目 录

1. 概述 1

1.1 实验目的 1

1.2 实验主要器材和设备 1

1.2.1 嵌入式系统基本开发环境 1

1.2.2 仪器工具 1

1.2.3 主要元件类器材 1

2. 部分硬件设计与实现 2

2.1 基准电压发生电路的设计与实现 2

2.2 可调分压和电压跟随电路的实现 2

2.3 温度传感器电路的实现 3

2.4 频率信号整形调理电路的实现 4

2.5 发射机各硬件功能模块的连接方式 4

2.6 接收机各硬件功能模块的连接方式 5

3. 算法及系统基本功能的软件设计与实现 6

3.1 温度-频率编码关系的设计和实现 6

3.2 发射端系统整合的软件和界面设计 6

3.3 接收端系统整合的软件和界面设计 8

4. 进阶功能的技术方案设计与实现 9

4.1 自播音频以及方式一接收并播放音频功能设计 9

4.2 方式二接收并播放音频以及录音功能设计 9

4.3 系统整合设计 10

5. 附录A 学习心得和意见建议 11

5.1 学习感受与心得 11

5.2 对本课程的一点小建议 11

6. 附录B 核心代码清单 12

6.1 发射端程序核心代码清单 12

6.2 接收端程序核心代码清单 35

# 概述

## 实验目的

初步掌握电子测量设计中关于标定校准的基本实验技能；

通过综合性实验获得关于设计开发较复杂工程电子系统的初步经验；

融入团队，扮演角色，承担职责，获得关于专业团队合作的直接经验。

## 实验主要器材和设备

### 嵌入式系统基本开发环境

电脑（带用于开发的工具软件集成环境）；A2000TM4实验底板（带TM4C1294板卡）。

### 仪器工具

台式稳压电源；数字示波器；多用电表；函数信号发生器；电烙铁等。

### 主要元件类器材

本实验的主要元件器材见列表1-1。

表1-1 实验主要元件类器材

|  |  |  |
| --- | --- | --- |
| **型号或名称** | **规格 封装** | **备注** |
| 50KW | 多圈精密可调电阻 | 用作可调分压电阻 |
| 单排针 | 20芯2.54mm | 做接线点或检测点等 |
| TL431 | 插脚TO-92 | 基准电压源 |
| 一单元洞洞板 | 课程定制线路板 | 用于焊装自制电路 |
| ø0.8焊锡 |  | 焊料 |
| ø0.5导线 | 单芯硬线 | 焊连电路用 |
| 香蕉头、螺母、垫圈 | 三件套装 | 与洞洞板配用，作为电路供电接线柱（脚） |
| 色环电阻 | 0.25W系列，插脚 | 根据设计选用 |
| 陶瓷电容 | 系列，插脚 | 根据设计选用 |
| 电解电容 | 系列，插脚 | 根据设计选用 |
| 杜邦头跳接线 | 成品，单根 | 线路连接用 |
| TLV2372 | DIP8 | 运放芯片 |
| LM358 | DIP8 | 运放芯片，引脚与 TLV2372 兼容。在实验  现场临时借用，用完必须及时归还。 |
| IC 插座 8 脚 | DIP8 | 与运放芯片配套 |
| 1N4148 | 插脚 | 普通开关二极管 |
| 3.3V 齐纳二极管 | 插脚 | 稳压二极管 |
| 数字化温度传感器LM75B模块 | 模块电路 | 必须插装在单排座上，不得直接焊死在洞洞板 |
| A2000TM4实验底板带TM4C1294NCPDT单片机板卡 |  | 附带 USB-miniUSB、USB-microUSB线缆各 1 根 |
| 单排座 | 引脚间距 2.54mm | 做温度传感器模块插座等 |

# 部分硬件设计与实现

## 基准电压发生电路的设计与实现

与常规稳压电路相比，所谓“基准”(Reference)电压电路有更高的稳定度。在本实验中，需要设计一个基准电压发生电路，为其他电路提供足够稳定的工作条件。我们选型使用集成器件TL431为核心器件，设计构建该电路。TL431在电路中充当分路调节器（shuntregulator）来实现基准稳压。使用TL431的基准电压电路，对应图2-1（2）中电压𝑉𝐾的取值，在一定条件范围内“可设计”或称“可设定”（programmable）。课程推荐的电路设计如图2-2。在该图中与𝑉𝐾对应的信号为𝑉𝑜𝑢𝑡， 以及标识“基准+3.10V”，即将基准电压𝑉𝑜𝑢𝑡设计为+3.10V，供后级电路使用。通过合理设计图2-2中的电阻𝑅2和𝑅5的取值，可以获得比较高精度和稳定度的+3.10V电压信号。

图表, 箱线图

描述已自动生成

(1)不带分路调节器时 (2)带分路调节器时

图 2-1 分路调节器稳压原理

根据技术资料基准电压发生电路图中的部分参数如下𝑉𝑖𝑛=+5V，𝑉ref=+2.5V，𝑉𝐾=+3.10V，根据电路图计算出𝑅2、𝑅5取值满足𝑉𝐾=𝑉ref(1+𝑅2/𝑅5)，即𝑅2/𝑅5的值为0.24，对此我们小组选用的组合为𝑅2=240Ω，𝑅5=1kΩ。基准电压发生电路图见图2-2。

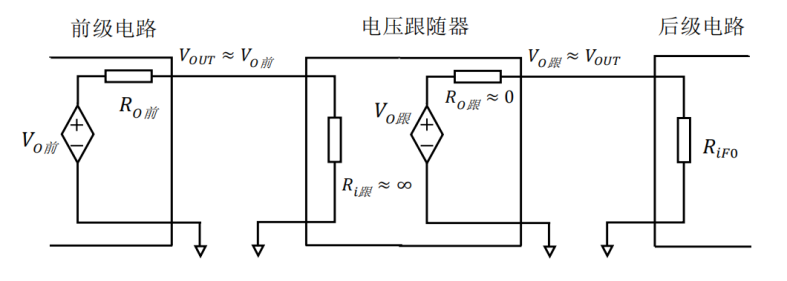
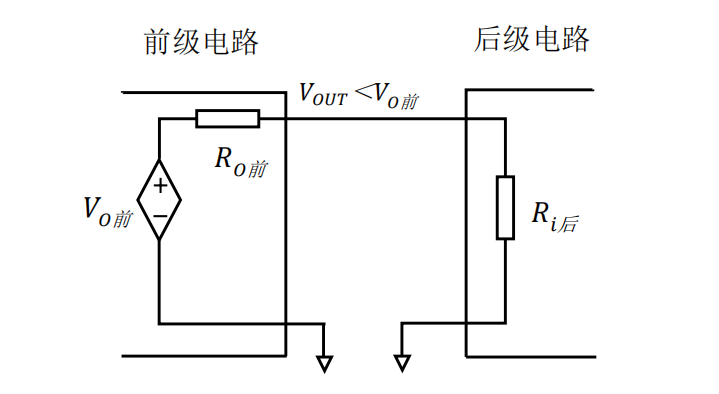
图示, 示意图

描述已自动生成

图 2-2 基准电压发生电路

## 可调分压和电压跟随电路的实现

电压跟随器顾名思义是一种电路，其输出电压跟随输入电压变化，即𝑉𝑜𝑢𝑡 = 𝑉𝑖𝑛。电路的输入输出特性，除了电压，还包括其他方面，比如阻抗特性。电压跟随器电路的输入阻抗很大，通常可近似看作无穷；它的输出阻抗很小，很多场合可近似看作零。图2-3（1）中，后级等效输入阻抗𝑅𝑖后相对于前级输出阻抗𝑅𝑜前，不能近似看作无穷（或说高阻），所以端口实际电压𝑉𝑜将小于𝑉𝑜前。这可以说是前级电路负载能力不足的一种表现，有一部分功率被𝑅𝑜前所消耗。而当两者间插入一级电压跟随器后，如图2-3（2），问题得到了解决。



(1)电路负载能力不足 (2)插入电压跟随器

图 2-2 使用电压跟随器改善电路负载能力

实际电路中，有些器件对接入的信号电压范围有技术限制。比如本实验中，单片机芯片引脚输

入信号电压应尽量限制在[0V，3.3V]（即单片机所获供电的上下电轨）之间，或至少不要越界过多，

否则可能导致单片机片内电路损伤。

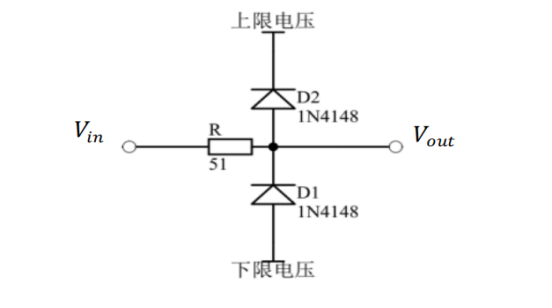


图 2-3 简易二极管钳位防过压保护

图2-3电路中，假定信号电压𝑉𝑖𝑛在正常情况下不会超越[0V，3.3V]区间，但为了防止某些特殊或极端状况下（比如上电初始、人为操作疏失等）的电压越界，利用了二极管正向导通特性限制𝑉𝑜𝑢𝑡 的电压范围。实际大致可以限制在[下限电压-0.7V，上限电压+0.7V]区间。正常情况时，二极管均不导通，𝑉𝑜𝑢𝑡 ≈ 𝑉𝑖𝑛；当𝑉𝑖𝑛越界，某一二极管导通，对𝑉𝑜𝑢𝑡钳位（钳制电位继续越界变化），为防止二极管电流过大，设计了电阻R可以起到一定限流作用。

图2-4为加了防过压保护电路的可调分压和电压跟随电路，实际测得的P2点电压可调范围为0~3.105V，P3点为0~3.102V。

图示, 示意图

描述已自动生成

图 2-4 为加了防过压保护电路的可调分压和电压跟随电路

## 温度传感器电路的实现

图2-5为温度传感器模块焊装排针引脚，焊装配置两枚温度传感器模块的从设备地址最低3位。将其中一枚配置为000安装在左侧，另一枚配置为111安装在右侧。带I2C接口的温度传感器模块电路见图2-6，

