# 2.综合设计-交流电表

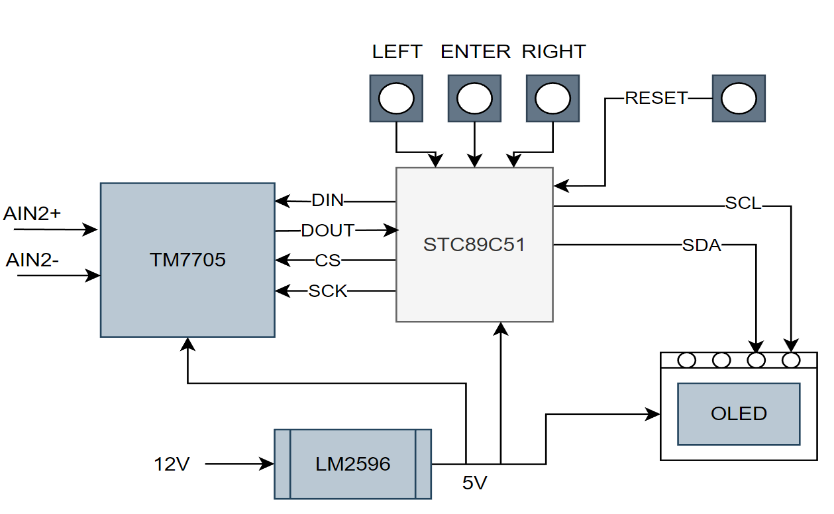
**《A5.数字交流电压表的设计》**

以MCS-51单片机为核心，设计一个交流数字电压表，交流信号由变压器提供，二次侧0-5V AC，匝数比200:1。测量值显示在LCD显示器上，被测电压由8位AD转换器将模拟量转换成数字量信号有：启动转换信号（START,输出），转换结束(EOC, 输入)和读控制信号（OE，低电平有效）；另外，可以选择测量量程，例如通过单片机输出一个3位二进制数来选择不同的前置放大器增益。

**综合课题的分析设计：**

**硬件方面：**主控采用宏晶科技的STC89C52RC,使用面包板搭建51的最小系统。该系统包含晶振电路、按键输入电路、电阻分压电路以及若干外部模块接口。使用的模块有：一块0.95寸IIC驱动的OLED屏幕，双路16bit高精度ADC转换模块TM7705，DC-DC降压稳压模块LM2596。

1. **OLED屏幕：**采用软件模拟IIC，故只需要从单片机引出两条普通的GPIO口控制其SDA和SCL引脚进行通信，单片机主要对其进行写指令和写数据操作，内部自带RAM保存显示数据，5V供电显示更稳定。
2. **TM7705模块：**TM7705为AD7705仿真的国产高精度Σ-Δ型ADC芯片，其功能完全兼容ADC程序，具备双通道差分输入，可编程PGA增益，以及高达500Hz的转换频率，线性度佳。主要采用其通道二引脚（AIN2+、AIN2-），5V供电，模块内部**自带2.5V的稳压源**作为其ADC参考电压。模块采用SPI进行通信，DIN、CS、DOUT、SCK引脚接入单片机GPIO进行软件模拟SPI通信，该模块另有DRDY用作忙信号状态输出，RST用作模块复位。
3. **稳压模块LM2596：**该模块用于将三节18650电池串联成的12V电压降压到5V输出给单片机系统以及模块。DC-DC降压，效率高，波纹较小，最大输出电流为3A，适合小型设备的稳压供电。

整体的硬件电路框架如下图所示：

**软件方面：**主要分为OLED的IIC通信驱动函数以及相关画图函数、TM7705的SPI驱动函数以及RMS计算函数、按键处理逻辑及菜单UI逻辑框架。

**OLED方面：**

* **IIC驱动函数：**使用软件IIC对P20和P21两个引脚进行控制，遵循IIC协议进行编写，用到的驱动函数有i2c\_start、i2c\_stop、i2c\_SendByte、i2c\_WaitAck，不涉及读操作，对OLED只写不读。
* **画图函数：**由于89C52的内部RAM空间仅有512bytes不足以开辟一个大数组来作为显示缓冲区，故画图函数**不能精确到一个像素点进行绘制**，只能按照八位的数据进行实时绘制。0.95寸OLED尺寸为64\*128，实际上的行数为8行，这是因为OLED每帧数据都是一个字节，屏幕的短边被分为8个字节的长度。主要用到了OLED\_ShowStr、OLED\_DrawBMP、OLED\_Fill这三个画图函数，分别对应于显示字符串、显示BMP图片、填充屏幕的画图功能。

**TM7705方面：**

* **SPI驱动函数：**使用软件IO口对SPI协议进行模拟，主要用到TM7705\_Send8Bit、TM7705\_Recive8Bit、TM7705\_WaitDRDY、TM7705\_ResetHard这四个函数，分别对应于发送一个字节、接收一个字节、等待忙信号、硬件复位。由于需要通过SPI对TM7705进行参数配置，每一次对TM7705内部寄存器进行读写操作必须要对其通信寄存器进行一次写入操作，读数据需要从其内部数据寄存器读两个字节（16位），故SPI对其既读又写。
* **RMS计算公式：**从TM7705采样读出来的ADC原始值需要经过一系列转换才能得到实际电压值，对于单极性工作模式、2.5V为参考电压的转换公式如下：



