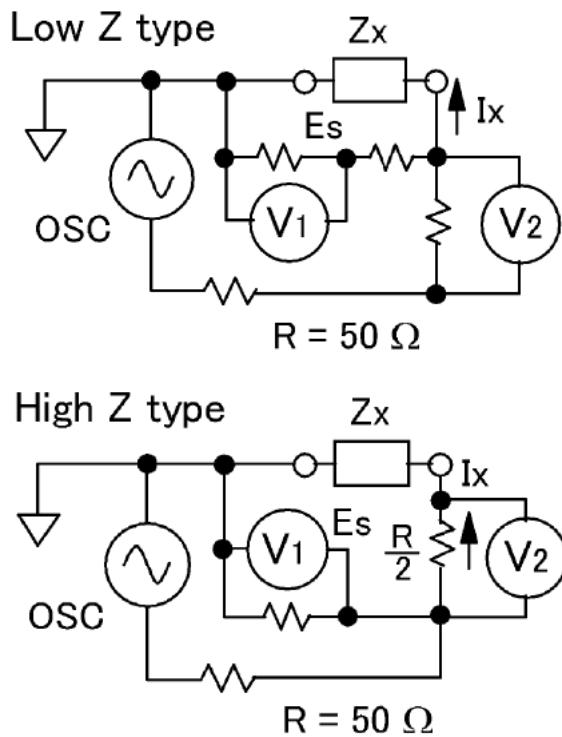


图 1-4 RF-IV 法测量原理（分为低阻态和高阻态测量两种电路）

不同测量方法的适用频段范围和阻抗范围如图1-5所示。（传输反射法采取了网络分析仪的原理，测量电路中待测原件对信号的反射系数，从而计算出其精确阻抗，涉及射频信号的应用，在此不作过多讨论。）

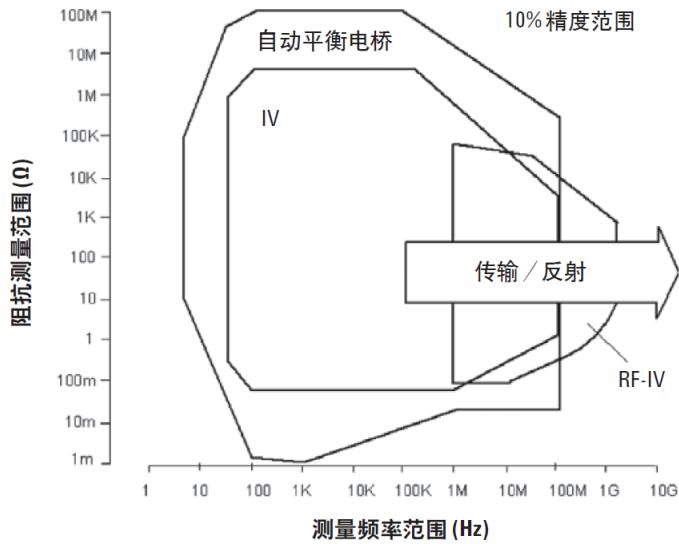


图 1-5 不同测量方法的适用频段范围和阻抗范围

表1-1总结了不同测量方法的适用范围与优劣供参考。

测量方法	优点	缺点	频率范围	阻抗范围	主要应用
电桥法	精度高, 测量成本低	手动平衡调节, 标准电阻需要换挡	DC-300MHz	$m\Omega - M\Omega$	高精度测量
谐振法	可测量高 Q 值	需要确定谐振电容, 每次测量要扫频	22k-70M	$\Omega - M\Omega$	高 Q 值器件
I-V 法	可测接地元件, 配合探针使用	受限于探针的互感线圈	10K-110M	$100m\Omega - 1M\Omega$	测接地和电路板
RF I-V 法	高频时精度高, 宽阻抗	受限于互感	1M-1.8G	$100m\Omega - 100K\Omega$	高频、射频元件
网络分析法 (传输反射法)	高频范围	阻抗范围小	300 以上	50Ω 附近	高频、射频元件
自动平衡电桥法	宽频率范围、宽阻抗范围	不适用于高 频	20-110M	$m\Omega - \text{几百 } M\Omega$	通用测量

表 1-1 各种测量阻抗方法比较

1.1.3 电容测量影响因素分析

采用直接 I-V 法, 测量值和真实值理论上没有误差。但实际应用中, 应考虑外部 ADC 的分辨率与输入极限, 选择与电容等效电阻相近的采样电阻 R_S 串联。采样电阻过小, 导致 U_b 测量过小, 采样电阻过大, 则会导致 $U_a - U_b$ 过小, ADC 读取到的数值极易受到外界电磁环境影响, 最终都会对测量不利。

1.2 电感测量方案理论分析

1.2.1 谐振法测量电感

本方法的测量原理如图1-6所示。

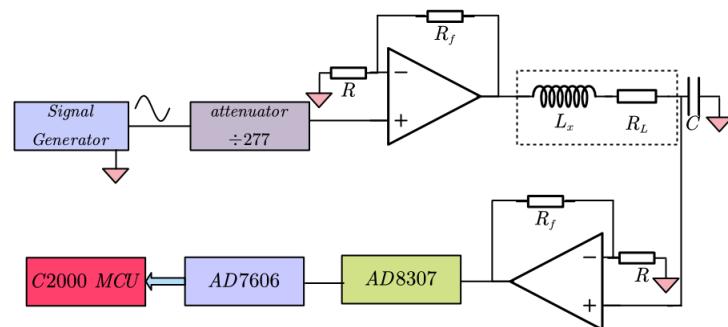


图 1-6 谐振法测量原理示意图

由信号发生器产生的一定幅度的扫频正弦激励信号, 经由两级衰减器衰减 277 倍之后, 经

由一级运算放大器组成两倍同相放大电路隔离后，进入谐振电路，与标称电容振荡，再经由一级运算放大器组成两倍同相放大电路隔离后进入检波器，检出关于输入正弦信号幅度指数相关的直流信号，送至外置 ADC 采集和微控制器进行处理，输出结果。

假设信号发生器的输入为 U_i ，经由衰减器和前级放大的输出为 U_o ，经由待测元件谐振之后的输出为 R_o 。谐振电路一旦进入谐振状态，表现为电路中阻抗最小，电流最大。理想情况下的 LC 串联谐振的条件为

$$\begin{aligned}\omega L_x &= \frac{1}{\omega C} \\ \omega &= \frac{1}{\sqrt{L_x C}}\end{aligned}\quad (1-8)$$

当考虑电感的等效串联电阻 (ESR, 图1-6中的 R_L)，根据谐振时电流最大原理，可根据待测元件两端的电压之比最大（极大值点）时的条件，得到实际的谐振频点：

$$\begin{aligned}\frac{U_o}{U_i} &= \frac{\frac{1}{j\omega C}}{R_L + j\omega L_x + \frac{1}{j\omega C}} = \frac{1}{j\omega C R_L - \omega^2 L_x C + 1} \\ \left| \frac{U_o}{U_i} \right|^2(\omega) &= f(\omega) = \frac{1}{(1 - \omega^2 L_x C)^2 + \omega^2 C^2 R_L^2}\end{aligned}\quad (1-9)$$

$$\omega|_{f'(\omega)=0} = \left\{ \begin{array}{l} 0 \\ -\sqrt{\frac{2CL_x - C^2 R_L^2}{2C^2 L_x^2}} \\ \sqrt{\frac{2CL_x - C^2 R_L^2}{2C^2 L_x^2}} \end{array} \right\}\quad (1-10)$$

取正频率，即式 1-10 中的第三个解，此时的输出相对于输入的比值为

$$\begin{aligned}\left| \frac{U_o}{U_i} \right|^2(\omega)_{max} &= \frac{1}{\left(\frac{2L_x - CR_L^2}{2L_x} - 1 \right)^2 + \frac{CR_L^2(2L_x - CR_L^2)}{2L_x^2}} \\ &= \frac{1}{\left(\frac{1}{2} \frac{CR_L^2}{L_x} \right)^2 + \frac{CR_L^2}{L_x} - \frac{1}{2} \frac{C^2 R_L^4}{L_x^2}} \\ &= \frac{1}{-\frac{1}{4} \alpha^2 + \alpha} \\ where \quad \alpha &= \frac{CR_L^2}{L_x} > 0\end{aligned}\quad (1-11)$$

实际测量中，电路的谐振频率视为近似等于 $\frac{1}{\sqrt{LC}}$ ，根据题目要求，电感测量误差 $0.95L - 1.05L$ ，则由 $\omega = \frac{1}{\sqrt{LC}}$ 得到频率测量误差应满足 $0.9759\omega - 1.0260\omega$ 。

$$\begin{aligned}
 0.9759 &\leq \frac{\omega_{\text{测量值}}}{\omega_{\text{理想值}}} \leq 1.0260 \\
 0.9759^2 &\leq \frac{2CL_x - C^2R_L^2}{2L_xC} \leq 1.0260^2 \\
 0.95238081 &\leq 1 - 0.5\alpha \leq 1.052676 \\
 0 &\leq \alpha \leq 0.0952
 \end{aligned} \tag{1-12}$$

进一步可以得到满足此误差范围的测量 Q 值范围 ($|\frac{U_o}{U_i}|$)

$$|\frac{U_o}{U_i}| \geq 10.5042 \tag{1-13}$$

待测元件的品质因数 (Q) 定义为

$$Q = \frac{\omega L_x}{R_L} \tag{1-14}$$

将谐振频率代入上式中，得到

$$\begin{aligned}
 Q &= \sqrt{\frac{2CL_x - C^2R_L^2}{2C^2R_L^2}} \\
 &= \sqrt{\frac{1}{\alpha} - \frac{1}{2}}
 \end{aligned} \tag{1-15}$$

下面探讨品质因数真实值与电路中谐振放大倍数的关系。放大倍数在测量中也被视为是测量得到的 Q 值。定义误差 η 为

$$\begin{aligned}
 \eta &= \frac{Q_{\text{真实值}} - Q_{\text{测量值}}}{Q_{\text{真实值}}} \\
 &= 1 - \frac{Q_{\text{测量值}}}{Q_{\text{真实值}}} \\
 &= 1 - \frac{\sqrt{2}CR_L \sqrt{\frac{1}{(\frac{2L_x - CR_L^2}{2L_x} - 1)^2 + \frac{CR_L^2(2L_x - CR_L^2)}{2L_x^2}}}}{\sqrt{C(2L_x - CR_L^2)}} \\
 &= 1 - \sqrt{\frac{2}{(1 - \frac{1}{4}\alpha)(2 - \alpha)}}
 \end{aligned} \tag{1-16}$$

利用式 1-13，可以反解出 α :

$$\alpha = \frac{2}{2Q^2 + 1} \tag{1-17}$$

于是，误差可以转换为关于元件真实 Q 值的函数式：

$$\eta(Q) = 1 - \frac{2Q^2 + 1}{Q\sqrt{4Q^2 + 1}} \quad (1-18)$$

其函数图像如图1-7所示。可以看到，在题目要求的误差范围内，如果测量误差很小时，理论上可以测量 Q 大于 2.73 的元件。

结合式 (1-13) 和 (1-18)，可以综合得出此方案在待测元件 Q 值为 10-110 左右表现最佳，误差最小。

根据《中华人民共和国国家计量校准规范》高频 Q 表校准规范(征求意见稿)，高频 Q 表的计量范围在 Q=10-500 之间，最小极限值与计算结果吻合，而对于过高 Q 值得元件，则需要采用其他方法进行测量。