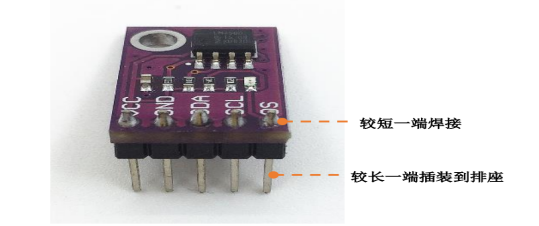


图 2-5 温度传感器模块焊装排针引脚

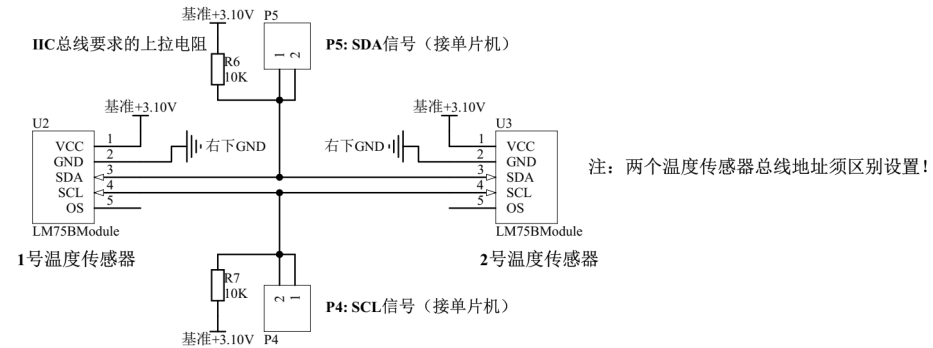


图 2-6 带I2C接口的温度传感器模块电路

## 频率信号整形调理电路的实现

图2-7为接收机侧频率信号整形调理电路，当P4接入正弦波时，TP1处为正弦波，而TP2 TP3 P5处波形均为矩形波。

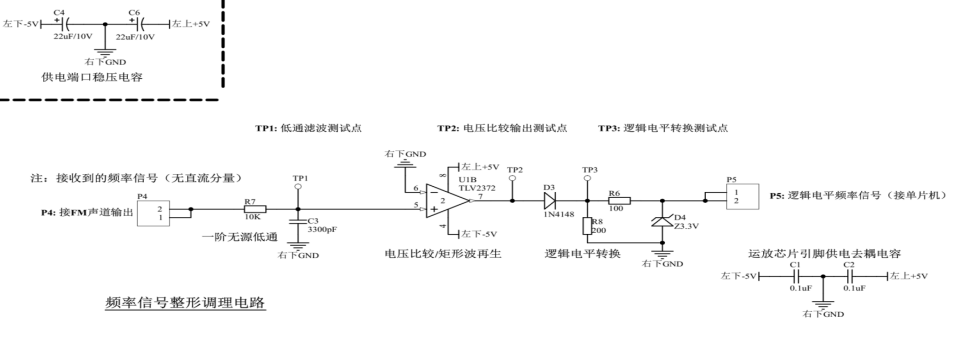


图2-7 接收机侧频率信号整形调理电路

## 发射机各硬件功能模块的连接方式

图2-8为A2000TM4 底板的供电链方案示意图，发射机洞洞板部分安装在拓展版插位B上，单片机引脚、液晶显示模块、FM发射模块之间连接方式见图2-9，安装在拓展版插位A上。

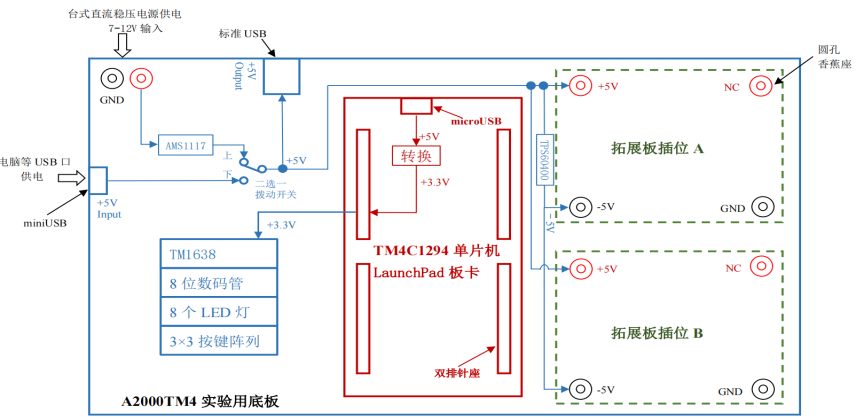


图 2-8 A2000TM4底板的供电链方案示意图

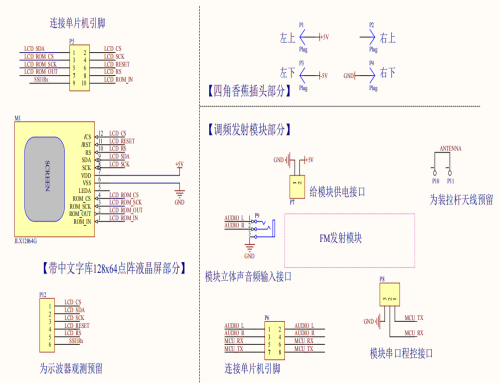


图 2-9 调频发射实验单元板电气原理图

## 接收机各硬件功能模块的连接方式

图2-8为A2000TM4 底板的供电链方案示意图，接收机洞洞板部分安装在拓展版插位B上，单片机引脚、液晶显示模块、FM接收模块之间连接方式见图2-10，安装在拓展版插位A上。

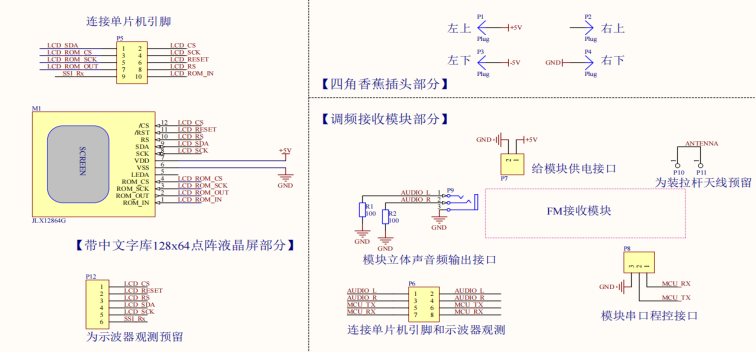


图 2-10 调频接收实验单元板电气原理图

# 算法及系统基本功能的软件设计与实现

## 温度-频率编码关系的设计和实现

使用课程组指导材料中建议的编码对应函数关系:

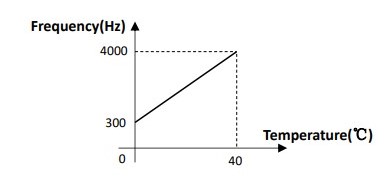


图3-1 温度-频率对应关系示意图

得到待输出的频率值，并通过PWM模块产生对应频率的方波到FM模块，最后由FM模块对应声道输出该方波信号。这样，就完成了发射端输出信号的功能。

对于接收端，由FM模块对应声道接收信号后，经过硬件工程师设计的调理、整形电路，得到所需的方波信号，选一引脚输入单片机。利用例程Frequency\_measure，用Timer1中断设置1s的定时，再用Timer0中断在这1s内按上升沿计数，得到的计数值就为频率测量值。最后利用原函数对应的反函数关系，将频率值转换为温度值：

便完成了发射-接收端温度-频率编码关系的设计和实现。

在正式实验中，我们开始时遇到了问题，发现没有实际收到信号，全是噪音。经过软、硬件工程师的排查，发现是左、右声道没有匹配，并不是代码实现的问题。经过修改连线，最后接收端译码后的温度与发射端温度测量值仅有0.3℃以内的误差；实测时发射端为28.2℃，接收端为28.1℃，很好地实现了实验需求中的效果。

## 发射端系统整合的软件和界面设计

本组在发射端、接收端系统整合的软件设计均遵循“先设计界面，再一步一步添加功能，最后得到完全整合的设计代码”这一思想，故而先写完了基本的界面设计。其UI界面设计、状态机描述均沿用指导材料中的建议，示意图如下：

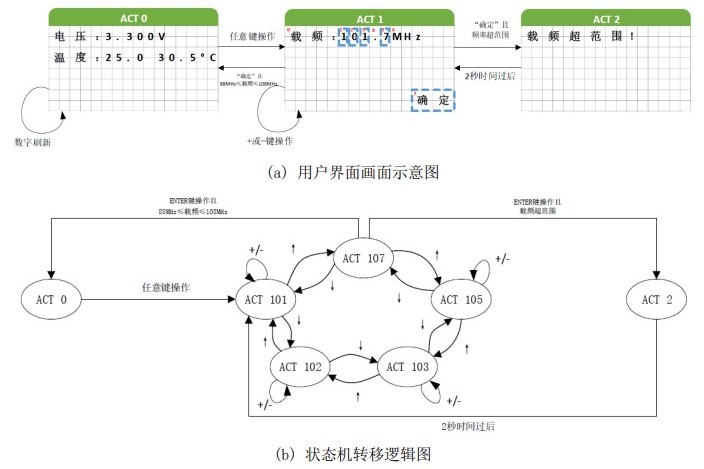


图3-2 发射端用户界面示意图

载频、温度、电压等数据的记录使用unsigned int类型，在需要显示时一一取位显示。在定时中断中使用switch语句切换状态机，以实现状态机转移的功能。最后实现的用户界面如下图所示。



图3-3 发射端用户界面ACT0实际图

其余状态不再一一展示。最后，在定时中断中刷新液晶屏，就完成了用户界面的设计。

之后开始添加功能。首先引入ADC采样模块，在数码管上分别显示ADC采样值以及电压实际值，调试完毕后显示到用户界面；其后，引入温度传感器部分，调试完毕后显示到用户界面，就完成了基础内容的实现。

最后， 添加无线发射模块。先设置FM的UART模块与UART模块通讯，使TM4C1294板卡能够控制修改FM模块的载频、音量等参数，这里采用UART6模块。调试完毕后就完成了用户界面修改载频的功能（之前只能改液晶屏的显示，没法改FM实际载频）；然后添加PWM模块，为温度编码后，将对应频率输出到FM模块，这里选用PG0引脚作为PWM复用引脚。