其中Gain为前端增益，默认为1，最大可达128，可以通过设置寄存器对其进行更改。

交流电压通常采用均方根的方式求取其有效值，其基本公式如下：



对于带有直流偏置的交流电压有效值计算，则需要通过采样交流电压平均值，即为直流分量的电压幅值记为“Average”，然后在采用以下的通用公式进行计算：

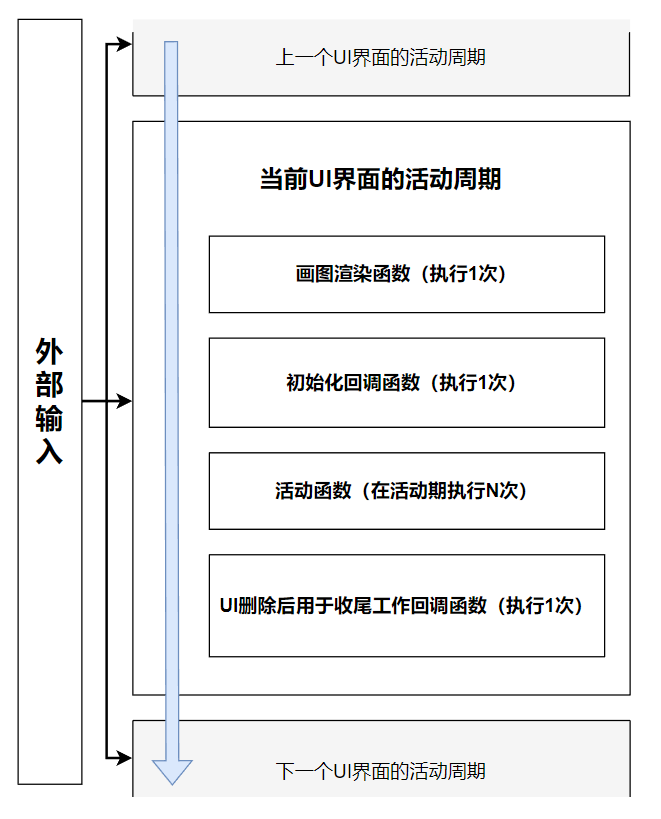


**按键处理逻辑及菜单UI逻辑框架：**

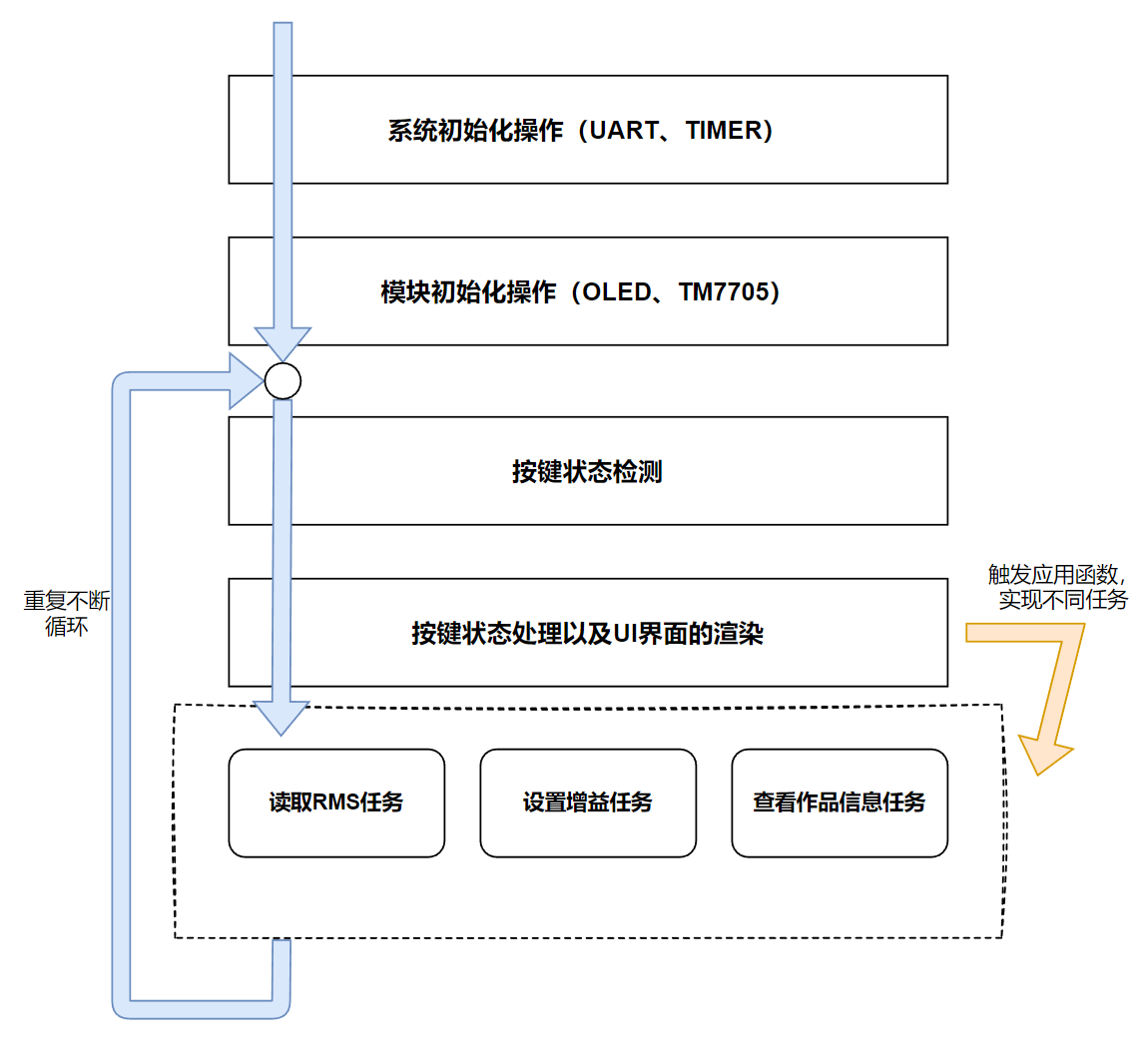
借鉴了安卓系统的活动运作机制以及LVGL小型嵌入式开源图形库的原理，进行了重头到尾的新的设计，裁剪了一些复杂无用的功能（比如事件机制、样式渲染机制、多显示设备驱动），**使总体核心代码限制在300行以内**，对于51的程序存储空间很友好。采用链表结构使得菜单便于拓展、移植。主要分为两个核心过程：外部设备信息采集（按键信息）、按键状态响应及界面切换渲染。外部按键有三个，分别为KEY\_LEFT、KEY\_ENTER、KEY\_RIGHT，分别为“向前键”、“确认键”、“向后键”。按键状态总共有四种，分别为NONE、CLICKED、LONG\_PRESSED、PRESSING，分别为“无操作”、“短按点击”、“长按”、“正被按下”。其中**只有确认键有长按的状态**，长按的功能用于退出菜单的某一级界面。当按键状态被采集，紧接着是按键状态逻辑的判断，向前、向后键用于改变菜单游标指针的位置，指向当前界面（Page）中的每个元素（Item），短按确认键则触发目标Item的回调函数，并在每个UI时间周期重复执行该回调函数直至长按确认键返回到上一级界面。