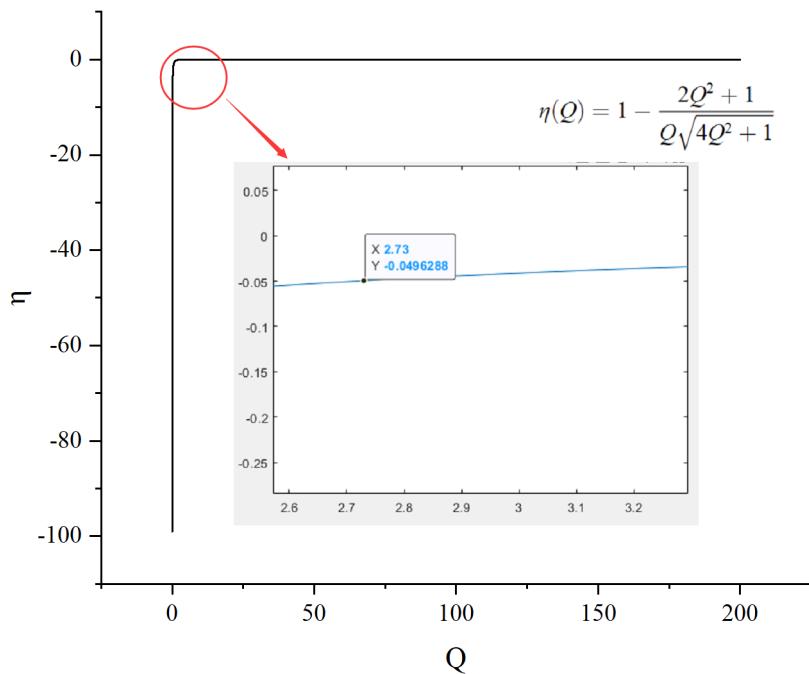


图 1-7 误差-真实 Q 值关系

1.2.2 电感测量的其他影响因素

实际电路中，由于电路板设计、焊接工艺、以及谐振电容的质量问题，应考虑电容的等效串联电阻、电路板的寄生电阻（电容、电感暂时不考虑），等效电路变为图1-8所示的电路。

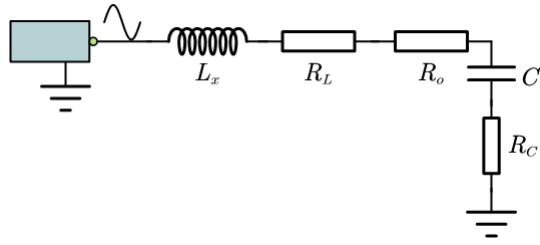


图 1-8 考虑电容 ESR 及板上寄生电阻影响的等效测量电路

其中, R_o 为电路板的寄生电阻, R_C 为电容的等效串联电阻。此时的实测谐振放大倍数为:

$$\begin{aligned} \frac{U_o}{U_i} &= \frac{\frac{1}{j\omega C} + R_C}{j\omega L + R_C + R_o + \frac{1}{j\omega C} + R_C} \\ &= \frac{1 + j\omega CR_C}{-\omega^2 CL_x + j\omega C(R_L + R_o + R_C) + 1} \end{aligned} \quad (1-19)$$

谐振频率认为近似等于理想化 RLC 电路的谐振频率, 故将 $\omega = \frac{1}{\sqrt{L_x C}}$ 代入, 此时谐振放大倍数为:

$$\begin{aligned} \frac{U_o}{U_i} &= \frac{1 + j\omega CR_C}{j\omega C(R_L + R_o + R_C)} \\ &= \frac{1}{R_C + R_o + R_L} [R_C + j(-\frac{1}{\omega C})] \\ |\frac{U_o}{U_i}|_{max} &= \frac{1}{R_C + R_o + R_L} \sqrt{R_C^2 + \frac{L}{C}} \\ \frac{Q_{\text{测量值}}}{Q_{\text{理想值}}} &= R_L \sqrt{\frac{CR_C^2 + L}{(R_C + R_o + R_L)^2 L}} \\ &\approx \frac{R_L}{R_C + R_o + R_L} \end{aligned} \quad (1-20)$$

应用中, 电容的容值在 pf 级别, 电阻阻值为 Ω 级别, 分子中 CR_C^2 项非常小, 为便于分析, 忽略不计。

由式 (1-20), 可以看出, 当测量高 Q 值得元件时, 元件本身的 R_L 过小, 导致 R_C 和 R_o 的电阻效应较明显, 测量结果与实际值相比偏小。

应注意到, 上述分析只考虑在 R_C 和 R_o 不随频率变化或者变化很小时对测量结果的影响, 实际问题中, 电容的 ESR 变化情况较为复杂, 无法用简单的阻抗模型分析得到。

1.2.3 其他测量法

除上述方法, 还有利用耦合电路进行测量、基于网络分析仪的反射法测量, 与电容测量方式类似, 共同思想就是利用元件本身的等效串联电阻产生的衰减加以计算。此类方法通常需要输出阻抗为 50Ω 的信号源和输出阻抗为 50Ω 的测量仪器。由于组内并未深入研究射频电路的应用, 故不再赘述。图1-9至图1-10展示了一些方法的测量原理。

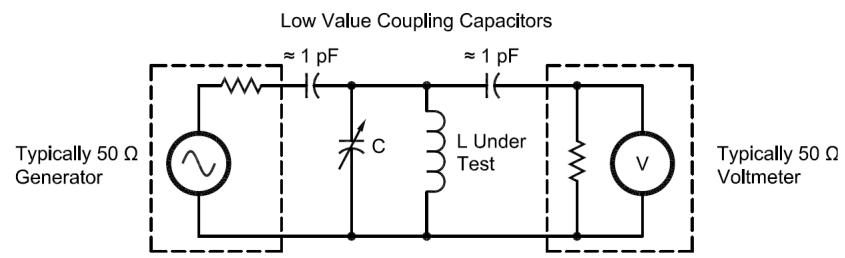


图 1-9 椅合电容传输测量法示意图

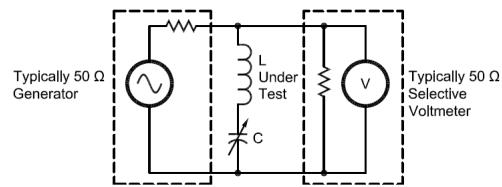


图 1-10 并联 LC 传输测量法示意图

第二章 测量方案介绍

2.1 硬件电路搭建

本题的整体电路如图2-1所示。下面对电路中部分电路做详细介绍和说明。

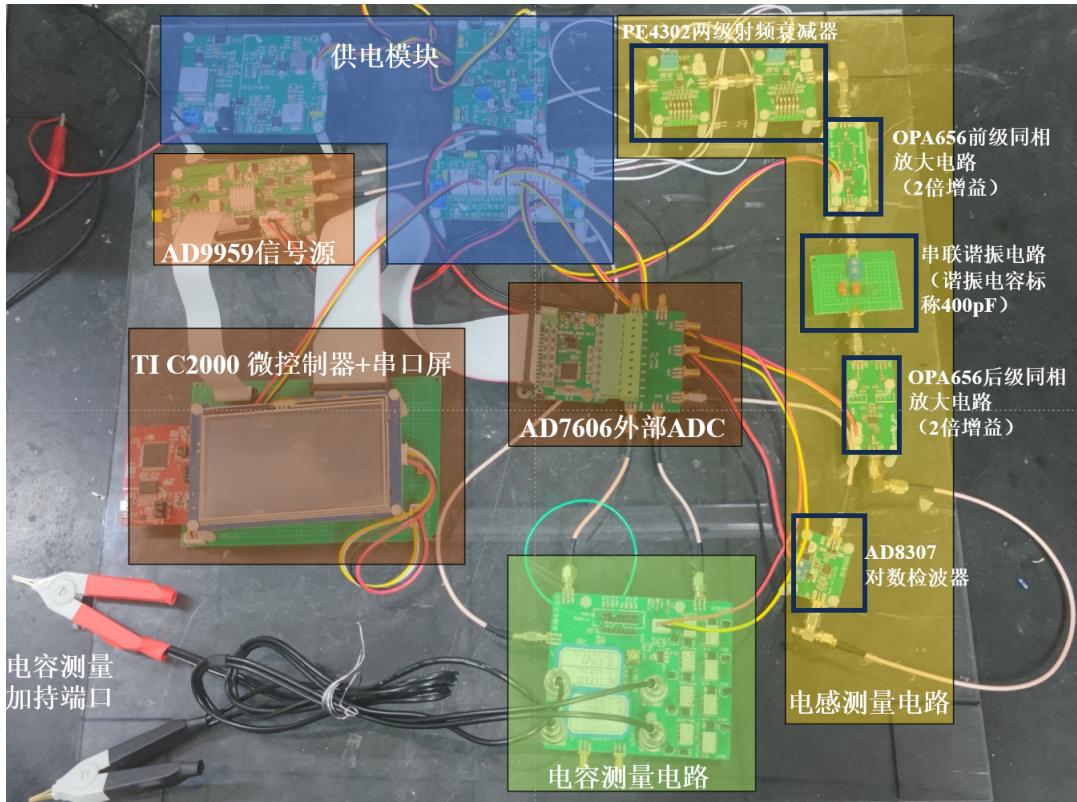


图 2-1 测量电路实体图

2.1.1 供电模块

供电部分采用 TPS55330 和 TPS54360 组合的升降压开关电源、以及 LT3045、LT3094 组合的正负线性稳压电路、和拓展接口板组成。其中，图2-2展示了以 TPS55330 芯片为核心的升压电路原理图。

开关电源涉及以下几个设计步骤：

1. 选择开关频率 (f_{freq} 值)：开关频率的高低需要权衡利弊。开关频率越高，电感值越低，输出电容越小，电路板尺寸越小。开关频率越低，电路板尺寸越大，但转换效率更高。通常会将频率设定为最低可容忍效率，以避免外部元件过大。开关频率可以通过式2-1计算得到。在本设计方案中，开关频率为 579.02kHz

$$f_{freq}(\text{kHz}) = 57500 \times f_{sw}(\text{kHz})^{-1.03} \quad (2-1)$$

2. 选择电感：保守估计电源效率 (η_{EST}) 在 85%，输出电流 (I_{OUT}) 为 2A，则根据式2-2计算

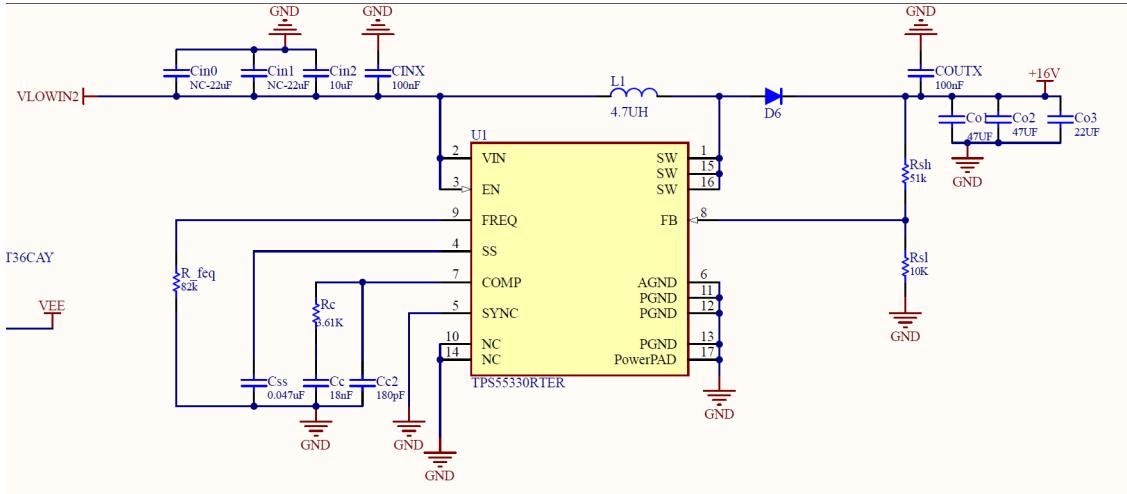


图 2-2 TPS55330 芯片为核心的升压电路原理图

出最大输入电流 (I_{INDC}) 为 4.34A 左右。利用式2-3近似计算所需的最小电感值。本方案中，经计算采取标称 $4.7\mu H$ 的贴片功率电感。

$$I_{INDC} = \frac{V_{OUT} \times I_{OUT}}{\eta_{EST} \times V_{IN \min}} \quad (2-2)$$

$$L_O \min \geq \frac{V_{IN}}{I_{INDC} \times K_{IND}} \times \frac{D}{f_{SW}} , D \neq 50\%, V_{IN} \text{ with } D \text{ closest to } 50\% \quad (2-3)$$