这样，我们就完成了发射端系统整合的软件和界面设计。

## 接收端系统整合的软件和界面设计

接收端系统整合的工作过程同发射端同理，不再赘述；其UI界面设计、状态机描述也沿用指导材料中的建议，示意图如下：

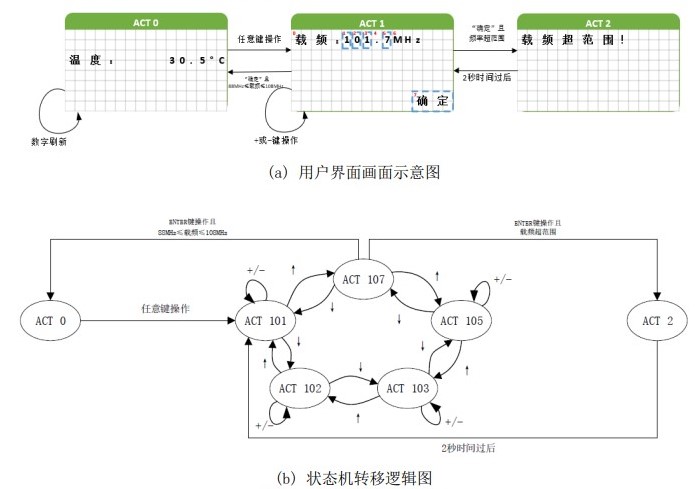


图3-4 接收端用户界面示意图

同理，在接收端添加功能。首先引入频率测量功能，设置PG0为输入引脚，先用信号发生器输入信号，保证功能正常；之后添加无线接收模块，设置FM的UART模块与UART模块通讯，使TM4C1294板卡能够控制修改FM模块的载频、音量等参数，这里和发射端一致，也采用UART6模块。最后，将接收到的频率解码为温度值，并在液晶屏上显示即可。

这样，我们就完成了接收端系统整合的软件和界面设计。经实验验证，发射机、接收机系统各项功能实现良好，完全实现了实验要求。

# 进阶功能的技术方案设计与实现

## 自播音频以及方式一接收并播放音频功能设计

本进阶功能实验中，发声模块是无源蜂鸣器，用一个特定频率的方波信号即可使之发出对应频率的蜂鸣声。故而，只需要频率、持续时间两个参数便可表示乐谱中的一个音符，我们从此便可得到记录待播放乐谱，并循环播放的思路：

使用一个自定义的Struct Note记录一个音符，再创建一个Note类型的数组melody，记录整首曲子。定义部分的伪代码如下：

struct Note {

int frequency; // 频率，单位为Hz

int duration; // 持续时间，单位为20ms\*周期数（duration）

};

struct Note melody[] = {

{440, 45}, // 6

{659, 15}, // 3

{587, 15}, // 2

{659, 15}, // 3

{523, 15}, // 1

{494, 15}, // 7

//乐曲其后的部分同理……

}

最后定义一个play函数，循环播放：

void play() {

if (currentDuration == 0) {

// 如果当前音符的持续时间已经结束，开始播放下一个音符

PWMStart(melody[currentNote].frequency);

currentDuration = melody[currentNote].duration;

currentNote++;

if (currentNote >= sizeof(melody) / sizeof(melody[0])) {

currentNote = 0; // 循环播放

}

}

currentDuration--;

}

同时，将PWM模块按照指导材料中的伪代码，改为GEN1输出，GPIO引脚复用到PF3，恰为底板上无源蜂鸣器对应的引脚。在Systick\_Handler中断函数中调用play函数，就完成了自播音频的实现。

同理，若要输出相应方波音频信号到接收端，则可在发射端将PWM模块同时配置GEN2输出，GPIO引脚复用到PG0，再将PG0连接到FM模块；最后设置PWMStart函数为同时在GEN1与GEN2输出相应频率的方波，即可发射乐曲。

在接收端，与基础功能同理，只需将FM模块接收到的信号经过调理、整形电路，输入到PF3引脚，再将PF3设置为GPIO输入，即可实现以方式一接收并播放音频的功能。

## 方式二接收并播放音频以及录音功能设计

本组实现以方式二接受并播放音频的技术路径采用的是测频率的方法。实现方式如下所述。

实现的总体思路为：利用课程组提供的频率测量例程测量输入到PG0的频率，再利用PWM模块生成一个相同频率的方波，输出到PF3引脚。这样，就实现了先使用板卡读入音频信号，再输出的功能。

实际代码编写过程中，发现频率测量例程中测频率的时间（也即Timer1中断单次计时的时间）过长，乐曲失真很明显。我们先进行了讨论，注意到每个音符的持续时间最短为300ms，故而我们认为设置中断计时为其因子100ms可能是一个比较好的选择。经过不断实验，我们发现，基本上中断时长设置越短，失真情况越少；但若设置过短，可能会出现卡顿现象。最终，我们将中断计时设置为50ms。这时，已经较难听出大面积的失真了。

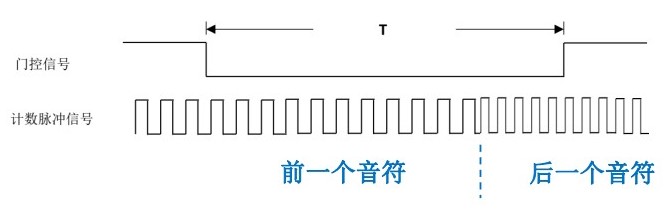


图4-1 频率测量法出现失真原因示意图

最终，我们的Timer1相关设置代码如下：

void TIMER1A\_Handler(void)

{

TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // 清除中断标志

g\_ui32TPreCount = g\_ui32TCurCount; // 保存上一次TIMER0边沿计数值

g\_ui32TCurCount = TimerValueGet(TIMER0\_BASE, TIMER\_A); // 读取TIMER0边沿计数值

TimerDisable(TIMER0\_BASE, TIMER\_A); // 停止TIMER0边沿计数

g\_ui8INTStatus = 1; // 50ms计数完成

}

void Timer1Init(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1); // TIMER1 使能

TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_ONE\_SHOT); // 设置为 32 位 1次 定时器

TimerLoadSet(TIMER1\_BASE, TIMER\_A, (ui32SysClock / 20) - 1); // TIMER1A装载计数值50ms

}

本组录音功能的设计也是采用了技术路径中的建议，但是做了一处简化的改动：利用一个数组存储每次接收到的频率，而非使用原来{频率，持续时间}的组合。这样，存储的内容更加简单，只需每次测量完频率的同时记录到数组中即可；坏处是占用存储空间更大，最长的录音时间不如原方案。

最终，虽然没有测试最长的录音时间为多大，但在测试时在老师的建议下完成了40s的录音，顺利实现了这一功能。

## 系统整合设计

各项功能实现完全后，我们采用底板上未使用的按键（即按键1,3,7,9）实现系统整合、切换系统工作状态的功能。底板按键切换功能介绍如下：

1.发射端：

“1”键：按下切换播音、测温功能。播音时，液晶屏界面不刷新，FM模块发射音符频率的方波，蜂鸣器鸣叫；测温时，液晶屏界面刷新显示温度与电压，FM模块发射温度编码频率的方波，蜂鸣器不鸣叫。

“3”键：按下便可实现“一键静音”，只在播音状态下有作用。

2.接收端：

“1”键：按下切换播音、测温功能。播音时，测量频率后产生对应频率方波接PF3引脚，液晶屏界面不刷新，蜂鸣器鸣叫；测温时， 测量频率后解码为对应温度值，液晶屏界面刷新显示接收温度，蜂鸣器不鸣叫。

接收端未要求“一键静音”功能，故不做整合。

综上设计，即完成了系统整合，可以让系统在基础功能与进阶功能之间自由切换。

# 附录A 学习心得和意见建议

## 学习感受与心得

经过一学期的学习，本组组员已熟练掌握对应负责部分的知识、技能，基本达到了课程组的培养要求。组员们积极分工合作，以最高的效率完成自己负责部分的内容，再合作debug，圆满完成了实验任务。

同时，组员还认识到了查找厂商资料能力的重要性——这也是作为一个工程师的基本技能。例如，在进阶功能验收时，老师要求修改输出引脚，但我们未能在规定时间完成；事后查找user guide，发现失败原因为当时试图进行引脚复用的引脚并不支持相应功能，必须找到可用的引脚才可完成。这次经历让我们认识到了自己的不足之处，同时也鞭策我们继续完善自己的认识与技能，提升自己的能力。

## 对本课程的一点小建议

本课程的教学用课堂讲授辅以视频教学的形式，高效且简洁。我们唯一的小建议是，希望老师们在一些进阶功能的引路部分做更详细的讲解，或是将入门部分的内容放在更醒目的地方。这样，同学们就不会在入门部分就因为找不到教程而犯错，从而丧失自信心或是降低效率。

例如，在进阶实验内容中，最初利用PWM模块输出方波到PF3引脚时，相应的伪代码在课程组提供的“指导材料：利用无源蜂鸣器实现音乐播放.pdf”文件的最后，而本组起步时未发现该资料的位置，导致开始时错误地设置GEN2输出。之后翻阅所有材料无果，才终于在网络上搜索到相关信息并改正。在之后编写乐谱之时，才发现在乐谱的最后有相应的教学伪代码，由此发现自己浪费了大量时间。如果将相应方法作成视频讲述，或是放在技术路径pdf中，我们可能就能更加顺利地完成进阶任务。

以上就是本小组对课程的一点小建议，希望本课程在各位老师们的带领下越办越好！

# 附录B 核心代码清单

## 发射端程序核心代码清单

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h" // 基址宏定义

#include "inc/hw\_types.h"

#include "inc/hw\_timer.h" // ??????????

#include "inc/hw\_ints.h"

#include "driverlib/debug.h" // 调试用

#include "driverlib/gpio.h" // 通用IO口宏定义

#include "driverlib/pin\_map.h" // TM4C系列MCU外围设备管脚宏定义

#include "driverlib/sysctl.h" // 系统控制定义

#include "driverlib/systick.h" // SysTick Driver 原型

#include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型

#include "driverlib/fpu.h"

#include "driverlib/timer.h" // ?Timer???????