**UI的渲染方面，每一次界面的切换都会调用“pageJumper”函数，均会调用相关OLED显示函数进行页面的刷新。每一次渲染新的UI的初始化操作以及清除UI时的收尾工作都会有一个定制的回调函数被执行。**

整个UI系统框架的运行机制大致可以用下图表示：

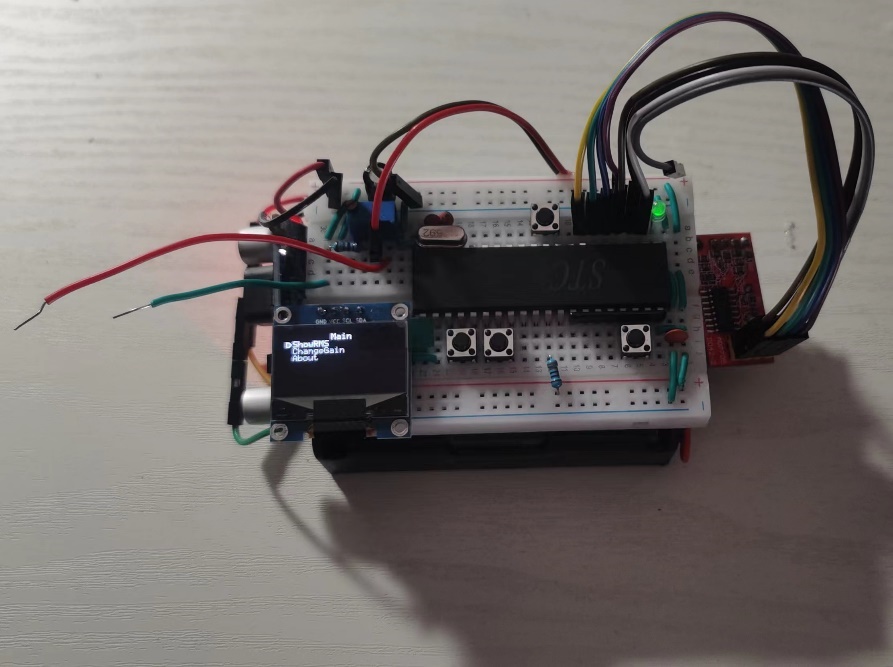


综上，整体的软件流程可以用下面流程图来表示：



**综合课题的实现：**

1. **实物开发板：**本次综合设计的单片机系统由一块面包板搭建而成，以下是实物的全貌：结构比较紧凑，有三个输入按键（底下的三个），一个复位按键（最上面的），最左边附着的是LM2596稳压模块，面包板底下是三节18650构成的串联电池组，左面另一个模块是OLED屏幕，最右边的带有一串杜邦线的模块为TM7705。



1. **实际运作调试：**采用数字信号发生器产生的20Hz的正弦波作为外部输入信号，分别在以下四个测试条件下进行测试：
   1. 幅度5.0Vpp，偏移2.5V
   2. 幅度2.0Vpp，偏移2.5V
   3. 幅度100mVrms，偏移2.1V
   4. 幅度3.0mVrms，偏移2.1V