3. 选配输出电压调节电阻：根据式2-4可以计算出所需的输出电阻。考虑到通过电阻器分压器的漏电流和对 FB 引脚的噪声去耦，R2 的最佳值约为 $10 k\Omega$ 。本方案中选取 $R1=51k\Omega$, $R2=10k\Omega$ 。

$$V_{OUT} = 1.229V \times \left(\frac{R1}{R2} + 1 \right) \quad (2-4)$$

4. 选取输入输出电容：理想情况下，不考虑电容的 ESR，根据式2-5可计算出最小的输出电容值。至于输入端，应至少使用 $4.7\mu F$ 的陶瓷输入电容。为满足纹波和/或瞬态要求，可能需要额外的输入电容。本方案中，采用不同大小的电容，尽量减小输入端的纹波。

$$C_{OUT} \geq \frac{D_{max} \times I_{OUT}}{f_{SW} \times V_{RIPPLE}} \quad (2-5)$$

5. 设置软启动时间：选择合适的电容器来设置软启动时间，避免过冲。本例中使用的是 $0.047 \mu F$ 陶瓷电容器。

6. 选择肖特基二极管：TPS55330 的开关频率很高，需要高速整流以获得最佳效率。确保二极管的平均和峰值额定电流超过平均输出电流和电感器峰值电流。此外，二极管的反向击穿电压必须超过调节输出电压，额定功率需满足式2-6

$$P_D = V_D \times I_{\text{OUT}} \quad (2-6)$$

7. 补偿控制回路: TPS55330 需要外部补偿, 以使得在不同应用场景下优化环路响应。依据式2-7, 可计算所需的补偿电阻大小, 其中 G_{ea} 为芯片内部误差放大器的跨导, 在芯片手册的电气参数中给出, f_{BW} 为控制环路带宽, 为 10KHz, $K_{PS}(f_{BW})$ 为功率增益, 取 13.3dB。(这些值均参考 TPS55330 芯片手册, 为估计值, 设计电源时, 环路带宽等参数需要借助网络分析仪来标定) 本方案中, $G_{ea} = 360\mu S$, 计算得补偿电阻 $R_3 = 3.66k\Omega$.

$$R3 = \frac{1}{\left(G_{ea} \times \frac{R2}{(R1+R2)} \times 10^{\frac{K_{PS}(f_{BW})}{20}} \right)} \quad (2-7)$$

依据式2-8可计算得补偿电容所需大小。本例中, $C_4 = 18nF$, 为了环路补偿性能更加出色, 推荐焊接 C5, 本方案中取 180pF。

$$C4 = \frac{1}{2\pi \times R3 \times \frac{f_{BW}}{10}} \quad (2-8)$$

为运算放大器供电的正负电源由后级的降压电路产生。降压电路所使用的芯片为 TPS54360. 其设计思路与升压电路的步骤基本类似, 遵从芯片手册的推荐设计。降压部分的原理图如图2-3所示。

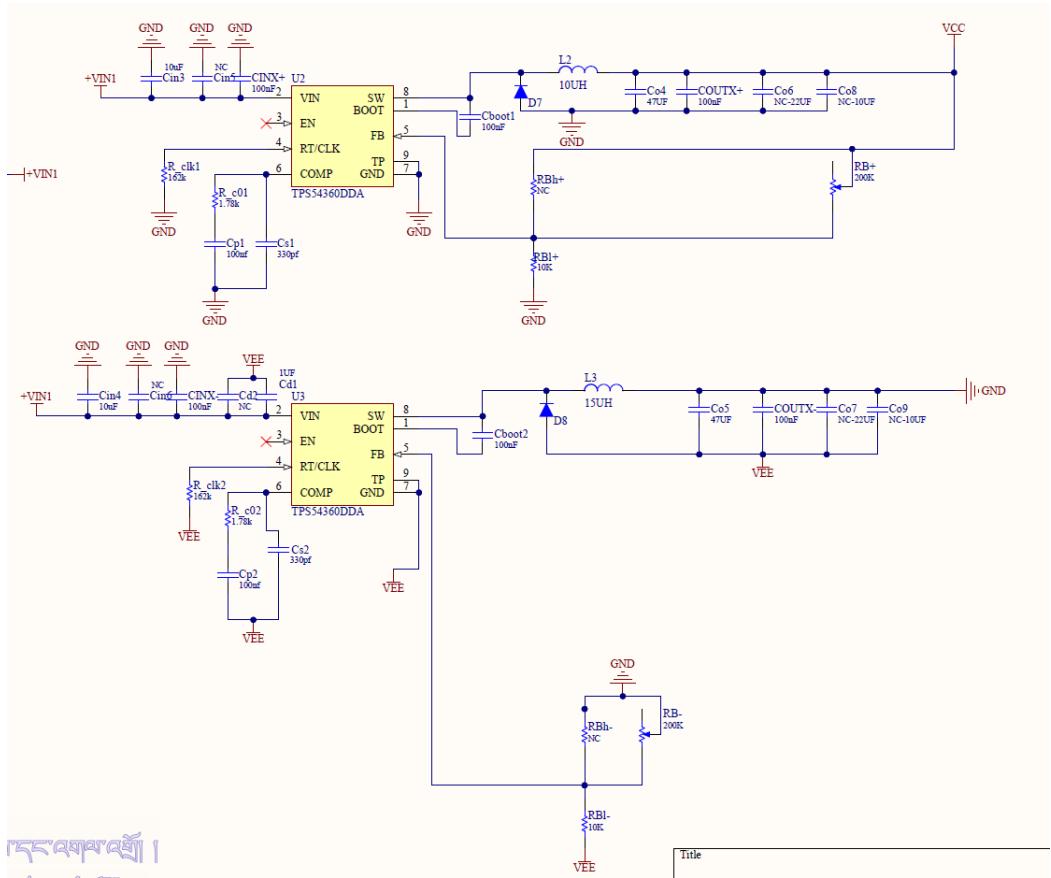


图 2-3 TPS54360 正负降压电路原理图

升降压开关电源整体的相关参数见表2-1。

表 2-1 升降压压电路参数

参数	值
输出电压	7.36 V
输入电压	7 V
最大输出电流	3.5 A
转换效率	85 %
纹波	$\approx 80\text{-}120\text{mVpp}$

在开关电源的后级，为了使输出电压满足运算放大器、信号源等的需要，使用线性稳压电源输出正负 5V。线性稳压电源的芯片采用 LT3045 和 LT3094。稳压电路的原理图如图2-4所示。

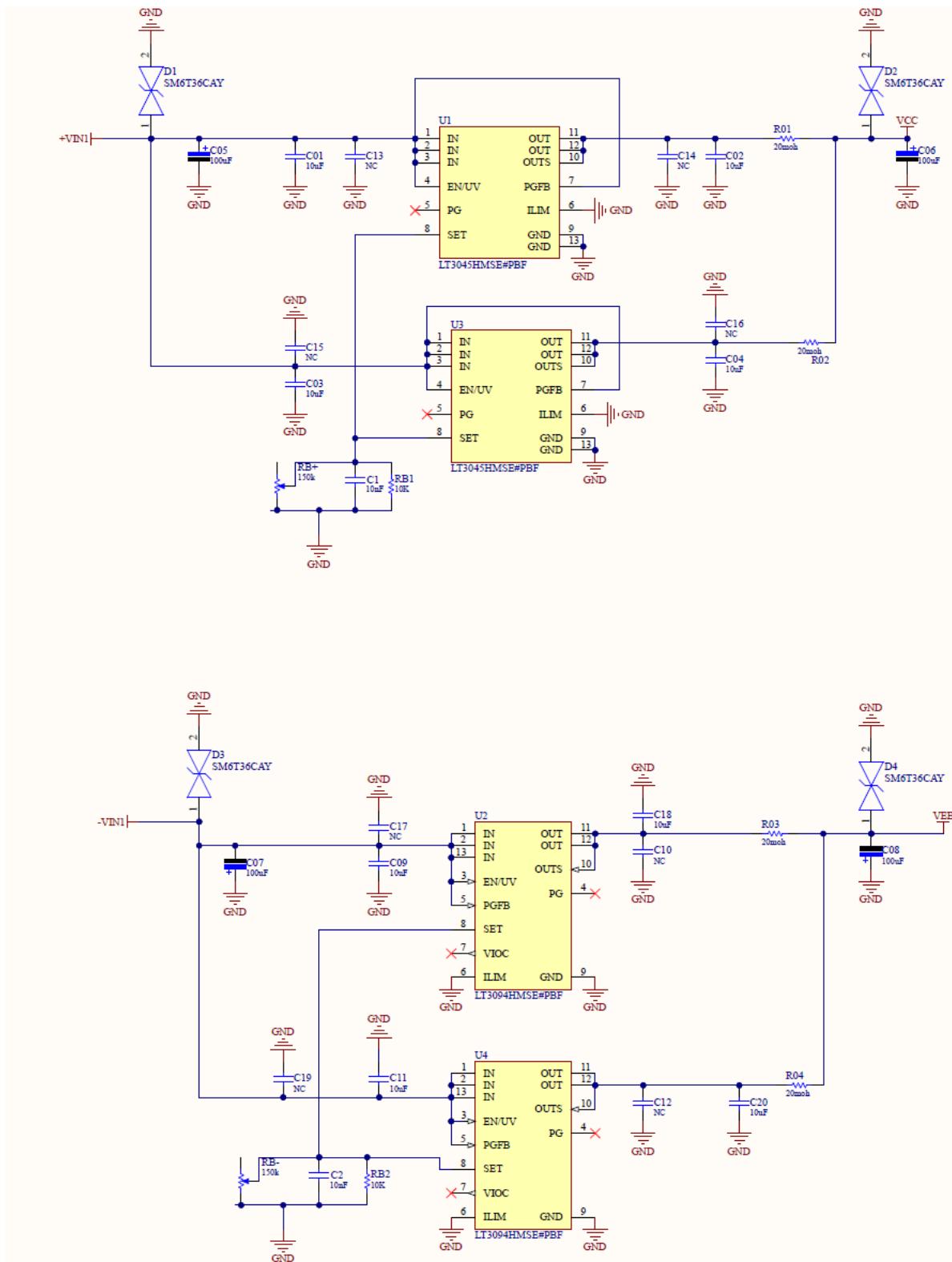


图 2-4 LT3045 和 LT3094 稳压电路原理图

起输出电压调节作用的电阻为串联在 8 号引脚 (SET) 的电阻 R_{SET} 。依据芯片手册，取 $R_{SET} = 49.9\text{k}\Omega$ 。为便于调试，增加了并联的电位器，起到微调的作用。

2.1.2 运算放大器模块

在电感测量电路中，为了使 AD9959 板载运放输出不受谐振电路的影响从而发生自激，需要在前后各用一个输入高阻抗输出几乎为零阻抗的宽带运放电路进行隔离。采用 OPA656 搭建 2 倍同相放大电路，电路原理图如图2-5所示。OPA656 自身的相关参数见于表2-2.