#include "driverlib/pwm.h"

#include "driverlib/uart.h"

#include "driverlib/adc.h"

#include "JXL.c"

#include "tm1638.h" // 与控制TM1638芯片有关的函数

#include "LM75BD.h" // 与控制LM75BD芯片有关的函数

#include "PWM.h"

#include "ADC.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 宏定义

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define SYSTICK\_FREQUENCY 50 // SysTick频率为50Hz，即循环定时周期20ms

#define MilliSecond 4000

#define V\_T1s 50 // 1s软件定时器溢出值，50个20ms

#define V\_T40ms 2

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型声明

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GPIOInit(void); // GPIO初始化

void SysTickInit(void); // 设置SysTick中断

void DevicesInit(void); // MCU器件初始化，注：会调用上述函数

void UARTInit(void);

void UARTStringPut(uint32\_t ui32Base,const char \*cMessage);

void DelayMilliSec(uint32\_t ui32DelaySecond);

void disp\_number(uint32\_t page,uint32\_t column,uint8\_t number,uint8\_t inverse);

void refresh\_temp\_in(void);

void refresh\_volt\_in(void);

void refresh\_temp\_lcd(void);

void refresh\_volt\_lcd(void);

void trans\_temp(void);

void play(void);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 变量定义

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

uint8\_t clock40ms = 0;

uint8\_t clock40ms\_flag = 0;

// 软件定时器计数

uint8\_t clock1s = 0;

// 软件定时器溢出标志

uint8\_t clock1s\_flag = 0;

// 测试用计数器

uint32\_t test\_counter = 0;

// 8位数码管显示的数字或字母符号

// 注：板上数码位从左到右序号排列为4、5、6、7、0、1、2、3

uint8\_t digit[8]={' ',' ',' ',' ',' ',' ',' ',' '};

// 8位小数点 1亮 0灭

// 注：板上数码位小数点从左到右序号排列为4、5、6、7、0、1、2、3

uint8\_t pnt = 0x44;

// 8个LED指示灯状态，0灭，1亮

// 注：板上指示灯从左到右序号排列为7、6、5、4、3、2、1、0

// 对应元件LED8、LED7、LED6、LED5、LED4、LED3、LED2、LED1

uint8\_t led[] = {1, 1, 1, 1, 1, 1, 1, 0};

uint8\_t temp1[]={0,0,0,0};

uint8\_t temp2[]={0,0,0,0};

uint8\_t volt[]={0,0,0,0};

uint32\_t freq\_square=0;

uint8\_t freq\_lcd[]={1,0,1,0};

uint8\_t freq\_lcd\_random[]={1,0,1,0};

// 当前按键值

uint8\_t key\_code = 0;

uint8\_t key\_cnt = 0;

uint8\_t prestate=0;

uint8\_t curstate=0;

uint8\_t key=0;

uint8\_t actmode=0;

uint8\_t freqmode=1;

uint32\_t range\_flag=0;

// AIN2(PE1) ADC???[0-4095]

uint32\_t ui32ADC0Value;

// AIN2???(???0.01V) [0.00-3.30]

uint32\_t ui32ADC0Voltage;

// 系统时钟频率

uint32\_t ui32SysClock;

uint32\_t g\_ui32SysClock;

// 返回温度值 1LSB=0.1℃

int16\_t i16Temperature1;

int16\_t i16Temperature2;

uint8\_t ui8DigitRefresh = 0;

uint16\_t ui16Temp1,ui16Temp2;

// 系统当前功能状态

bool system\_state = true;

//是否暂停

bool non\_stop = true;

struct Note {

int frequency; // 频率，单位为Hz

int duration; // 持续时间，单位为20ms周期数

};

struct Note melody[] = {

{440, 45}, // 6. ·，频率为440Hz，持续时间为900ms

{659, 15}, // 3 \_ ，频率为659Hz，持续时间为300ms

{587, 15}, // 2 \_ ，频率为587Hz，持续时间为300ms

{659, 15}, // 3 \_ ，频率为659Hz，持续时间为300ms

{523, 15}, // 1 \_ ，频率为523Hz，持续时间为300ms

{494, 15}, // 7. \_ ，频率为494Hz，持续时间为300ms

{440, 120}, // 6. - - - ，频率为523Hz，持续时间为2400ms

{659, 15}, // 3 \_

{880, 30}, // 6

{784, 15}, // 5 \_

{659, 15}, // 3 \_

{587, 15}, // 2 \_

{523, 15}, // 1 \_

{587, 15}, // 2 \_

{659, 90}, // 3 - -

{659, 15}, // 3 \_

{784, 15}, // 5 \_

{880, 45}, // 6·

{988, 15}, // 7 \_

{880, 15}, // 6 \_

{784, 15}, // 5 \_

{659, 15}, // 3 \_

{587, 15}, // 2 \_

{523, 15}, // 1 \_

{523, 30}, // 1

{587, 15}, // 2 \_

{659, 60}, // 3 -

{587, 45}, // 2·

{659, 15}, // 3 \_

{494, 15}, // 7. \_

{440, 15}, // 6. \_

{392, 30}, // 5.

{440, 120}, // 6. - - -

// 在此添加更多音符...

};

volatile int currentNote = 0; // 当前播放的音符索引

volatile int currentDuration = 0; // 当前音符已经播放的时间

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 主程序

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main(void)

{

DevicesInit(); // MCU器件初始化

SysCtlDelay(60 \* ( ui32SysClock / 3000)); // 延时>60ms,等待TM1638上电完成

TM1638\_Init(); // 初始化TM1638

initial\_lcd();

UARTStringPut(UART6\_BASE, (const char \*)"AT+FRE=1010\r\n");

UARTStringPut(UART6\_BASE, (const char \*)"AT+VOL=30\r\n");

i16Temperature1 = GetTemputerature(LM75BD\_ADR2);

i16Temperature2 = GetTemputerature(LM75BD\_ADR1);

if(i16Temperature1 < 0)

{

ui16Temp1 = -i16Temperature1;

temp1[0]=1;

}

else

{

ui16Temp1 = i16Temperature1;

temp1[0]=0;

}

if(i16Temperature2 < 0)

{

ui16Temp2 = -i16Temperature2;

temp2[0]=1;

}

else

{

ui16Temp2 = i16Temperature2;

temp2[0]=0;

}

temp1[1] = ui16Temp1 / 100; // 计算十位数

temp1[2] = ui16Temp1 / 10 % 10; // 计算个位数

temp1[3] = ui16Temp1 % 10;

temp2[1] = ui16Temp2 / 100; // 计算十位数

temp2[2] = ui16Temp2 / 10 % 10; // 计算个位数

temp2[3] = ui16Temp2 % 10;

ui32ADC0Value = ADC\_Sample();

ui32ADC0Voltage = ui32ADC0Value \* 3300 / 4095;

volt[0] = (ui32ADC0Voltage / 1000) % 10; // ????????

volt[1] = (ui32ADC0Voltage / 100) % 10; // ?????????

volt[2] = (ui32ADC0Voltage/10) % 10; // ?????????

volt[3] = ui32ADC0Voltage%10;

clear\_screen();

display\_GB2312\_string(1,1,"电压:",0);

display\_GB2312\_string(3,1,"温度:",0);

disp\_number(1,41,volt[0],0);

display\_GB2312\_string(1,49,".",0);

disp\_number(1,57,volt[1],0);

disp\_number(1,65,volt[2],0);

disp\_number(1,73,volt[3],0);

display\_GB2312\_string(1,81,"V",0);

if(temp1[0]==1){display\_GB2312\_string(3,41,"-",0);}

else{display\_GB2312\_string(3,41," ",0);}

disp\_number(3,49,temp1[1],0);

disp\_number(3,57,temp1[2],0);

display\_GB2312\_string(3,65,".",0);

disp\_number(3,73,temp1[3],0);

display\_GB2312\_string(3,89,"℃",0);

if(temp2[0]==1){display\_GB2312\_string(5,41,"-",0);}

else{display\_GB2312\_string(5,41," ",0);}

disp\_number(5,49,temp2[1],0);

disp\_number(5,57,temp2[2],0);

display\_GB2312\_string(5,65,".",0);

disp\_number(5,73,temp2[3],0);

display\_GB2312\_string(5,89,"℃",0);

while (1)

{

if(system\_state){}

else {

if (clock1s\_flag == 1) // ??0.1??????

{

clock1s\_flag=0;

refresh\_temp\_in();

trans\_temp();

if(actmode ==0) {refresh\_temp\_lcd();}

}

if (clock40ms\_flag == 1) // ??0.1??????

{

clock40ms\_flag=0;

refresh\_volt\_in();

if(actmode ==0) {refresh\_volt\_lcd();}

}

if(key==0){

key=0;

}

else if(key==1){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

}

else if(key==2){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

else if(actmode ==1){

if(freqmode ==1){

if(freq\_lcd\_random[0]!=9){

++freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],1);

}

else{

freq\_lcd\_random[0]=0;

disp\_number(1,41,freq\_lcd\_random[0],1);

}

}

else if(freqmode ==2){

if(freq\_lcd\_random[1]==9){

freq\_lcd\_random[1]=0;

disp\_number(1,49,freq\_lcd\_random[1],1);

if(freq\_lcd\_random[0]==9){

freq\_lcd\_random[0]=0;

disp\_number(1,41,freq\_lcd\_random[0],0);

}

else{

++freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],0);

}

}

else{

++freq\_lcd\_random[1];

disp\_number(1,49,freq\_lcd\_random[1],1);

}

}

else if(freqmode ==3){

if(freq\_lcd\_random[2]==9){

freq\_lcd\_random[2]=0;

disp\_number(1,57,freq\_lcd\_random[2],1);

if(freq\_lcd\_random[1]==9){

freq\_lcd\_random[1]=0;

disp\_number(1,49,freq\_lcd\_random[1],0);

if(freq\_lcd\_random[0]==9){

freq\_lcd\_random[0]=0;

disp\_number(1,41,freq\_lcd\_random[0],0);

}

else{

++freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],0);

}

}

else{

++freq\_lcd\_random[1];

disp\_number(1,49,freq\_lcd\_random[1],0);

}

}

else{

++freq\_lcd\_random[2];

disp\_number(1,57,freq\_lcd\_random[2],1);

}

}

else if(freqmode ==5){

if(freq\_lcd\_random[3]==9){

freq\_lcd\_random[3]=0;

disp\_number(1,73,freq\_lcd\_random[3],1);

if(freq\_lcd\_random[2]==9){

freq\_lcd\_random[2]=0;

disp\_number(1,57,freq\_lcd\_random[2],0);

if(freq\_lcd\_random[1]==9){

freq\_lcd\_random[1]=0;

disp\_number(1,49,freq\_lcd\_random[1],0);

if(freq\_lcd\_random[0]==9){

freq\_lcd\_random[0]=0;

disp\_number(1,41,freq\_lcd\_random[0],0);

}

else{

++freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],0);

}

}

else{

++freq\_lcd\_random[1];

disp\_number(1,49,freq\_lcd\_random[1],0);