其中（a）、（b）在1倍增益下进行测量；（c）、（d）需要调节外部增益在2倍增益下进行。调节增益只需要在主界面选择ChangeGain子项然后按左右键调节增益，确定好增益后长按确认键使增益生效，操作界面如下图所示：

将数字电源的正负级接入两根引出的正负信号导线，在主界面选择ShowRMS进入交流有效电压的测量UI界面，界面中显示了直流偏置量（DC）和交流的RMS值，四次测量结果及实验现象如下所示：

A）幅度5.0Vpp，偏移2.5V

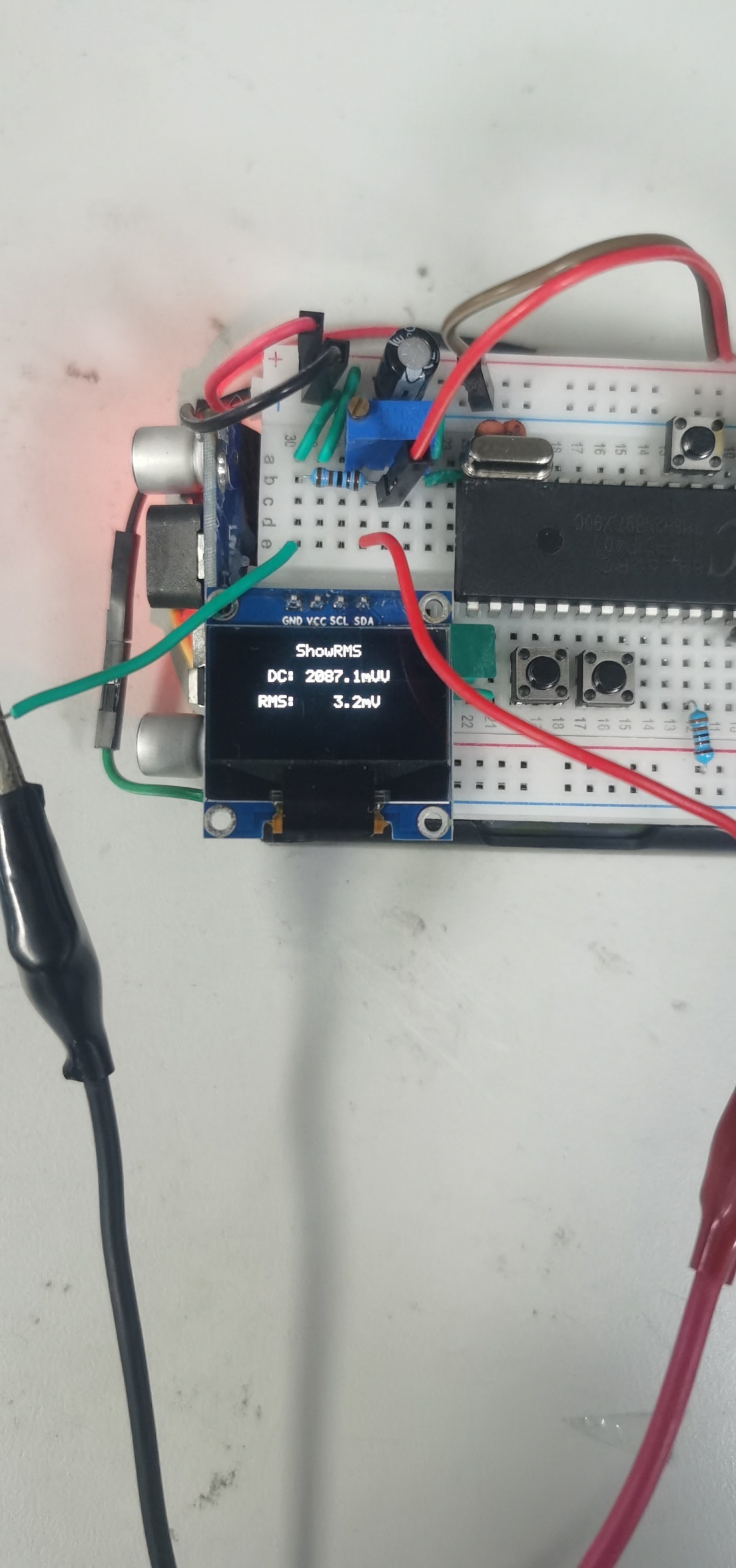
B）幅度2.0Vpp，偏移2.5V

C）幅度100mVrms，偏移2.1V

D）幅度3.0mVrms，偏移2.1V

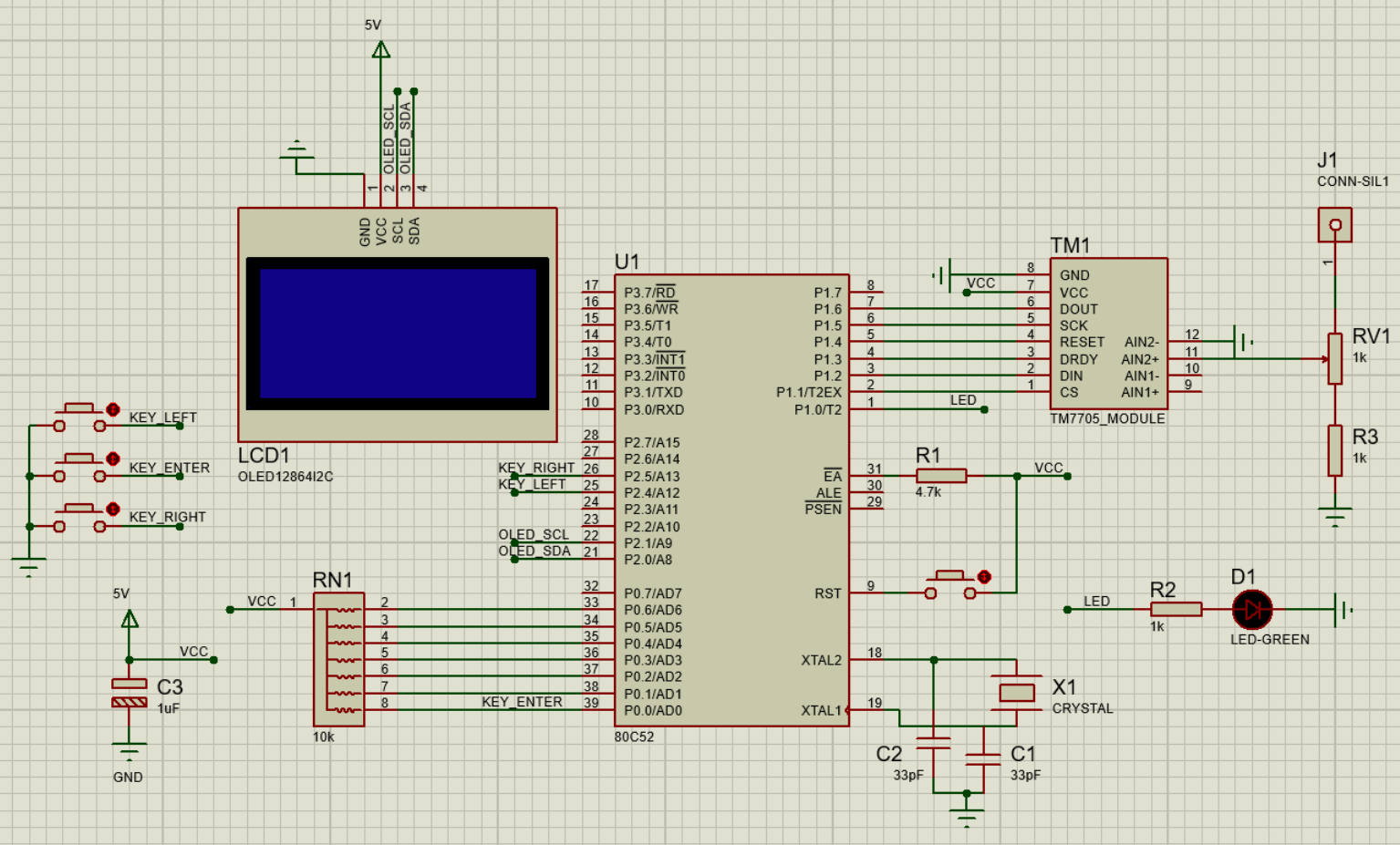
 