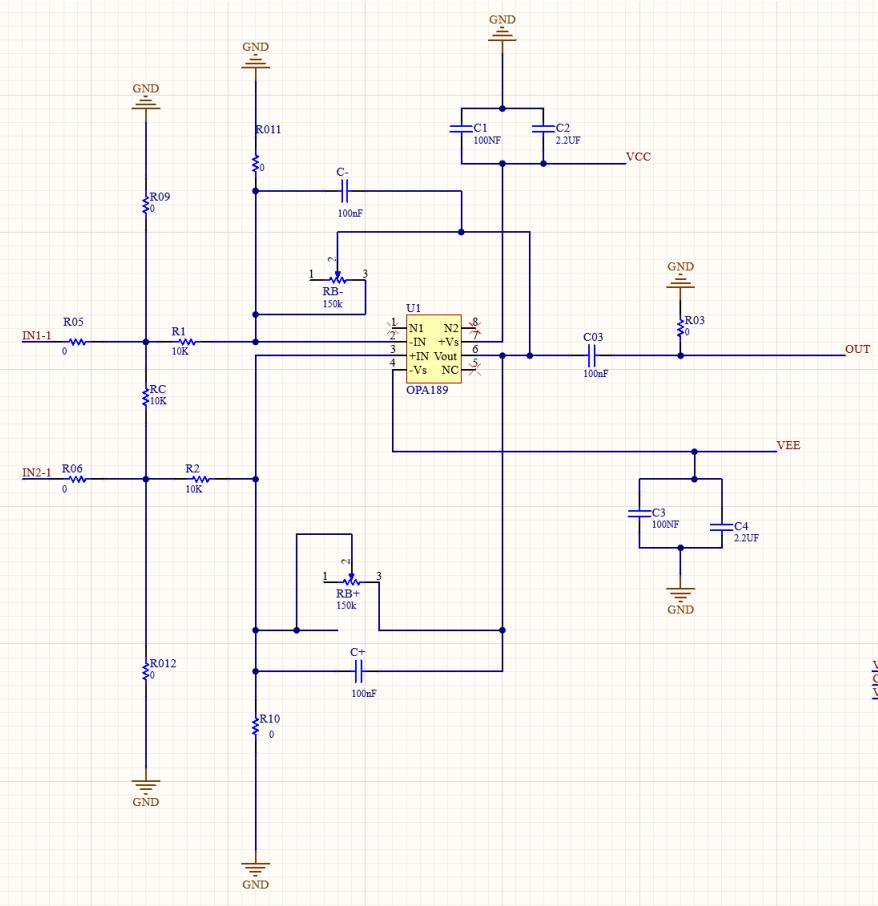


图 2-5 通用运放电路原理图

表 2-2 OPA656 相关参数

参数	值
结构	FET / CMOS Input, Voltage FB
通道数	1
供电范围	8-12 V (正负 5 V=10 V)
增益带宽积	230 MHz

2.1.3 电容测量电路

电容测量电路由测试夹接口、三运放仪表放大器、采样电阻与继电器组成。三运放仪表放大器选择 AD8065，相关参数见表2-3。原理图如图2-6所示，当原理图中的 R5 不焊接时，前级

的两个运放分别做电压跟随器，送至后级的放大器进行相减运算。为了测试不同挡位的采样电阻，设计了继电器对电阻切挡。

表 2-3 AD8056 相关参数

参数	值
结构	FET Input
通道数	1
供电范围	5 V to 24 V (正负 5 V=10 V)
增益带宽积	145 MHz
输入失调电流	1 pA

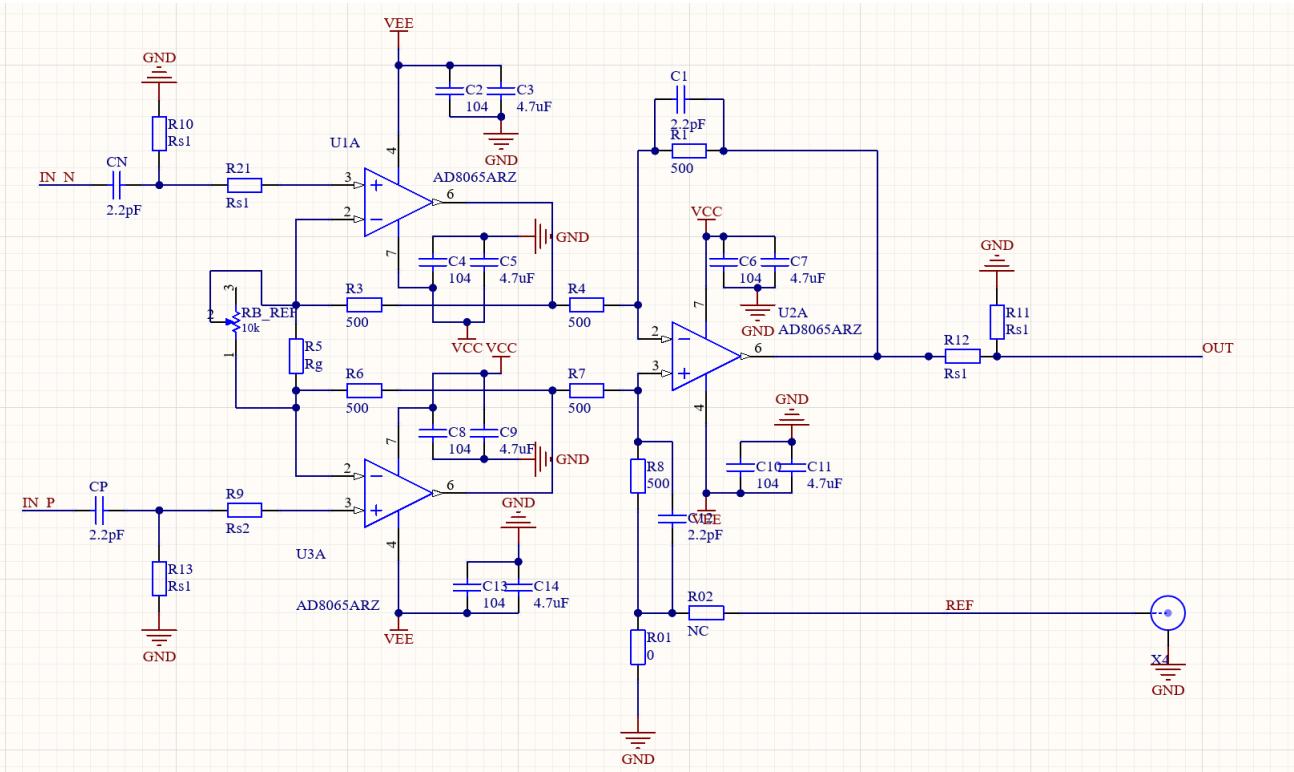


图 2-6 三运放仪表放大电路原理图

2.2 软件工程搭建与程序编写

2.2.1 C2000 微控制器配置

2.2.1.1 驱动 AD7606 进行串联模式下采样

AD7606 与 C2000 的控制接口如图2-7所示。OSI0-2 用于设置过采样，本次题目的测量均在低频 (DC-1k)，故过采样位均置零。SER 引脚用于配置 AD7606 模式，本方案中工作在串行模式。CO-A 和 CO-B 引脚用于启动模拟输入通道上的转换，前者启动 V1-V4 前四个通道

转换，后者启动 V5-V8 后四个通道的转换。两者转换启动信号由 C2000 微控制器的 EPWM 提供。REST 引脚用于重置 AD7606 的输出寄存器。RD/SC 用于外部时钟输入，串行模式下读取 AD7606 的输出值。本方案中调用了 C2000 的 SPI 接口的时钟，频率为 2MHz。由此也可以计算出 ADC 的最大采样率（TBCLK=100MHz 时，EPWM 工作在增减计数模式下）为 $\frac{1}{\frac{1}{100MHz} \times 5000 \times 2} = 10kHz$ 。DB7，即 DoutA，用于输出 ADC 的转换数值。

需注意，经实测，SPI 时钟设置为 2MHz 以上时，时钟波形严重失真，无法作为驱动信号。且 C2000 无并行通信接口，故本方案不使用并行状态下 AD7606 的采样功能。

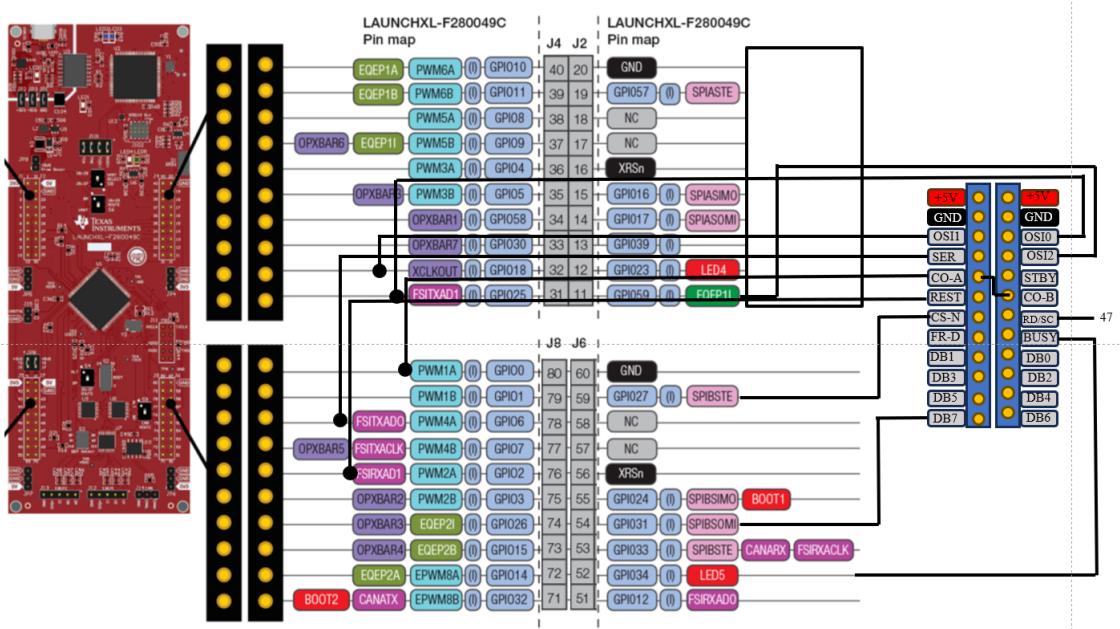


图 2-7 AD7606 与 C2000 的控制接口

2.2.1.2 驱动 AD9959 数字信号合成器

AD9959 与 C2000 的控制接口如图2-8所示。SCLK 与 C2000 的 GPIO 模拟时钟（翻转）相连，UP(IO_UPDATE) 引脚由 C2000 的某个 GPIO 产生上升沿将数据从串行 I/O 端口缓冲器传输到活动寄存器。CS 用于微控制器 SPI 总线有多个从机时，作为使能信号输入共用总线。SDIO0 为波形幅值、相位、频率等数据的串行输入引脚。RST 用于重置 AD9959 的寄存器值。