}

}

else{

++freq\_lcd\_random[2];

disp\_number(1,57,freq\_lcd\_random[2],0);

}

}

else{

++freq\_lcd\_random[3];

disp\_number(1,73,freq\_lcd\_random[3],1);

}

}

else if(freqmode ==7){

}

}

}

else if(key==3){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

}

else if(key==4){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

else if(actmode ==1){

if(freqmode ==1){

freqmode =7;

disp\_number(1,41,freq\_lcd\_random[0],0);

display\_GB2312\_string(7,97,"确定",1);

}

else if(freqmode ==2){

if(freq\_lcd\_random[0]==0){

freqmode =7;

display\_GB2312\_string(7,97,"确定",1);

disp\_number(1,49,freq\_lcd\_random[1],0);

}

else {

freqmode =1;

disp\_number(1,49,freq\_lcd\_random[1],0);

disp\_number(1,41,freq\_lcd\_random[0],1);

}

}

else if(freqmode ==3){

freqmode =2;

disp\_number(1,49,freq\_lcd\_random[1],1);

disp\_number(1,57,freq\_lcd\_random[2],0);

}

else if(freqmode ==5){

freqmode =3;

disp\_number(1,57,freq\_lcd\_random[2],1);

disp\_number(1,73,freq\_lcd\_random[3],0);

}

else if(freqmode ==7){

freqmode =5;

display\_GB2312\_string(7,97,"确定",0);

disp\_number(1,73,freq\_lcd\_random[3],1);

}

}

}

else if(key==5){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

else if(actmode ==1){

if(freqmode ==7){

range\_flag =freq\_lcd\_random[0]\*1000+freq\_lcd\_random[1]\*100+freq\_lcd\_random[2]\*10+freq\_lcd\_random[3];

if(range\_flag>=880&&range\_flag<=1080){

freq\_lcd[0]=freq\_lcd\_random[0];

freq\_lcd[1]=freq\_lcd\_random[1];

freq\_lcd[2]=freq\_lcd\_random[2];

freq\_lcd[3]=freq\_lcd\_random[3];

UARTCharPut(UART6\_BASE,'A');

UARTCharPut(UART6\_BASE,'T');

UARTCharPut(UART6\_BASE,'+');

UARTCharPut(UART6\_BASE,'F');

UARTCharPut(UART6\_BASE,'R');

UARTCharPut(UART6\_BASE,'E');

UARTCharPut(UART6\_BASE,'=');

if(freq\_lcd[0]==0){

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[1]);

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[2]);

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[3]);

UARTCharPut(UART6\_BASE,'\r');

UARTCharPut(UART6\_BASE,'\n');

}

else{

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[0]);

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[1]);

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[2]);

UARTCharPut(UART6\_BASE,'0'+freq\_lcd[3]);

UARTCharPut(UART6\_BASE,'\r');

UARTCharPut(UART6\_BASE,'\n');

}

actmode =0;

clear\_screen();

display\_GB2312\_string(1,1,"电压:",0);

display\_GB2312\_string(3,1,"温度:",0);

disp\_number(1,41,volt[0],0);

display\_GB2312\_string(1,49,".",0);

disp\_number(1,57,volt[1],0);

disp\_number(1,65,volt[2],0);

disp\_number(1,73,volt[3],0);

display\_GB2312\_string(1,81,"V",0);

if(temp1[0]==1){display\_GB2312\_string(3,41,"-",0);}

else{display\_GB2312\_string(3,41," ",0);}

disp\_number(3,49,temp1[1],0);

disp\_number(3,57,temp1[2],0);

display\_GB2312\_string(3,65,".",0);

disp\_number(3,73,temp1[3],0);

display\_GB2312\_string(3,89,"℃",0);

if(temp2[0]==1){display\_GB2312\_string(5,41,"-",0);}

else{display\_GB2312\_string(5,41," ",0);}

disp\_number(5,49,temp2[1],0);

disp\_number(5,57,temp2[2],0);

display\_GB2312\_string(5,65,".",0);

disp\_number(5,73,temp2[3],0);

display\_GB2312\_string(5,89,"℃",0);

}

else{

freq\_lcd\_random[0]=freq\_lcd[0];

freq\_lcd\_random[1]=freq\_lcd[1];

freq\_lcd\_random[2]=freq\_lcd[2];

freq\_lcd\_random[3]=freq\_lcd[3];

clear\_screen();

display\_GB2312\_string(1,1,"载频超范围！",0);

LCD\_delay(81000);

clear\_screen();

actmode =1;

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

}

}

}

else if(key==6){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

else if(actmode ==1){

if(freqmode ==1){

freqmode =2;

disp\_number(1,41,freq\_lcd\_random[0],0);

disp\_number(1,49,freq\_lcd\_random[1],1);

}

else if(freqmode ==2){

freqmode =3;

disp\_number(1,49,freq\_lcd\_random[1],0);

disp\_number(1,57,freq\_lcd\_random[2],1);

}

else if(freqmode ==3){

freqmode =5;

disp\_number(1,57,freq\_lcd\_random[2],0);

disp\_number(1,73,freq\_lcd\_random[3],1);

}

else if(freqmode ==5){

freqmode =7;

disp\_number(1,73,freq\_lcd\_random[3],0);

display\_GB2312\_string(7,97,"确定",1);

}

else if(freqmode ==7){

if(freq\_lcd\_random[0]==0){

freqmode =2;

display\_GB2312\_string(7,97,"确定",0);

disp\_number(1,49,freq\_lcd\_random[1],1);

}

else {

freqmode =1;

display\_GB2312\_string(7,97,"确定",0);

disp\_number(1,41,freq\_lcd\_random[0],1);

}

}

}

}

else if(key==7){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

}

else if(key==8){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

else if(actmode ==1){

if(freqmode ==1){

if(freq\_lcd\_random[0]==0){

freq\_lcd\_random[0]=9;

disp\_number(1,41,freq\_lcd\_random[0],1);

}

else{

--freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],1);

}

}

else if(freqmode ==2){

if(freq\_lcd\_random[1]==0){

freq\_lcd\_random[1]=9;

disp\_number(1,49,freq\_lcd\_random[1],1);

if(freq\_lcd\_random[0]==0){

freq\_lcd\_random[0]=9;

disp\_number(1,41,freq\_lcd\_random[0],0);

}

else{

--freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],0);

}

}

else{

--freq\_lcd\_random[1];

disp\_number(1,49,freq\_lcd\_random[1],1);

}

}

else if(freqmode ==3){

if(freq\_lcd\_random[2]==0){

freq\_lcd\_random[2]=9;

disp\_number(1,57,freq\_lcd\_random[2],1);

if(freq\_lcd\_random[1]==0){

freq\_lcd\_random[1]=9;

disp\_number(1,49,freq\_lcd\_random[1],0);

if(freq\_lcd\_random[0]==0){

freq\_lcd\_random[0]=9;

disp\_number(1,41,freq\_lcd\_random[0],0);

}

else{

--freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],0);

}

}

else{

--freq\_lcd\_random[1];

disp\_number(1,49,freq\_lcd\_random[1],0);

}

}

else{

--freq\_lcd\_random[2];

disp\_number(1,57,freq\_lcd\_random[2],1);

}

}

else if(freqmode ==5){

if(freq\_lcd\_random[3]==0){

freq\_lcd\_random[3]=9;

disp\_number(1,73,freq\_lcd\_random[3],1);

if(freq\_lcd\_random[2]==0){

freq\_lcd\_random[2]=9;

disp\_number(1,57,freq\_lcd\_random[2],0);

if(freq\_lcd\_random[1]==0){

freq\_lcd\_random[1]=9;

disp\_number(1,49,freq\_lcd\_random[1],0);

if(freq\_lcd\_random[0]==0){

freq\_lcd\_random[0]=9;

disp\_number(1,41,freq\_lcd\_random[0],0);

}

else{

--freq\_lcd\_random[0];

disp\_number(1,41,freq\_lcd\_random[0],0);

}

}

else{

--freq\_lcd\_random[1];

disp\_number(1,49,freq\_lcd\_random[1],0);

}

}

else{

--freq\_lcd\_random[2];

disp\_number(1,57,freq\_lcd\_random[2],0);

}

}

else{

--freq\_lcd\_random[3];

disp\_number(1,73,freq\_lcd\_random[3],1);

}

}

else if(freqmode ==7){

}

}

}

else if(key==9){

key=0;

if(actmode ==0){

actmode =1;

clear\_screen();

display\_GB2312\_string(1,1,"载频:",0);

if(freq\_lcd[0]==0){freqmode =2;display\_GB2312\_string(1,41," ",0);}

else{freqmode =1;disp\_number(1,41,freq\_lcd[0],1);}

if(freq\_lcd[0]==0){disp\_number(1,49,freq\_lcd[1],1);}

else{disp\_number(1,49,freq\_lcd[1],0);}

disp\_number(1,57,freq\_lcd[2],0);

display\_GB2312\_string(1,65,".",0);

disp\_number(1,73,freq\_lcd[3],0);

display\_GB2312\_string(1,81,"MHz",0);

display\_GB2312\_string(7,97,"确定",0);

}

}

}

}

}

void trans\_temp(void) //使用第二路温度值

{

freq\_square =92.5\*(temp2[1]\*10+temp2[2]+0.1\*temp2[3])+300;

PWMStart2(freq\_square);

}

void refresh\_volt\_lcd(void)

{

disp\_number(1,41,volt[0],0);

disp\_number(1,57,volt[1],0);

disp\_number(1,65,volt[2],0);

disp\_number(1,73,volt[3],0);

}

void refresh\_temp\_lcd(void)

{

if(temp1[0]==1){display\_GB2312\_string(3,41,"-",0);}

else{display\_GB2312\_string(3,41," ",0);}

disp\_number(3,49,temp1[1],0);

disp\_number(3,57,temp1[2],0);

disp\_number(3,73,temp1[3],0);

if(temp2[0]==1){display\_GB2312\_string(5,41,"-",0);}

else{display\_GB2312\_string(5,41," ",0);}

disp\_number(5,49,temp2[1],0);

disp\_number(5,57,temp2[2],0);

disp\_number(5,73,temp2[3],0);

}

void DelayMilliSec(uint32\_t ui32DelaySecond)

{

uint32\_t ui32Loop;

ui32DelaySecond = ui32DelaySecond \* MilliSecond;

for(ui32Loop = 0; ui32Loop < ui32DelaySecond; ui32Loop++){ };