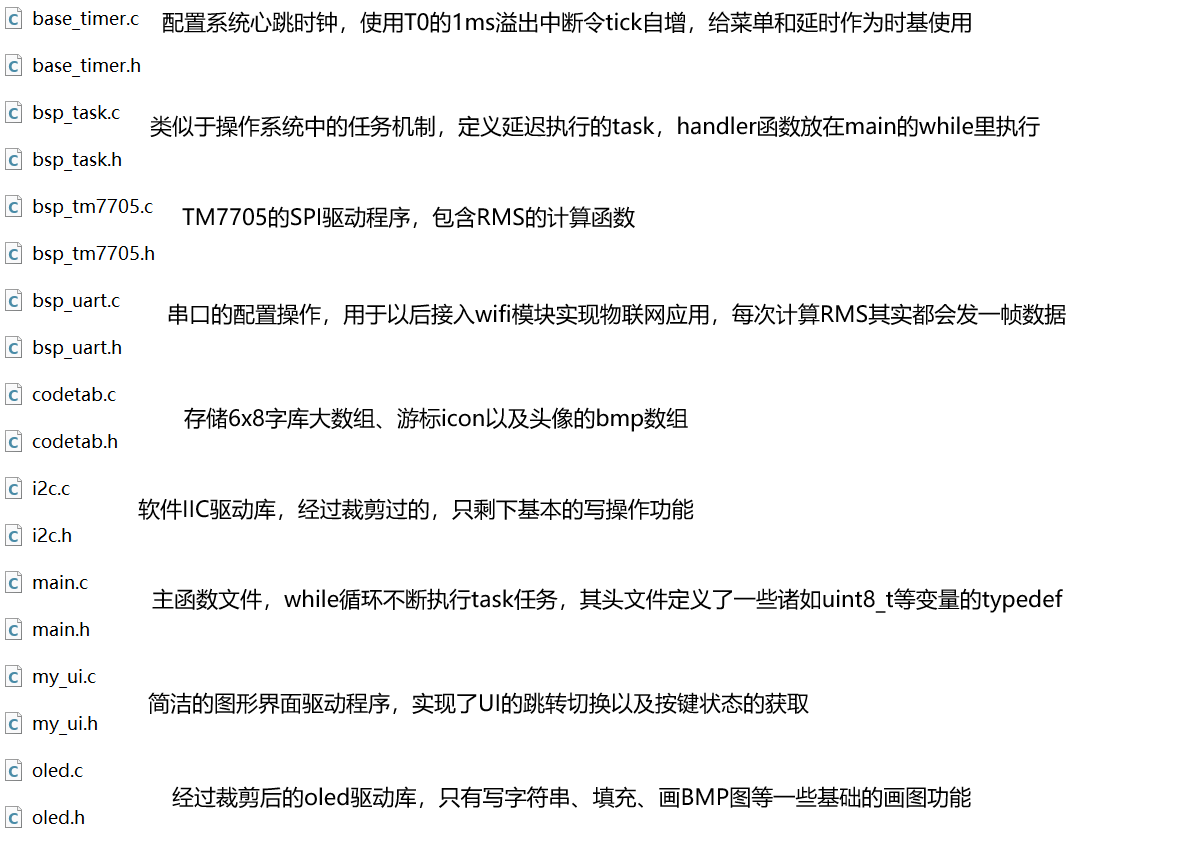
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 实际偏置（mV） | 实际RMS（mV） | 测试偏置(mV) | 测试RMS（mV） | 偏置误差% | RMS误差% |
| a | 2500.0 | 1767.8 | 2508.4 | 1715.7 | 0.3 | 2.9 |
| b | 2500.0 | 707.1 | 2495.5 | 694.0 | 0.18 | 1.85 |
| c | 2100.0 | 100.0 | 2061.8 | 99.1 | 1.8 | 0.9 |
| d | 2100.0 | 3.0 | 2087.1 | 3.2 | 0.61 | 6.7 |