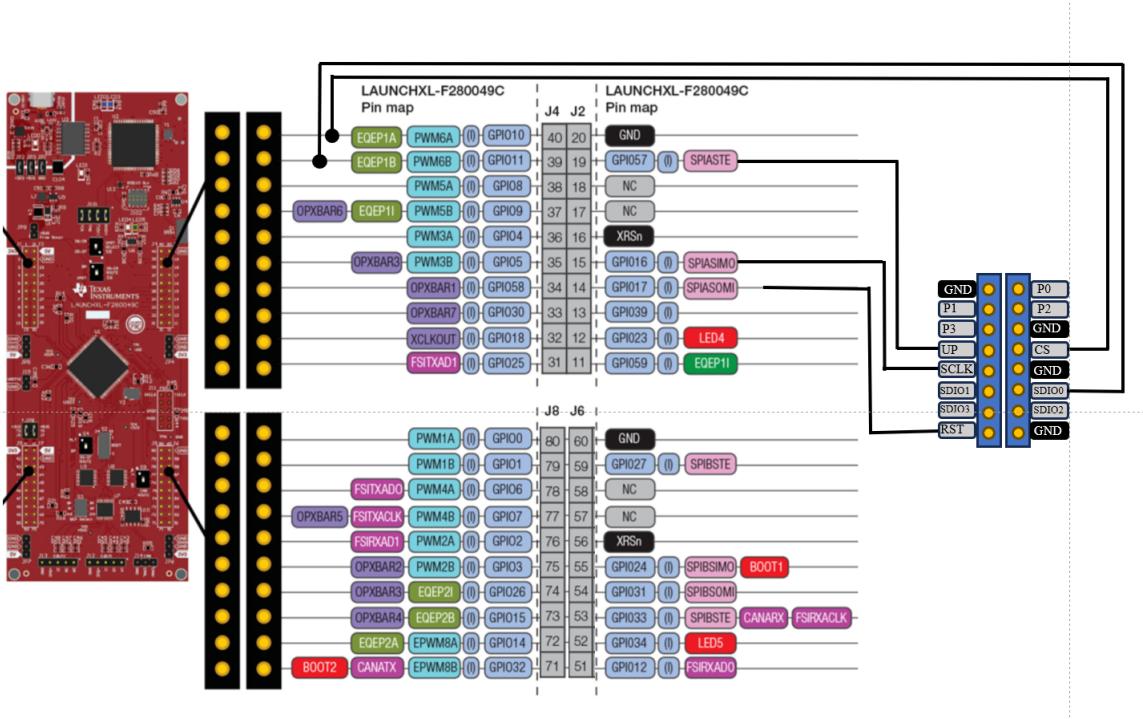


图 2-8 AD9959 与 C2000 的控制接口

2.2.2 主函数程序流程

主程序的相关代码详见附录 B。主要流程如图2-9所示。其中，在电感参数测量时，为更加精确地找到谐振频率，扫频时采用了分段扫频的方法，即当存储的直流电压值小于当前时刻的测量值，则更新测量值；当存储的电压值已经大于当前时刻地测量值，则依扫频步长回退激励信号的频率，继续测量，并缩小扫频步长。若步长已经小于等于 1kHz，再缩小扫频步长已经对测量影响极小，可以停止步长改变。以此循环执行，直至找到直流电压最大的时刻的频率，再依据该频率计算当前元件的相应参数。

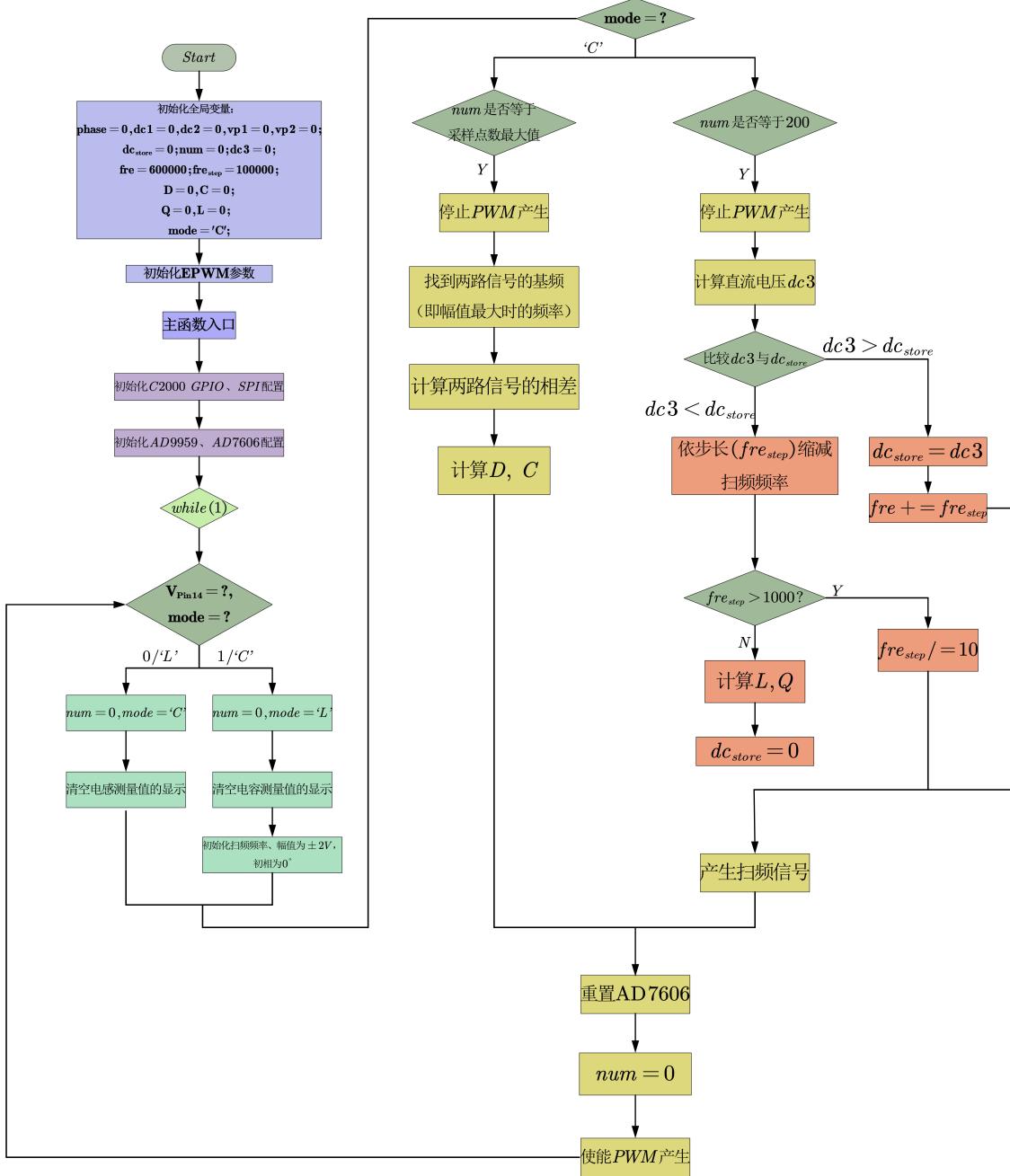


图 2-9 主函数程序流程

2.2.3 中断函数流程

中断函数流程如图2-10所示。主要任务是获取 AD7606 的采样数据并存储，以便后续的 FFT 计算与参数计算。

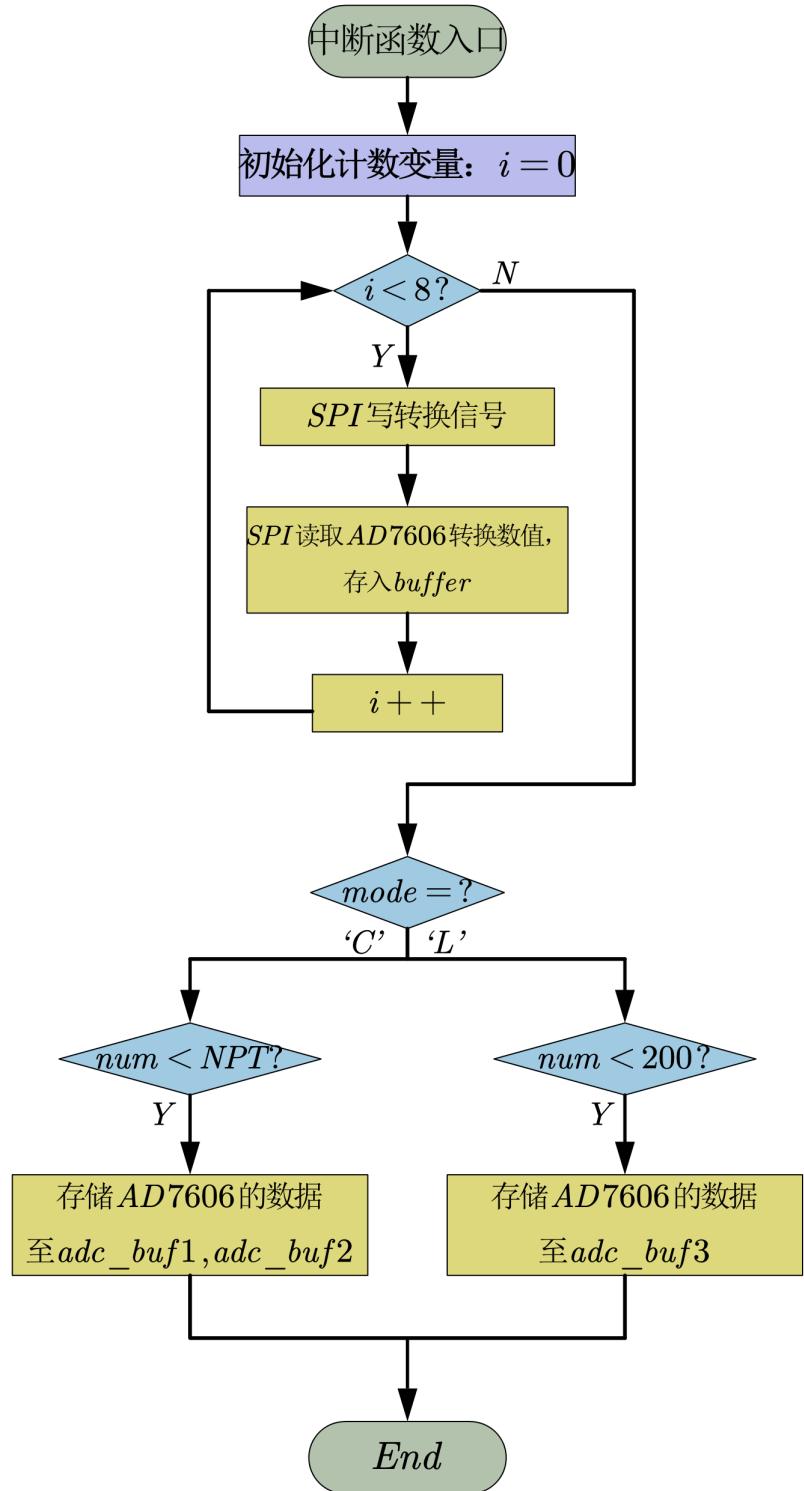


图 2-10 中断函数程序流程

第三章 系统调试与测量结果

3.1 硬件调试问题与解决方案

3.1.1 电容测量电路中采样电阻切档

实际测试中，我们发现采样电阻选择 159Ω （最小挡位）时，测试结果最为精确，采样电阻越大，容值感值的测量偏差越大。我们猜想测量不准确的原因是采样电阻过大导致待测元件分压急剧减小，进一步导致差分信号不准确，导致测量结果偏离。实际结果也并非像预想的采用与待测元件估算的 ESR 相近的采样电阻测量时效果最好。目前暂时的解决办法是放弃切档的方法，针对不同电气特性的电容元件，采用同一个采样电阻（ 159Ω ）进行测量，组内同学针对此问题的探究仍在进行。