}

void disp\_number(uint32\_t page,uint32\_t column,uint8\_t number,uint8\_t inverse){

if(number ==0){display\_GB2312\_string(page,column,"0",inverse);}

if(number ==1){display\_GB2312\_string(page,column,"1",inverse);}

if(number ==2){display\_GB2312\_string(page,column,"2",inverse);}

if(number ==3){display\_GB2312\_string(page,column,"3",inverse);}

if(number ==4){display\_GB2312\_string(page,column,"4",inverse);}

if(number ==5){display\_GB2312\_string(page,column,"5",inverse);}

if(number ==6){display\_GB2312\_string(page,column,"6",inverse);}

if(number ==7){display\_GB2312\_string(page,column,"7",inverse);}

if(number ==8){display\_GB2312\_string(page,column,"8",inverse);}

if(number ==9){display\_GB2312\_string(page,column,"9",inverse);}

}

void refresh\_temp\_in(void){

i16Temperature1 = GetTemputerature(LM75BD\_ADR2);

i16Temperature2 = GetTemputerature(LM75BD\_ADR1);

if(i16Temperature1 < 0)

{

ui16Temp1 = -i16Temperature1;

temp1[0]=1;

}

else

{

ui16Temp1 = i16Temperature1;

temp1[0]=0;

}

if(i16Temperature2 < 0)

{

ui16Temp2 = -i16Temperature2;

temp2[0]=1;

}

else

{

ui16Temp2 = i16Temperature2;

temp2[0]=0;

}

temp1[1] = ui16Temp1 / 100; // 计算十位数

temp1[2] = ui16Temp1 / 10 % 10; // 计算个位数

temp1[3] = ui16Temp1 % 10;

temp2[1] = ui16Temp2 / 100; // 计算十位数

temp2[2] = ui16Temp2 / 10 % 10; // 计算个位数

temp2[3] = ui16Temp2 % 10;

}

void refresh\_volt\_in(void){

ui32ADC0Value= (ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample()+ADC\_Sample())/15;

ui32ADC0Voltage = ui32ADC0Value \* 3300 / 4095;

volt[0] = (ui32ADC0Voltage / 1000) % 10; // ????????

volt[1] = (ui32ADC0Voltage / 100) % 10; // ?????????

volt[2] = (ui32ADC0Voltage/10) % 10; // ?????????

volt[3] = ui32ADC0Voltage%10;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void GPIOInit(void)

// 函数功能：GPIO初始化。

// 使能PortK，设置PK4,PK5为输出；使能PortM，设置PM0为输出；

// PK4连接TM1638的STB，PK5连接TM1638的DIO，PM0连接TM1638的CLK；

// 使能PortL，设置LPL0,PL1为输出；PL0和PL1分别连接DAC6571的SDA和SCL。

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GPIOInit(void)

{

//配置TM1638芯片管脚

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOK); // 使能端口 K

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOK)){}; // 等待端口 K 准备完毕

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOM); // 使能端口 M

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOM)){}; // 等待端口 M 准备完毕

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // 使能端口 M

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOF)){}; // 等待端口 M 准备完毕

// 设置端口 K 的第4,5位（PK4,PK5）为输出引脚 PK4-STB PK5-DIO

GPIOPinTypeGPIOOutput(GPIO\_PORTK\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_5);

// 设置端口 M 的第0位（PM0）为输出引脚 PM0-CLK

GPIOPinTypeGPIOOutput(GPIO\_PORTM\_BASE, GPIO\_PIN\_0);

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_3);

}

void UARTStringPut(uint32\_t ui32Base,const char \*cMessage)

{

while(\*cMessage != '\0') {

UARTCharPut(ui32Base, \*(cMessage++));

}

DelayMilliSec(1);

}

void UARTInit(void)

{

// ????

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0); // ??UART0??

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA); // ???? A

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOA)); // ???? A????

GPIOPinConfigure(GPIO\_PA0\_U0RX); // ??PA0?UART0 RX??

GPIOPinConfigure(GPIO\_PA1\_U0TX); // ??PA1?UART0 TX??

// ???? A??0,1?(PA0,PA1)?UART??

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// ?????????

UARTConfigSetExpClk(UART0\_BASE,

ui32SysClock,

38400, // ???:115200

(UART\_CONFIG\_WLEN\_8 | // ???:8

UART\_CONFIG\_STOP\_ONE | // ???:1

UART\_CONFIG\_PAR\_NONE)); // ???:?

// ????

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART6); // ??UART0??

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOP); // ???? A

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOP)); // ???? A????

GPIOPinConfigure(GPIO\_PP0\_U6RX); // ??PA0?UART0 RX??

GPIOPinConfigure(GPIO\_PP1\_U6TX); // ??PA1?UART0 TX??

// ???? A??0,1?(PA0,PA1)?UART??

GPIOPinTypeUART(GPIO\_PORTP\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// ?????????

UARTConfigSetExpClk(UART6\_BASE,

ui32SysClock,

38400, // ???:115200

(UART\_CONFIG\_WLEN\_8 | // ???:8

UART\_CONFIG\_STOP\_ONE | // ???:1

UART\_CONFIG\_PAR\_NONE)); // ???:?

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：SysTickInit(void)

// 函数功能：设置SysTick中断

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SysTickInit(void)

{

SysTickPeriodSet(ui32SysClock/SYSTICK\_FREQUENCY); // 设置心跳节拍,定时周期20ms

SysTickEnable(); // SysTick使能

SysTickIntEnable(); // SysTick中断允许

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void DevicesInit(void)

// 函数功能：MCU器件初始化，包括系统时钟设置、GPIO初始化和SysTick中断设置

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void DevicesInit(void)

{

// 使用外部25MHz主时钟源，经过PLL，然后分频为16MHz

ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ |SYSCTL\_OSC\_MAIN |

SYSCTL\_USE\_PLL |SYSCTL\_CFG\_VCO\_480),

16000000);

g\_ui32SysClock=ui32SysClock;

FPULazyStackingEnable();

FPUEnable();

GPIOInit(); // GPIO初始化

I2C0Init(); // I2C0初始化

SysTickInit(); // 设置SysTick中断

UARTInit();

IntMasterEnable(); // 总中断允许

ADCInit();

PWMInit();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void SysTick\_Handler(void)

// 函数功能：SysTick中断服务程序

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SysTick\_Handler(void) // 定时周期为20ms

{

// 0.1秒钟软定时器计数

if (++clock1s >= V\_T1s)

{

clock1s\_flag = 1; // 当0.1秒到时，溢出标志置1

clock1s = 0;

}

if (++clock40ms >= V\_T40ms)

{

clock40ms\_flag = 1; // ?40ms??,?????1

clock40ms = 0;

}

// 刷新全部数码管和LED指示灯

if(ui8DigitRefresh == 0)

TM1638\_RefreshDIGIandLED(digit, pnt, led);

// 检查当前键盘输入，0代表无键操作，1-9表示有对应按键

// 键号显示在一位数码管上

key\_code = TM1638\_Readkeyboard();

prestate =curstate ;

curstate =key\_code ;

if(prestate ==0&&curstate !=0){

key=curstate ;

} else {

key=0;}

if(key == 1){

system\_state = !system\_state;

}

if(key == 3){

non\_stop = !non\_stop;

}

if(system\_state && non\_stop){

play();

} else {

PWMStop1();

}

}

void play() {

if (currentDuration == 0) {

// 如果当前音符的持续时间已经结束，开始播放下一个音符

PWMStart(melody[currentNote].frequency);

currentDuration = melody[currentNote].duration;

currentNote++;

if (currentNote >= sizeof(melody) / sizeof(melody[0])) {

currentNote = 0; // 循环播放

}

}

currentDuration--;

}

unsigned char volt1[] = "1";

unsigned char volt2[] = "1";

unsigned char volt3[] = "1";

unsigned char volt4[] = "1";

unsigned char temp1\_1[] = "1";

unsigned char temp1\_2[] = "2";

unsigned char temp1\_3[] = "3";

unsigned char temp2\_1[] = "1";

unsigned char temp2\_2[] = "2";

unsigned char temp2\_3[] = "3";

unsigned char freq\_1[] = "1";

unsigned char freq\_2[] = "1";

unsigned char freq\_3[] = "1";

unsigned char freq\_4[] = "1";

int freq1 = 1;

int freq2 = 1;

int freq3 = 1;

int freq4 = 1;

struct struct\_act

{

    unsigned char num;

    unsigned char \*str[20];

    unsigned char x[20],y[20],inverse[20];

} ;

//////////////////////////////

//State0

struct struct\_act a0;

unsigned char a0\_s0[] = "电压:";

unsigned char \*a0\_s1;

unsigned char a0\_s2[] = ".";

unsigned char \*a0\_s3;

unsigned char \*a0\_s4;

unsigned char \*a0\_s5;

unsigned char a0\_s6[] = "V";

unsigned char a0\_s7[] = "温度:";

unsigned char \*a0\_s8;

unsigned char \*a0\_s9;

unsigned char a0\_s10[] = ".";

unsigned char \*a0\_s11;

unsigned char \*a0\_s12;

unsigned char \*a0\_s13;

unsigned char a0\_s14[] = ".";

unsigned char \*a0\_s15;

unsigned char a0\_s16[] = "℃";

//State1

struct struct\_act a1;

unsigned char a1\_s0[] = "载频:";

unsigned char \*a1\_s1;

unsigned char \*a1\_s2;

unsigned char \*a1\_s3;

unsigned char a1\_s4[] = ".";

unsigned char \*a1\_s5;

unsigned char a1\_s6[] = "MHz";

unsigned char a1\_s7[] = "确定";

//State2

struct struct\_act a2;

unsigned char a2\_s0[] = "载频超范围！";

struct struct\_act \*act[1];