如表格中所示，偏置误差都非常小，**大致在0.2%左右，RMS误差在3%左右**，越小的RMS精度相对会低一些。测量的精度还与单片机的采样速率有关，符合香农定律的前提下，采样速率越高越精确，实际测试单片机的采样速率大致**在120Hz左右**，算是比较低的，**这是因为TM7705自身的转换速率就由500Hz了，限于52的11.0592MHz主频无法发挥其最大的采样效力。**

1. **Proteus仿真**

使用Proteus8.16绘制系统的原理图并进行仿真，绘制系统原理图如下，其中，在原理图中最右边由RV1和R3构成的电阻分压网络，目的是将外部输入正电压分压成原先的一半，以使最终输入模块的电压范围从0-5V限制到0-2.5V之间，使得外部输入TM7705电压低于参考电压2.5，这样才能有效测量。由于库中缺少TM7705期间故仿真也就是在OLED简单地画图，但通过仿真原理图便于查找接线错误以及优化面包板上地接线布局，且验证了菜单的一些显示逻辑。



1. **程序文件清单及描述**
2. **主要的源代码及注释**

整个项目最关键的源代码当属UI菜单框架部分，其次是RMS计算函数。

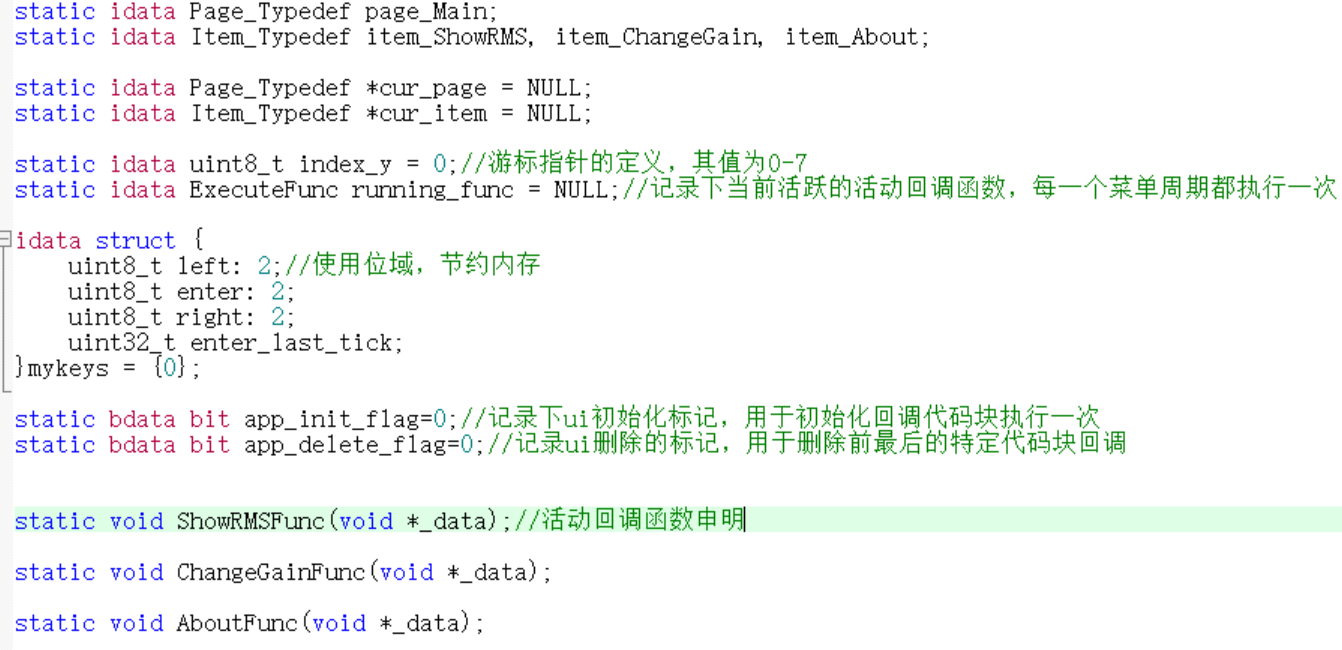
1. UI菜单框架：

my\_ui.h文件下定义了主要的结构体以及函数申明：