3.1.2 电感测量电路中谐振电容的选择

在电路制作过程中，我们采用了不同种类的电容制作谐振电路，包括 CBB、收音机四联电容、云母电容。这三种电容的温漂特性较其他种类的更优越、D 值更小。经过测试，发现使用收音机四联电容时，谐振电路中的放大倍数相较理论推导值小 5-15% 不等。对于感值测量影响很小，但对于 Q 值测量影响巨大，且随着谐振频率变化，放大倍数呈现非线性的增长。这导致我们无法采用拟合的办法修正测量值。云母电容的谐振效果最好，能够最大限度地保证放大倍数近似为 Q 值。

3.1.3 电感测量电路中杂散电容电阻对测量的影响

方案验证初期，我们对电感谐振电路进行了重新设计，将前后级放大隔离地同相放大电路与谐振电路布局在同一块 PCB 上，并且为了与收音机四联电容配合使用，自行绘制收音机电容的封装，以验证初期的收音机电容谐振效果较好的猜想。在推翻这一猜想之后，重新使用云母电容作为谐振电容时，依然将其焊接到之前的 PCB 上。最终的测量效果显示 Q 值偏差较大，谐振频率与 Q 表的标定谐振频率相差过大。于是我们猜测，由于 PCB 设计欠佳，不规则铺铜等增加了电路中的杂散电容与杂散电阻，使得测量不精确。最后，我们使用洞洞板重新制作了谐振电路，除了电感接口、电容、输入输出接口之外，不外加任何多余的元器件，改良效果显著。

3.2 软件调试问题与解决方案

3.2.1 C2000 片内 RAM 与 FLASH 空间分配问题

调试过程中，我们发现 FFT 是一个时空复杂度较高的计算模块。对序列长度为 4096 点数的采样序列进行 FFT 运算得到幅频信息虽然在程序编译上不报错，但是下板过程中会出现片内 RAM 空间不足的问题。C2000 工程中，“28004x_generic_flash_lnk.cmd”文件用于调节不同页的 RAM 与 FLASH 空间分配。我们将 PAGE 0 中的一部分 RAM 空间搬到 PAGE 1 中，并且在计算幅值过程中不采用 FFT，而是计算均方值的方法，增加了 RAM 空间的利用率，保证了程序计算的稳定性与准确性。

3.2.1.1 C2000 GPIO 翻转速率受限

最初，我们希望通过 GPIO 模拟时钟驱动 AD7606，但测试之后发现 C2000 的 GPIO 翻转速率最高至 600kHz 左右时，波形出现严重失真，无法作为优质的驱动信号。故后续采用硬件 SPI 时钟驱动。

3.3 测试结果

针对系统的表现性能，我们选取了部分电容电感元件进行测量，结果如表3-1、3-2所示。

表 3-1 电容测量结果

容值标定值 (nF)	D 值标定值	容值测量值 (nF)	D 值测量值	容 值 误 差 ($\Delta_C/\%$)	D 值误差 ($\Delta_D/\%$)
4.575	0.3956	4.64 ↑	0.3950 ↓	1.42	0.15
19.327	0.0291	19.56 ↑	0.0285 ↓	1.20	2.06
102.503	0.0476	103.53 ↑	0.0475 ↓	1.00	0.21
1.033	0.2117	1.05 ↑	0.2056 ↓	1.64	2.88
9.6217	0.0127	9.73 ↑	0.0108-	4.86	0.78-14.96 ¹
			0.0136 ¹		

¹ 针对小容值小 D 值元件，虽然其阻抗虚部测量在误差范围之内，但是 D 值表现不稳定，最大误差在 14.96%

表 3-2 电感测量结果

感 值 标 定 值 (μH)	谐 振 频 率 (MHz)	Q 值 标 定 值	感 值 测 量 值	谐 振 频 率 (MHz)	Q 值 测 量 值	感 值 误 差 ($\Delta_Q/\%$)	Q 值 误 差 ($\Delta_L/\%$)
21.51	1.67492	37.9	21.65 ↑	1.667 ↓	37.22 ↓	0.65	1.79
24.05	1.58350	34.6	24.66 ↑	1.562 ↓	35.47 ↑	2.53	2.51
84.20	0.84643	13.2	84.27 ↑	0.8439 ↓	13.02 ↓	0.08	1.36
36.31	1.28902	28.2	36.64 ↑	1.282 ↓	28.19 ↓	0.90	0.03
38.62	1.24980	28.3	38.97 ↑	1.250 ↑	27.24 ↓	0.90	3.74

第四章 总结与展望

2023 电赛结束后，我们重新思考了 C 题的测量方案，参考了国内外多家公司的 LCR 仪表说明书、Q 表说明书，以及部分文献，结合实际测量场景，不断完善测量方案，最终采用了电桥法与谐振法测量元件。电桥法从之前的自动平衡电桥法改良为直接 I-V 法，更换了主控与外部 ADC，依托 AD7606 的高质采集性能，测量性能在绝大多数元件上表现非常良好。相较于之前将难以采样的交流信号转换成直流信号，直接法无疑是最方便、最稳定的方法。电感测量方面，我们在原有的软硬件基础上又进行了多次尝试，在谐振电路的电容选择、电路板设计等进行了优化与结论补充，最终敲定了云母电容作为谐振电容是最佳方案。在第二章中，我们对一些重点模块进行了详细的设计步骤介绍，对软件部分的驱动进行了接口说明、并在附录中给出了完整的程序代码。至于其中的 FFT、AD9959、AD7606 等计算模块与驱动模块，不在本报告的重点讨论范围内。本次工程的所有代码将开源在https://github.com/Xintongkexie/2023_NUEDC_C_Solution中供老师与同学们参考。

本题的一大难点还在于 TI 微控制器的环境搭建、工程配置与驱动移植。需指出，在 TI 官网提供的 C2000 引脚说明的配图中，J8 和 J6 的引脚编号位置绘制错误，在图2-7和2-8中已经更正。读者也可拿手边的 F280049C LaunchPad 与包装盒内的引脚图进行对比。某些芯片 GPIO 支持的功能（例如 ePWM 等）在 LaunchPad 上可能会被覆盖掉，在使用官方开发板时需格外注意。工程配置主要参考 C2000Ware 支持包，在移植 ePWM、SPI 接口时，主要参考了相关的例程。

到此为止，对 C 题的测量方案还有一部分未解之谜，诸如第三章提及的电容测试中电阻切档使得测量不精确问题。如果读者有一些建设性的建议与想法，欢迎与我们交流。

附录 A 串联电路中电容 D 值测量原理分析

待测电容的 D 值如下定义，在电容串联模型中，视为 C_x 和 R_C 有同一电流，电容器的损耗正切角值是其有功功率与无功功率的比值。

$$D = \tan\delta = \frac{P}{Q} = \frac{R_C}{\frac{1}{\omega C_x}} = \omega R_C C_x \quad (\text{a-1})$$

利用串联电路阻抗模型分析法，可得：

$$\begin{aligned} \mathbf{U}_a - \mathbf{U}_b &= \frac{\frac{1}{j\omega C_x} + R_C}{R_S + R_C + \frac{1}{j\omega C_x}} \mathbf{U}_a \\ &= \frac{j\omega C_x R_C + 1}{j\omega(R_C + R_S)C_x + 1} \mathbf{U}_a \end{aligned} \quad (\text{a-2})$$

其中 P 为电容器的有功功率 (W)，Q 为电容器的无功功率 (var)。根据复变函数的相关性质，得到关于 \mathbf{U}_C 的幅值和相位信息：

$$\begin{aligned} |U_C| &= \frac{\sqrt{1 + \omega^2 C_x^2 R_C^2}}{\sqrt{1 + \omega^2 C_x^2 (R_C + R_S)^2}} A \\ \angle \theta_C &= \angle\left(\frac{j\omega C_x R_C + 1}{j\omega(R_C + R_S)C_x + 1}\right) + \angle(A \cos \omega t) \\ &= \angle\left(\frac{(j\omega C_x R_C + 1)[j\omega C_x (R_C + R_S) - 1]}{[j\omega(R_C + R_S)C_x + 1][j\omega(R_C + R_S)C_x - 1]}\right) \\ &= \angle\left(\frac{-\omega^2 C_x^2 R_C (R_C + R_S) - 1 + j(\omega C_x R_S)}{-\omega^2 C_x^2 (R_C + R_S)^2 - 1}\right) \\ &= \angle\left(\frac{\omega^2 C_x^2 R_C (R_C + R_S) + 1}{\omega^2 C_x^2 (R_C + R_S)^2 + 1} + j \frac{-\omega C_x R_S}{\omega^2 C_x^2 (R_C + R_S)^2 + 1}\right) \\ \tan \theta_C &= \frac{\Re(U_C)}{\Im(U_C)} = \frac{-\omega C_x R_S}{\omega^2 C_x^2 R_C (R_C + R_S) + 1} \\ &= \frac{1}{\omega C_x \frac{R_C^2}{R_S} + \omega C_x R_C + \frac{1}{\omega C_x R_S}} \end{aligned} \quad (\text{a-3})$$

同理，可以得到有关 \mathbf{U}_b 的幅值和相位信息：

$$\begin{aligned} |U_b| &= \frac{\omega C_x R_S}{\sqrt{1 + \omega^2 C_x^2 (R_C + R_S)^2}} A \\ \angle \theta_b &= \angle\left(\frac{j\omega C_x R_S}{j\omega(R_C + R_S)C_x + 1}\right) + \angle(A \cos \omega t) \\ &= \angle\left(\frac{j\omega C_x R_S [1 - j\omega C_x (R_C + R_S)]}{[1 + j\omega C_x (R_C + R_S)][1 - j\omega C_x (R_C + R_S)]}\right) \\ &= \angle\left(\frac{\omega^2 C_x^2 (R_C + R_S) R_S}{\omega^2 C_x^2 (R_C + R_S)^2 + 1} + j \frac{\omega C_x R_S}{\omega^2 C_x^2 (R_C + R_S)^2 + 1}\right) \\ \tan \theta_b &= \frac{\Re(U_b)}{\Im(U_b)} = \frac{1}{\omega C_x (R_C + R_S)} \end{aligned} \quad (\text{a-4})$$