//////////////////////////////

unsigned int act\_state = 0;

unsigned int ui\_state=0;  //状态号

unsigned int key\_ENTER\_state=0;

unsigned int key\_ENTER\_prestate=0;

unsigned int ENTER\_key\_timer=0;

unsigned int key\_ENTER\_flag=0;

unsigned int key\_DOWN\_state=0;

unsigned int key\_DOWN\_prestate=0;

unsigned int key\_DOWN\_timer=0;

unsigned int key\_DOWN\_flag=0;

unsigned int key\_UP\_state=0;

unsigned int key\_UP\_prestate=0;

unsigned int key\_UP\_timer=0;

unsigned int key\_UP\_flag=0;

unsigned int key\_INCREASE\_state=0;

unsigned int key\_INCREASE\_prestate=0;

unsigned int key\_INCREASE\_timer=0;

unsigned int key\_INCREASE\_flag=0;

unsigned int key\_DECREASE\_state=0;

unsigned int key\_DECREASE\_prestate=0;

unsigned int key\_DECREASE\_timer=0;

unsigned int key\_DECREASE\_flag=0;

void frequency\_init(uint16\_t work\_freq){

    freq1 = work\_freq / 1000;

    freq2 = (work\_freq / 100) % 10;

    freq3 = (work\_freq / 10) % 10;

    freq4 = work\_freq % 10;

}

void frequency\_set(uint16\_t \*work\_freq, int freq1, int freq2, int freq3, int freq4){

    \*work\_freq = freq1 \* 1000 + freq2 \* 100 + freq3 \* 10 + freq4;

}

bool checkFreq(int freq1, int freq2, int freq3, int freq4){

    int freq = freq1 \* 1000 + freq2 \* 100 + freq3 \* 10 + freq4;

    if(freq >= 880 && freq <= 1080) return false;

    else return true;

}

void itoafrequency (uint8\_t freq1, int freq2, int freq3, int freq4, unsigned char \*\*instrfrequency1, unsigned char \*\*instrfrequency2, unsigned char \*\*instrfrequency3, unsigned char \*\*instrfrequency4)//将后台opmode工作模式转换为lcd屏幕数据

{

    freq\_1[0] = '0' + freq1;

    freq\_2[0] = '0' + freq2;

    freq\_3[0] = '0' + freq3;

    freq\_4[0] = '0' + freq4;

    \*instrfrequency1 = freq\_1;

    \*instrfrequency2 = freq\_2;

    \*instrfrequency3 = freq\_3;

    \*instrfrequency4 = freq\_4;

}

void itoatempreture(uint8\_t tempreture, unsigned char \*\*instrfrequency1, unsigned char \*\*instrfrequency2, unsigned char \*\*instrfrequency3, unsigned char \*\*instrfrequency4, unsigned char \*\*instrfrequency5, unsigned char \*\*instrfrequency6){

    temp1\_1[0] = tempreture / 100 + '0';

    \*instrfrequency1 = temp1\_1;

    temp1\_2[0] = (tempreture / 10) % 10 + '0';

    \*instrfrequency2 = temp1\_2;

    temp1\_3[0] = tempreture % 10 + '0';

    \*instrfrequency3 = temp1\_3;

    temp2\_1[0] = tempreture / 100 + '0';

    \*instrfrequency4 = temp2\_1;

    temp2\_2[0] = (tempreture / 10) % 10 + '0';

    \*instrfrequency5 = temp2\_2;

    temp2\_3[0] = tempreture % 10 + '0';

    \*instrfrequency6 = temp2\_3;

}

void itoavoltage(uint32\_t ui32ADC0Voltage, unsigned char \*\*instrfrequency1, unsigned char \*\*instrfrequency2, unsigned char \*\*instrfrequency3, unsigned char \*\*instrfrequency4)

{

    volt1[0] = ui32ADC0Voltage / 1000;

    volt2[0] = (ui32ADC0Voltage / 100) % 10;

    volt3[0] = (ui32ADC0Voltage / 10) % 10;

    volt4[0] = ui32ADC0Voltage  10;

}

void ENTER\_detect(void)

{

    switch(key\_ENTER\_state)

    {

        case 0:

            if(key\_code==5)

            {key\_ENTER\_state=1; key\_ENTER\_flag=1;} break;

        case 1:

            if (key\_code!=5)

            {key\_ENTER\_state=0;} break;

        default: {key\_ENTER\_state=0;} break;

    }

}

void DOWN\_detect(void)

{

    if (key\_code==6) ///////////////////     DOWN

    {

        key\_DOWN\_prestate=key\_DOWN\_state;

        key\_DOWN\_state=0;

        if (key\_DOWN\_prestate==1) key\_DOWN\_flag=1;

    }

    else

    {

        key\_DOWN\_prestate = key\_DOWN\_state;

        key\_DOWN\_state=1;

    }

}

void UP\_detect(void)

{

    switch(key\_UP\_state)

    {

        case 0:

            if(key\_code==4)

            {key\_UP\_state=1; key\_UP\_flag=1;} break;

        case 1:

            if (key\_code!=4)

            {key\_UP\_state=0;} break;

        default: {key\_UP\_state=0;} break;

    }

}

void INCREASE\_detect(void)

{

    if (key\_code==2) ///////////////////     INCREASE

    {

        key\_INCREASE\_prestate=key\_INCREASE\_state;

        key\_INCREASE\_state=0;

        if (key\_INCREASE\_prestate==1)

        {   key\_INCREASE\_flag=1;    key\_INCREASE\_timer =0;  }

        else if (key\_INCREASE\_prestate==0)

        {

            if  (++key\_INCREASE\_timer>=KEYTMR\_OF)

            { key\_INCREASE\_flag=1; key\_INCREASE\_timer=0;}

        }

    }

    else

    {

        key\_INCREASE\_prestate = key\_INCREASE\_state;

        key\_INCREASE\_state=1;

        key\_INCREASE\_timer=0;

    }

}

void DECREASE\_detect(void)

{

    if (key\_code==8) ///////////////////     DECREASE

    {

        key\_DECREASE\_prestate=key\_DECREASE\_state;

        key\_DECREASE\_state=0;

        if (key\_DECREASE\_prestate==1)

        {   key\_DECREASE\_flag=1;    key\_DECREASE\_timer =0;  }

        else if (key\_DECREASE\_prestate==0)

        {

            if  (++key\_DECREASE\_timer>=KEYTMR\_OF)

            { key\_DECREASE\_flag=1; key\_DECREASE\_timer=0;}

        }

    }

    else

    {

        key\_DECREASE\_prestate = key\_DECREASE\_state;

        key\_DECREASE\_state=1;

        key\_DECREASE\_timer=0;

    }

}

void display\_ui\_act(unsigned int i)

{

    unsigned int j=0;

    clear\_screen();

    for (j=0;j<act[i]->num;j++)

    {

        display\_GB2312\_string(act[i]->x[j],(act[i]->y[j]-1)\*8+1,act[i]->str[j],act[i]->inverse[j]);

    }

}

void initial\_act(void)

{

//State: a0

    a0.num=17;

    a0.str[0]=a0\_s0; a0.x[0]=1;  a0.y[0]=1;  a0.inverse[0]=0;

    itoavoltage(ui32ADC0Voltage, &a0\_s1, &a0\_s3, &a0\_s4, &a0\_s5);

    itoatempreture(tempreture, &a0\_s8, &a0\_s9, &a0\_s11, &a0\_s12, &a0\_s13, &a0\_s15);

    a0.str[1]=a0\_s1; a0.x[1]=1;  a0.y[1]=6;  a0.inverse[1]=0;

    a0.str[2]=a0\_s2; a0.x[2]=1;  a0.y[2]=7;  a0.inverse[2]=0;

    a0.str[3]=a0\_s3; a0.x[3]=1;  a0.y[3]=8;  a0.inverse[3]=0;

    a0.str[4]=a0\_s4; a0.x[4]=1;  a0.y[4]=9;  a0.inverse[4]=0;

    a0.str[5]=a0\_s5; a0.x[5]=1;  a0.y[5]=10;  a0.inverse[5]=0;

    a0.str[6]=a0\_s6; a0.x[6]=1;  a0.y[6]=11;  a0.inverse[6]=0;

    a0.str[7]=a0\_s7; a0.x[7]=3;  a0.y[7]=1;  a0.inverse[7]=0;

    a0.str[8]=a0\_s8; a0.x[8]=3;  a0.y[8]=6;  a0.inverse[8]=0;

    a0.str[9]=a0\_s9; a0.x[9]=3;  a0.y[9]=7;  a0.inverse[9]=0;

    a0.str[10]=a0\_s10; a0.x[10]=3;  a0.y[10]=8;  a0.inverse[10]=0;

    a0.str[11]=a0\_s11; a0.x[11]=3;  a0.y[11]=9;  a0.inverse[11]=0;

    a0.str[12]=a0\_s12; a0.x[12]=3;  a0.y[12]=11;  a0.inverse[12]=0;

    a0.str[13]=a0\_s13; a0.x[13]=3;  a0.y[13]=12;  a0.inverse[13]=0;

    a0.str[14]=a0\_s14; a0.x[14]=3;  a0.y[14]=13;  a0.inverse[14]=0;

    a0.str[15]=a0\_s15; a0.x[15]=3;  a0.y[15]=14;  a0.inverse[15]=0;

    a0.str[16]=a0\_s16; a0.x[16]=3;  a0.y[16]=15;  a0.inverse[16]=0;

//State: 1

    a1.num=8;

    frequency\_init(work\_freq);

    itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

    a1.str[0]=a1\_s0; a1.x[0]=1;  a1.y[0]=1;  a1.inverse[0]=0;

    a1.str[1]=a1\_s1; a1.x[1]=1;  a1.y[1]=6;  a1.inverse[1]=1;

    a1.str[2]=a1\_s2; a1.x[2]=1;  a1.y[2]=7;  a1.inverse[2]=0;

    a1.str[3]=a1\_s3; a1.x[3]=1;  a1.y[3]=8;  a1.inverse[3]=0;

    a1.str[4]=a1\_s4; a1.x[4]=1;  a1.y[4]=9;  a1.inverse[4]=0;

    a1.str[5]=a1\_s5; a1.x[5]=1;  a1.y[5]=10;  a1.inverse[5]=0;

    a1.str[6]=a1\_s6; a1.x[6]=1;  a1.y[6]=11;  a1.inverse[6]=0;

    a1.str[7]=a1\_s7; a1.x[7]=7;  a1.y[7]=13;  a1.inverse[7]=0;

//State: 2

    a2.num=1;

    a2.str[0]=a2\_s0; a2.x[0]=1;  a2.y[0]=1;  a2.inverse[0]=0;

//Output with act[0]

    act[0]=&a0;

    display\_ui\_act(0);

}

void ui\_proc0\_0(void)

{

    if(key\_code!=0)

    {

        key\_UP\_flag=0;key\_DOWN\_flag=0;key\_INCREASE\_flag=0;key\_DECREASE\_flag=0;key\_ENTER\_flag=0;

        act[0] = &a1;

        display\_ui\_act(0);

        act\_state = 1;

        ui\_state=001;