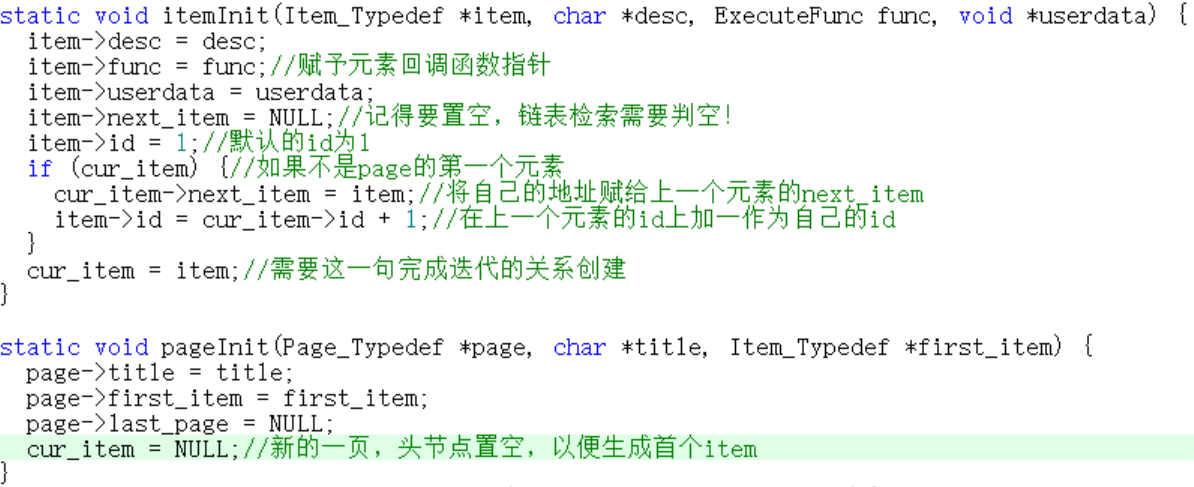


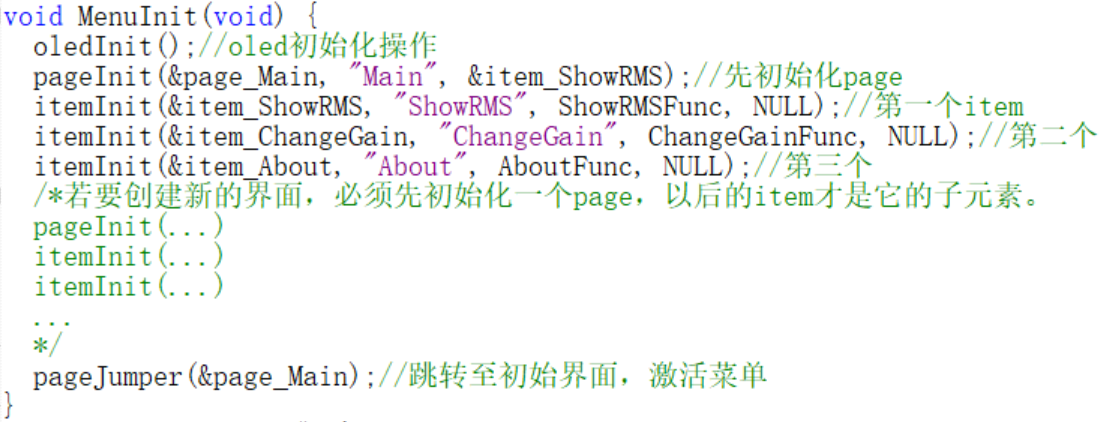
接着是my\_ui.c源文件中的相关函数：

首先是全局变量的定义：



创建一个菜单需要先通过pageInit初始化一个page页面，才可以用ItemInit来初始化其Item。





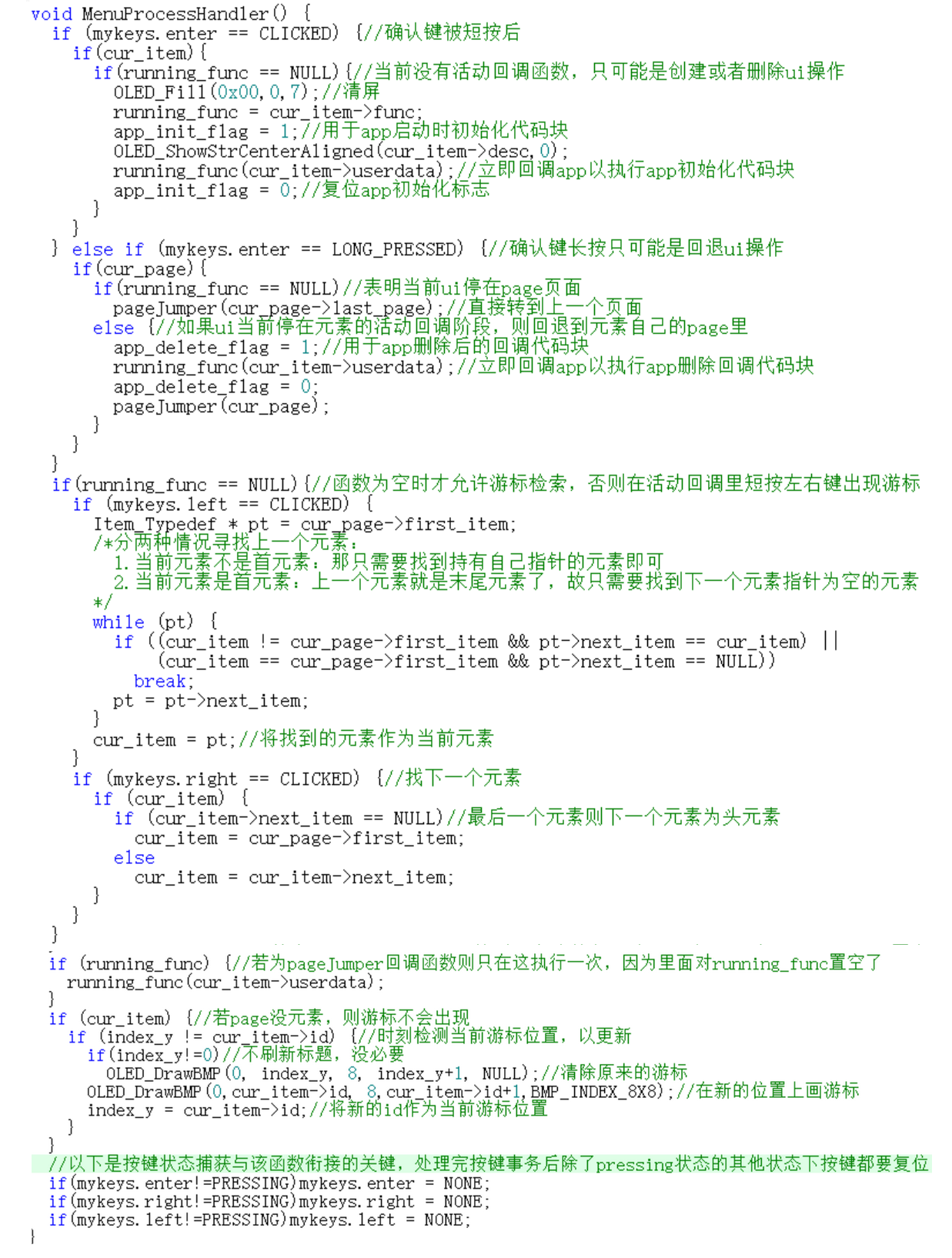
其中pageJumper是通用的页面跳转函数，其既可以直接调用也可以作为item的活动回调函数被调用，实现两个页面至多个之间的跳转！其函数原型如下：



接下来是按键状态的捕获函数MenuKeyHandler()的定义，它是每个ui周期的第一项执行操作。



接下来是UI框架最重要的函数MenuProcessHandler，该函数处理了按键状态信息，并渲染了UI画面，其完成了页面的跳转，游标的切换，活动回调函数的执行这些主要的功能：其函数原型如下：



1. RMS计算函数：先进行直流量的采集（平均值），然后作为偏置量在之后的RMS计算中的平方项中减去。具体的数学计算公式见前面软件原理章节的介绍。



# 3.实践总结

经历了四天的52单片机集中开发的日子，期间碰到了几点比较大的问题：