通过两处被测电压的相位差，即可得到待测原件的 D 值：

$$\begin{aligned}
 \tan(\theta_C - \theta_b) &= \frac{\tan \theta_C - \tan \theta_b}{1 + \tan \theta_C \tan \theta_b} \\
 &= \frac{\frac{-\omega C_x R_S}{\omega^2 C_x^2 R_C (R_C + R_S)} - \frac{1}{\omega C_x (R_C + R_S)}}{1 + \frac{-\omega C_x R_S}{[\omega^2 C_x^2 R_C (R_C + R_S) + 1] \omega C_x (R_C + R_S)}} \\
 &= \frac{-\omega^2 C_x^2 (R_C + R_S)^2 - 1}{\omega C_x (R_C + R_S) \omega^2 C_x^2 R_C (R_C + R_S) + \omega C_x R_C} \\
 &= \frac{-\omega^2 C_x^2 (R_C + R_S)^2 - 1}{\omega C_x R_C [\omega^2 C_x^2 (R_C + R_S)^2 + 1]} \\
 &= \frac{-1}{\omega C_x R_C} = -\frac{1}{D}
 \end{aligned} \tag{a-5}$$

只要可以用外部 ADC 实现对两路信号进行同步采样，此方法不存在理论误差，只存在不可避免的偶然误差。

附录 B 主程序代码

Listing 附录 B.1 main.c

```

1 #include "device.h"
2 #include "driverlib.h"
3 #include "stdio.h"
4 #include "string.h"
5 #include "math.h"
6 #include "Myprint.h"
7 #include "ad9959.h"
8 #include "ad7606.h"
9 #include "FFTv2.h"
10 /**
11 * main.c
12 */
13
14 #define EPWM1_TIMER_TBPRD 5000U
15 #define EPWM1_MAX_CMPA 2500U
16 #define EPWM1_MIN_CMPA 2500U
17
18 #define EPWM_CMP_UP 1U
19 #define EPWM_CMP_DOWN 0U
20
21 int NPT = 4096;
22
23 typedef struct
24 {
25     uint32_t epwmModule;
26     uint16_t epwmCompADirection;
27     uint16_t epwmTimerIntCount;
28     uint16_t epwmMaxCompA;
29     uint16_t epwmMinCompA;
30 }epwmInformation;
31
32 // Globals to hold the ePWM information used in this example
33 epwmInformation epwm1Info;
34
35 void Init(void);
36 void mySCIACconfig(void); //GPIO 28 29 SCIARX    2.. 115200 % 16% $»
37 __interrupt void sciaRxISR(void);
38 void initEPWM1(void);
39 __interrupt void epwm1ISR(void);
40 __interrupt void gpioInterruptHandler(void);
41 void updateCompare(epwmInformation *epwmInfo);
42 void SPI_init();

```

```

43 void PinMux_init();
44
45 uint16_t msg=0; // `@_` %p %
46 int16_t ad7606_data[8] = { 0 };
47 uint8_t flag = 0;
48 int32_t num = 0;
49
50 int16_t adc_buf1[NPT_max] = { 0 };
51 int16_t adc_buf2[NPT_max] = { 0 };
52 int16_t adc_buf3[200] = { 0 };
53
54 FFTresult ch1, ch2;
55 float32_t phase = 0, dc1 = 0, dc2 = 0, vp1 = 0, vp2 = 0;
56
57 float32_t D = 0, C = 0;
58 float32_t fre = 600000;
59 float32_t dc3 = 0, dc_store = 0, fre_step = 100000;
60 float32_t Q = 0, L = 0;
61
62 float vin = 0;
63
64 float vpp_test = 200;
65
66 char mode = 'C';
67
68 int main(void)
69 {
70     Init();
71
72     ad9959_init();
73     ad7606_init();
74
75     FFT_Init();
76
77     GPIO_setPadConfig(23, GPIO_PIN_TYPE_STD); // Push-pull output or floating input
78     GPIO_setDirectionMode(23, GPIO_DIR_MODE_OUT);
79     GPIO_setPadConfig(14, GPIO_PIN_TYPE_PULLUP); // Push-pull output or floating input
80     GPIO_setDirectionMode(14, GPIO_DIR_MODE_IN);
81
82
83
84     // Disable global interrupts.
85     //»µ ¸ z µl¤
86
87     ad9959_write_phase(AD9959_CHANNEL_2, 0); //0;
88     ad9959_write_frequency(AD9959_CHANNEL_2, 1000); //1kHz
89     ad9959_write_amplitude(AD9959_CHANNEL_2, 512); //1V

```

```

90
91
92     while(1)
93     {
94         if (GPIO_readPin(14) == 0 && mode == 'L') //Lg³ »-
95         {
96             num = 0;
97             mode = 'C';
98
99             print("L.txt=\\"0\\xff\xff\xff", 0);
100            print("Q.txt=\\"0\\xff\xff\xff", 0);
101        }
102        if (GPIO_readPin(14) == 1 && mode == 'C') //Cg³ »-
103        {
104            num = 0;
105            fre = 600000;
106            fre_step = 100000;
107            ad9959_write_phase(AD9959_CHANNEL_3, 0);
108            ad9959_write_frequency(AD9959_CHANNEL_3, fre);
109            ad9959_write_amplitude(AD9959_CHANNEL_3, 512);
110            mode = 'L';
111
112            print("C.txt=\\"0\\xff\xff\xff", 0);
113            print("D.txt=\\"0\\xff\xff\xff", 0);
114        }
115        if (mode == 'C')
116        {
117            if (num == NPT)
118            {
119                //stop pwm
120                EPWM_setCounterCompareValue(epwm1Info.epwmModule, EPWM_COUNTER_COMPARE_A, 0);
121                // PWM£¬ AD7606²
122                int j = 0; //CCS³ »·±-£¬²»    for(int j = 0; j < NPT; j++)
123
124                DEVICE_DELAY_US(1);
125
126                FFT_int16(adc_buf1, NPT, &ch1); // p £ºUa-Ub£¬
127                FFT_int16(adc_buf2, NPT, &ch2); // p6 £ºUb
128                int x1 = 1, x2 = 1;
129                float32_t max1 = Complex_Modular(ch1.result[1]) / NPT * 2, max2 =
130                    Complex_Modular(ch2.result[1]) / NPT * 2; //
131                for (j = 2; j < NPT / 2; j++)
132                {
133                    if (Complex_Modular(ch1.result[j]) / NPT * 2 > max1)
134                    {
135                        x1 = j;

```

```

135         max1 = Complex_Modular(ch1.result[j]) / NPT * 2;
136     }
137     if (Complex_Modular(ch2.result[j]) / NPT * 2 > max2)
138     {
139         x2 = j;
140         max2 = Complex_Modular(ch2.result[j]) / NPT * 2;
141     }
142 }
143
144 phase = (Complex_arg(ch1.result[x1]) - Complex_arg(ch2.result[x2])) / pi *
145     180; //^2
146 if (phase > 180)
147 {
148     phase -= 360;
149 }
150 if (phase < -180)
151 {
152     phase += 360;
153 }
154 float32_t sum1 = 0, sum2 = 0;
155 for (j = 0; j < NPT; j++) //%
156 {
157     sum1 += (adc_buf1[j] * 5.0f / 32767);
158     sum2 += (adc_buf2[j] * 5.0f / 32767);
159 }
160 dc1 = sum1 / NPT;
161 dc2 = sum2 / NPT;
162
163 float32_t rms1 = 0, rms2 = 0;
164 for (j = 0; j < NPT; j++) //%
165 {
166     rms1 += pow(adc_buf1[j] * 5.0f / 32767 - dc1, 2);
167     rms2 += pow(adc_buf2[j] * 5.0f / 32767 - dc2, 2);
168 }
169 vp1 = sqrt(rms1 / NPT) * sqrt(2);
170 vp2 = sqrt(rms2 / NPT) * sqrt(2);
171 // print("phase: {.2}, ", phase);
172 D = tanf((phase - 90) / 180 * pi); //% D
173 print("D.txt=\\"{.4}\\"xff\xff\xff\xff", D);
174 C = 1000000.0f / (2 * pi * 159.5 * cosf((phase - 90) / 180 * pi)) * (vp2 / vp1
175 ); //% C
176 print("C.txt=\\"{.2}nF\\xff\xff\xff\xff", C);
177 ad7606_reset(); //, AD7606
178 num = 0;
179 //start pwm

```

```

180         EPWM_setCounterCompareValue(epwm1Info.epwmModule, EPWM_COUNTER_COMPARE_A,
181                                     EPWM1_MAX_CMPA); //AD7606^
182     }
183     if (mode == 'L')
184     {
185         if (num == 200)
186         {
187             EPWM_setCounterCompareValue(epwm1Info.epwmModule, EPWM_COUNTER_COMPARE_A, 0);
188             // PWM^ AD7606^
189             DEVICE_DELAY_US(1);
190
191             float32_t sum = 0;
192             int j = 0;
193             for (j = 0; j < 200; j++)
194             {
195                 sum += (adc_buf3[j] * 5.0f / 32767);
196             }
197             dc3 = sum / 200; //AD8307^ -
198             if (dc3 < dc_store)
199             {
200                 fre -= (2 * fre_step);
201                 if (fre_step > 1000)
202                 {
203                     fre_step /= 10;
204                 }
205                 else//μHz . 1000Hz
206                 {
207                     L = 1.0f / (pow(2 * pi * fre, 2) * 420 * pow(10, -18));
208                     vin = 2.573 * pow(10, -5) * pow(2.718, 4.69 * dc3);
209                     Q = vin / 3.95 / 0.0036 * 1.1;
210                     print("L.txt=%.{2}uF\"\\xff\\xff\\xff", L);
211                     print("Q.txt=%.{2}\"\\xff\\xff\\xff", Q);
212                     fre = 600000;
213                     fre_step = 100000;
214                 }
215                 dc_store = 0;
216             }
217             else
218             {
219                 dc_store = dc3;
220                 fre += fre_step;
221             }
222             ad9959_write_frequency(AD9959_CHANNEL_3, fre);
223             ad7606_reset();
224             DEVICE_DELAY_US(10000);

```

```

225         num = 0;
226         EPWM_setCounterCompareValue(epwm1Info.epwmModule, EPWM_COUNTER_COMPARE_A,
227                                     EPWM1_MAX_CMPA);
228     }
229 }
230
231 }
232
233 void Init(void)
234 {
235     Device_init();
236     Device_initGPIO();
237     Interrupt_initModule();
238     Interrupt_initVectorTable();
239
240     GPIO_setPadConfig(34, GPIO_PIN_TYPE_PULLUP); // Push-pull output or floating input
241     GPIO_setDirectionMode(34, GPIO_DIR_MODE_IN);
242
243     DINT;
244
245     // Initialize interrupt controller and vector table.
246     Interrupt_initModule();
247     Interrupt_initVectorTable();
248     IER = 0x0000;
249     IFR = 0x0000;
250     Interrupt_register(INT_EPWM1, &epwm1ISR);
251
252     // GPIO0/1 I ePWM1A/1B
253     GPIO_setPadConfig(0, GPIO_PIN_TYPE_STD);
254     GPIO_setPinConfig(GPIO_0_EPWM1A);
255
256     // 1 % Disable sync(Freeze clock to PWM as well)
257     SysCtl_disablePeripheral(SYSCTL_PERIPH_CLK_TBCLKSYNC);
258
259     initEPWM1();
260     // 2 % Enable sync and clock to PWM
261     SysCtl_enablePeripheral(SYSCTL_PERIPH_CLK_TBCLKSYNC);
262
263     //
264     Interrupt_enable(INT_EPWM1);
265
266     mySCIAconfig();
267
268     GPIO_setInterruptType(GPIO_INT_XINT1, GPIO_INT_TYPE_FALLING_EDGE);
269     GPIO_setInterruptPin(34, GPIO_INT_XINT1);
270     GPIO_enableInterrupt(GPIO_INT_XINT1);

```

```

271     EALLOW;
272     PinMux_init();
273     SPI_init();
274     EDIS;
275
276     Interrupt_register(INT_XINT1, &gpioInterruptHandler);
277     Interrupt_enable(INT_XINT1);
278     // Enable global interrupts.
279
280     EINT;
281     ERTM;
282 }
283
284 void mySCIAconfig(void)
285 {
286     /*
287     SCIA    ú~  &GPIO28IRx&~GPIO29ITx
288         2~  115200~  [E]    SCIA_RX
289     */
290     // GPIO28 is the SCI Rx pin.
291     GPIO_setMasterCore(DEVICE_GPIO_PIN_SCIRXDA, GPIO_CORE_CPU1);
292     GPIO_setPinConfig(DEVICE_GPIO_CFG_SCIRXDA);
293     GPIO_setDirectionMode(DEVICE_GPIO_PIN_SCIRXDA, GPIO_DIR_MODE_IN);
294     GPIO_setPadConfig(DEVICE_GPIO_PIN_SCIRXDA, GPIO_PIN_TYPE_STD);
295     GPIO_setQualificationMode(DEVICE_GPIO_PIN_SCIRXDA, GPIO_QUAL_ASYNC);
296
297     // GPIO29 is the SCI Tx pin.
298     GPIO_setMasterCore(DEVICE_GPIO_PIN_SCITXDA, GPIO_CORE_CPU1);
299     GPIO_setPinConfig(DEVICE_GPIO_CFG_SCITXDA);
300     GPIO_setDirectionMode(DEVICE_GPIO_PIN_SCITXDA, GPIO_DIR_MODE_OUT);
301     GPIO_setPadConfig(DEVICE_GPIO_PIN_SCITXDA, GPIO_PIN_TYPE_STD);
302     GPIO_setQualificationMode(DEVICE_GPIO_PIN_SCITXDA, GPIO_QUAL_ASYNC);
303
304     // Map the ISR to the wake interrupt.
305     Interrupt_register(INT_SCIA_RX, sciaRxISR);
306
307     // Initialize SCIA and its FIFO.
308     SCI_performSoftwareReset(SCIA_BASE);
309
310     // 2~  115200 8  %  1
311     SCI_setConfig(SCIA_BASE, 25000000, 115200, (SCI_CONFIG_WLEN_8 |
312             SCI_CONFIG_STOP_ONE |
313             SCI_CONFIG_PAR_NONE));
314
315     SCI_resetChannels(SCIA_BASE);
316     SCI_clearInterruptStatus(SCIA_BASE, SCI_INT_RXRDY_BRKDT);
317     SCI_enableModule(SCIA_BASE);

```

```

318     SCI_performSoftwareReset(SCIA_BASE);
319
320     // Enable the TXRDY and RXRDY interrupts.
321     SCI_enableInterrupt(SCIA_BASE, SCI_INT_RXRDY_BRKDT);
322
323     // Clear the SCI interrupts before enabling them.
324     SCI_clearInterruptStatus(SCIA_BASE, SCI_INT_RXRDY_BRKDT);
325
326     // Enable the interrupts in the PIE: Group 9 interrupts 1 & 2.
327     Interrupt_enable(INT_SCIA_RX);
328     Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP9);
329 }
330
331
332 // sciaRxISR - Read the character from the RXBUF and echo it back.
333 __interrupt void sciaRxISR(void)
334 {
335     uint16_t receivedChar;
336
337     // Read a character from the RXBUF.
338     receivedChar = SCI_readCharBlockingNonFIFO(SCIA_BASE);
339     msg=receivedChar; // , 0% -
340
341     // Echo back the character.
342     SCI_writeCharBlockingNonFIFO(SCIA_BASE, receivedChar);
343
344     // Acknowledge the PIE interrupt.
345     Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP9);
346 }
347
348 __interrupt void epwm1ISR(void) //
349 {
350     // , ± Update the CMPA and CMPB values
351     //updateCompare(&epwm1Info);
352
353     // Clear INT flag for this timer
354     EPWM_clearEventTriggerInterruptFlag(EPWM1_BASE);
355
356     // Acknowledge interrupt group
357     Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP3);
358 }
359
360 __interrupt void gpioInterruptHandler(void)
361 {
362     int i = 0;
363
364     for (i = 0; i < 8; i++)

```

```

365     {
366         SPI_writeDataNonBlocking(SPIB_BASE, 0x00);
367         ad7606_data[i] = (int16_t)SPI_readDataNonBlocking(SPIB_BASE);
368     }
369
370     if (mode == 'C')
371     {
372         if (num < NPT)
373         {
374             adc_buf1[num] = ad7606_data[0];
375             adc_buf2[num++] = ad7606_data[1];
376         }
377     }
378     if (mode == 'L')
379     {
380         if (num < 200)
381         {
382             adc_buf3[num++] = ad7606_data[2];
383         }
384     }
385
386     Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP1);
387 }
388
389 // ^-- initEPWM1 - Configure ePWM1
390
391 void initEPWM1()
392 {
393     // ePWM1 TBCLK 100MHzµ  £¬ÿ , TBCLK¶  10ns
394     EPWM_setTimeBasePeriod(EPWM1_BASE, EPWM1_TIMER_TBPRD);
395     EPWM_setPhaseShift(EPWM1_BASE, OU);
396     EPWM_setTimeBaseCounter(EPWM1_BASE, OU);
397
398     // CMPA° CMPB
399     EPWM_setCounterCompareValue(EPWM1_BASE,
400                                 EPWM_COUNTER_COMPARE_A,
401                                 EPWM1_MIN_CMPA);
402
403     // I  £¬  PWM1      10ns*2000*2=40us£¬¶  25kHz
404     EPWM_setTimeBaseCounterMode(EPWM1_BASE, EPWM_COUNTER_MODE_UP_DOWN);
405     EPWM_disablePhaseShiftLoad(EPWM1_BASE);
406     EPWM_setClockPrescaler(EPWM1_BASE,
407                           EPWM_CLOCK_DIVIDER_1,
408                           EPWM_HSCLOCK_DIVIDER_1);
409
410     // J      IO
411     EPWM_setCounterCompareShadowLoadMode(EPWM1_BASE,

```

```

412                                     EPWM_COUNTER_COMPARE_A,
413                                     EPWM_COMP_LOAD_ON_CNTR_ZERO);
414
415 //      ePWM¶-
416 // :¶  EPWM1A,µ±    ==CMPA     ,µ±¶    ==CMPA
417 EPWM_setActionQualifierAction(EPWM1_BASE,
418                             EPWM_AQ_OUTPUT_A,
419                             EPWM_AQ_OUTPUT_HIGH,
420                             EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
421 EPWM_setActionQualifierAction(EPWM1_BASE,
422                             EPWM_AQ_OUTPUT_A,
423                             EPWM_AQ_OUTPUT_LOW,
424                             EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
425
426 //      ®·¢      ,      I¥      IO ¢·¢;£
427 //      £
428 //      ý, 3¢¥·¢'    -    15¢P£
429 EPWM_setInterruptSource(EPWM1_BASE, EPWM_INT_TBCTR_ZERO);
430 EPWM_enableInterrupt(EPWM1_BASE);
431 EPWM_setInterruptEventCount(EPWM1_BASE, 3U);
432
433 //
434 // Information this example uses to keep track of the direction the
435 // CMPA/CMPB values are moving, the min and max allowed values and
436 // a pointer to the correct ePWM registers
437 //
438 epwm1Info.epwmCompADirection = EPWM_CMP_UP;
439 epwm1Info.epwmTimerIntCount = 0U;
440 epwm1Info.epwmModule = EPWM1_BASE;
441 epwm1Info.epwmMaxCompA = EPWM1_MAX_CMPA;
442 epwm1Info.epwmMinCompA = EPWM1_MIN_CMPA;
443 }
444
445 // , +
446 void updateCompare(epwmInformation *epwmInfo)
447 {
448     uint16_t compAValue;
449
450     compAValue = EPWM_getCounterCompareValue(epwmInfo->epwmModule,
451                                             EPWM_COUNTER_COMPARE_A);
452
453 //
454 // Change the CMPA/CMPB values every 10th interrupt.
455 //
456 if(epwmInfo->epwmTimerIntCount == 10U)
457 {
458     epwmInfo->epwmTimerIntCount = 0U;

```

```

459
460    //
461    // If we were increasing CMPA, check to see if we reached the max
462    // value. If not, increase CMPA else, change directions and decrease
463    // CMPA
464    //
465    if(epwmInfo->epwmCompADirection == EPWM_CMP_UP)
466    {
467        if(compAValue < (epwmInfo->epwmMaxCompA))
468        {
469            EPWM_setCounterCompareValue(epwmInfo->epwmModule,
470                                         EPWM_COUNTER_COMPARE_A,
471                                         ++compAValue);
472        }
473        else
474        {
475            epwmInfo->epwmCompADirection = EPWM_CMP_DOWN;
476            EPWM_setCounterCompareValue(epwmInfo->epwmModule,
477                                         EPWM_COUNTER_COMPARE_A,
478                                         --compAValue);
479        }
480    }
481    //
482    // If we were decreasing CMPA, check to see if we reached the min
483    // value. If not, decrease CMPA else, change directions and increase
484    // CMPA
485    //
486    else
487    {
488        if( compAValue == (epwmInfo->epwmMinCompA))
489        {
490            epwmInfo->epwmCompADirection = EPWM_CMP_UP;
491            EPWM_setCounterCompareValue(epwmInfo->epwmModule,
492                                         EPWM_COUNTER_COMPARE_A,
493                                         ++compAValue);
494        }
495        else
496        {
497            EPWM_setCounterCompareValue(epwmInfo->epwmModule,
498                                         EPWM_COUNTER_COMPARE_A,
499                                         --compAValue);
500        }
501    }
502
503    //
504    // If we were increasing CMPB, check to see if we reached the max
505    // value. If not, increase CMPB else, change directions and decrease

```

```
506     // CMPB
507     //
508 }
509 else
510 {
511     epwmInfo->epwmTimerIntCount++;
512 }
513 }
514 }
515
516 void PinMux_init()
517 {
518     //
519     // SPIB -> SPIB_master Pinmux
520     //
521     GPIO_setPinConfig(GPIO_24_SPIB_SIMO);
522     GPIO_setPinConfig(GPIO_31_SPIB_SOMI);
523     GPIO_setPinConfig(GPIO_22_SPIB_CLK);
524     GPIO_setPinConfig(GPIO_27_SPIB_STE);
525
526 }
527
528 void SPI_init()
529 {
530     //SPIB_master initialization
531     SPI_disableModule(SPIB_BASE);
532     SPI_setConfig(SPIB_BASE, DEVICE_LSPCLK_FREQ, SPI_PROT_POL1PHA1,
533                   SPI_MODE_MASTER, 2000000, 16);
534     SPI_disableLoopback(SPIB_BASE);
535     SPI_enableFIFO(SPIB_BASE);
536     SPI_enableModule(SPIB_BASE);
537 }
```