1. TM7705模块通道一采集过程中会卡死，adc值固定不变，最后是换成了通道2才解决，估计是模块本身的问题，具体不明。
2. 52单片机内存不够用，首先是RAM用爆了，后来发现是存在许多未用的变量但编译器没有帮我优化掉，通过修改keil的编译等级自动忽略变量最终大大减少了RAM的使用量。其次是随着程序的不断增加ROM也用爆了，经过一顿排查，发现许多函数并没用上、8x16的大字库文件占用过大、标准库的vsprintf很耗ROM，于是通过不断地进行删减、合并函数，最后把ROM整整8KB全部用完，刚好不剩，且完成课程的设计。
3. 使用格式化字符串函数时，对于整型变量的格式代码不是单纯的“%u”和“%d”，C51的定义很特殊，例如用printf输出时1byte、2byte、4byte有符号整型数据分别用“%bd”、“%hd”、“%ld”表示，对应的无符号数分别用“%bu”、“%hu”、“%lu”

这些问题都是我先前大一时入门51时没有关注过的问题，于这段时间一一拾起。稍微有点遗憾的课设的时间可能有些仓促，一周不到的时间。如果再多一些时间，也许我会加上外部ROM拓展程序的空间，写上物联网的相关数据处理接口，接入云平台。而目前该部分只是在每次计算完RMS后用串口输出一帧数据而已。不过比较满意的一点是，我用面包板和单片机裸片搭了一个最小系统，紧凑的电路设计让我一时充满成就感！