    }

    itoavoltage(ui32ADC0Voltage, &a0\_s1, &a0\_s3, &a0\_s4, &a0\_s5);

    itoatempreture(tempreture, &a0\_s8, &a0\_s9, &a0\_s11, &a0\_s12, &a0\_s13, &a0\_s15);

    display\_ui\_act(0);

}

void ui\_proc1\_001(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[1]=0;

        a1.inverse[7]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 007;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[1]=0;

        a1.inverse[2]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 002;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq1++;

        if (freq1>9) freq1=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[1] = a1\_s1;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq1--;

        if (freq1<0) freq1=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[1] = a1\_s1;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_002(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[2]=0;

        a1.inverse[1]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 001;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[2]=0;

        a1.inverse[3]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 003;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq2++;

        if (freq2>9) freq2=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[2] = a1\_s2;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq2--;

        if (freq2<0) freq2=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[2] = a1\_s2;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_003(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[3]=0;

        a1.inverse[2]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 002;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[3]=0;

        a1.inverse[5]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 005;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq3++;

        if (freq3>9) freq3=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[3] = a1\_s3;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq3--;

        if (freq3<0) freq3=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[3] = a1\_s3;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_005(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[5]=0;

        a1.inverse[3]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 003;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[5]=0;

        a1.inverse[7]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 007;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq4++;

        if (freq4>9) freq4=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[5] = a1\_s5;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq4--;

        if (freq4<0) freq4=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[5] = a1\_s5;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_007(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[7]=0;

        a1.inverse[5]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 005;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[7]=0;

        a1.inverse[1]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 001;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

        a1.inverse[7]=0;

        a1.inverse[1]=1;

        if(checkFreq(freq1, freq2, freq3, freq4)){

            act\_state = 2;

            ui\_state = 0;

            NOKEY\_clock5s\_flag = 0;

            act[0] = &a2;

            display\_ui\_act(0);

        } else {

            frequency\_set(&work\_freq, freq1, freq2, freq3, freq4);

            itoatempreture(tempreture, &a0\_s1, &a0\_s2, &a0\_s4);

            act[0] = &a0;

            display\_ui\_act(0);

            act\_state = 10;

            ui\_state = 0;

        }

    }

}

void ui\_proc2\_0(void)

{

    if(key\_code!=0)

    {

        key\_UP\_flag=0;key\_DOWN\_flag=0;key\_INCREASE\_flag=0;key\_DECREASE\_flag=0;key\_ENTER\_flag=0;

    }

    if(NOKEY\_clock5s\_flag==1)

    {

        act[0] = &a1;

        display\_ui\_act(0);

        act\_state = 1;

        ui\_state = 001;

    }

}

void delay60ms()

{

    if(key\_code!=0)

    {

        key\_UP\_flag=0;key\_DOWN\_flag=0;key\_INCREASE\_flag=0;key\_DECREASE\_flag=0;key\_ENTER\_flag=0;

    }

    if(clock100ms\_flag==1)

    {

        act[0] = &a0;

        display\_ui\_act(0);

        act\_state = 0;

        ui\_state = 0;

    }

}

void ui\_state\_proc(unsigned int ui\_state)

{

    switch (act\_state)

        {

            case 0:

                switch (ui\_state){

                    case 0: ui\_proc0\_0(); break;

                    default: break;

                }

            break;

            case 1: switch (ui\_state){

                    case 001: ui\_proc1\_001(); break;

                    case 002: ui\_proc1\_002(); break;

                    case 003: ui\_proc1\_003(); break;

                    case 005: ui\_proc1\_005(); break;

                    case 007: ui\_proc1\_007(); break;

                    default: break;

                }

            break;

            case 2: switch (ui\_state){

                    case 0: ui\_proc2\_0(); break;

                    default: break;

                }

            break;

            case 10: delay60ms(); break;

            default: break;

        }

}

## 接收端程序核心代码清单

#include <stdint.h>

#include <stdbool.h>

#include "define.h" // 定义NCPDT

#include "inc/hw\_memmap.h" // 基址宏定义

#include "inc/hw\_ints.h"

#include "inc/hw\_types.h" // 数据类型宏定义，寄存器访问函数

#include "driverlib/debug.h" // 调试用

#include "driverlib/gpio.h" // 通用IO口宏定义

#include "driverlib/pin\_map.h" // TM4C系列MCU外围设备管脚宏定义

#include "driverlib/sysctl.h" // 系统控制定义

#include "driverlib/systick.h" // SysTick Driver 原型

#include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型

#include "driverlib/uart.h" // 与UART有关的宏定义和函数原型

#include "driverlib/timer.h" // 与Timer有关的函数原型

#include "driverlib/pwm.h" // 与Timer有关的函数原型

#include "tm1638.h" // 与控制TM1638芯片有关的函数

#include "JLX12864\_2.c"

#include "modes.c"

#include "ADC.h"

#include "PWM.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 宏定义

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define SYSTICK\_FREQUENCY 50 // SysTick频率为50Hz，即循环定时周期20ms

#define FREQUENCY\_MIN 300 // 有效频率最小值，根据实际需要设置

#define FREQUENCY\_MAX 4000 // 有效频率最大值，根据实际需要设置

#define V\_T100ms 5 // 0.1s软件定时器溢出值，5个20ms

#define V\_T400ms 20 // 400ms软件定时器溢出值，20个20ms

#define V\_T500ms 25 // 0.5s软件定时器溢出值，25个20ms

#define V\_T1500ms 3 // 1.5s软件定时器溢出值，设为测温间隔 改为60ms

#define V\_T2s 100 // 2s软件定时器溢出值，100个20ms

#define V\_T5s 250 // 5s软件定时器溢出值

#define V\_T10s 500 // 10s软件定时器溢出值

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型声明

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GPIOInit(void); // GPIO初始化

void SysTickInit(void); // 设置SysTick中断

void DevicesInit(void); // MCU器件初始化，注：会调用上述函数

void UARTInit(void); // UART初始化

void UARTStringPut(uint32\_t ui32Base,const char \*cMessage);// 向UART发送字符串。发送到缓冲区

void UARTStringGet(uint32\_t ui32Base, char\* s);//读取字符串

void UART6\_Handler(void);//中断处理程序

void passFreq(uint8\_t\* f);

void Timer0Init(void); // Timer0初始化

void Timer1Init(void);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 变量定义

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 软件定时器计数

uint8\_t clock400ms = 0;

uint8\_t clock100ms = 0;

uint8\_t clock500ms = 0;

uint8\_t clock1500ms = 0;

uint8\_t clock2s = 0;

uint8\_t clock5s = 0;

uint16\_t clock10s = 0;

// 软件定时器溢出标志

uint8\_t clock100ms\_flag = 0;

uint8\_t clock500ms\_flag = 0;

uint8\_t clock1500ms\_flag = 0;

uint8\_t clock2s\_flag = 1; //初始2s标志位为1

uint8\_t clock5s\_flag = 0;

uint8\_t clock10s\_flag = 0;

uint8\_t clock400ms\_flag = 0;

bool if\_count\_2s = false; // 是否开始计数

//LCD屏幕状态机

uint8\_t lcd\_act = 0; //指示当前状态

uint8\_t lcd\_subact = 0; //加入子状态

bool if\_refresh = true; // 赋值为true便于初始化时必定切入模式0

// 测试用计数器

uint32\_t test\_counter = 0;

// 8位数码管显示的数字或字母符号

// 注：板上数码位从左到右序号排列为4、5、6、7、0、1、2、3

uint8\_t digit[8]={' ',' ',' ','H',' ',' ',' ',' '};

// 8位小数点 1亮 0灭

// 注：板上数码位小数点从左到右序号排列为4、5、6、7、0、1、2、3

uint8\_t pnt = 0x00;

// 8个LED指示灯状态，0灭，1亮

// 注：板上指示灯从左到右序号排列为7、6、5、4、3、2、1、0

// 对应元件LED8、LED7、LED6、LED5、LED4、LED3、LED2、LED1

uint8\_t led[] = {1, 1, 1, 1, 1, 1, 1, 0};

// 存储接受发送字符串用

char str[50] = {'\0'};

// 当前按键值

uint8\_t key\_code = 0;

// 系统时钟频率

uint32\_t ui32SysClock;

uint32\_t g\_ui32SysClock;

// 面板参数变量

uint8\_t\* letter[3]={"A", "B", "C"};

uint8\_t\* number[10] = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"};

uint16\_t voltage = 0; //保存电压值

uint8\_t\* v\_passin[4] = {0}; //测量电压显示

uint16\_t tempre = 0;//保存温度值

uint8\_t\* t\_passin[3] = {0}; //测量温度显示

uint8\_t freq[4] = {1, 0, 1, 0};

uint8\_t freq\_temp[4] = {0}; // 用于保存频率参数，若不合法则返回修改前参数

uint8\_t\* f\_passin[4] = {0}; //显示频率

// 定义函数指针数组便于切换模式

void (\*func[3][5])(uint8\_t\* [], uint8\_t\* [], uint8\_t\* []) = {0};

// AIN2(PE1) ADC采样值[0-4095]

uint32\_t ui32ADC0Value; //PE1为输入引脚

// AIN2电压值(单位为0.01V) [0.00-3.30]

uint32\_t ui32ADC0Voltage;//电压值与ADC采样值线性对应

// 1s计数结束标志

volatile uint8\_t g\_ui8INTStatus = 0;

// 保存上一次TIMER0边沿计数值

volatile uint32\_t g\_ui32TPreCount = 0;

// 保存本次TIMER0边沿计数值

volatile uint32\_t g\_ui32TCurCount = 0;

// 系统当前功能状态

bool system\_state = true;

bool record = true;

bool play\_record = false;

bool start = false;

int record\_melody[400] = {0};

int ii = 0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 主程序

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main(void)

{

uint8\_t temp,i;

uint32\_t ui32Freq; // 记录测量的方波频率

DevicesInit(); // MCU器件初始化

while (clock100ms < 3); // 延时>60ms,等待TM1638上电完成

TM1638\_Init(); // 初始化TM1638

initial\_lcd();

clear\_screen();

//将函数装入函数指针数组中

func[0][0] = mode0\_0;

func[1][0] = mode1\_0;

func[1][1] = mode1\_1;

func[1][2] = mode1\_2;

func[1][3] = mode1\_3;

func[1][4] = mode1\_4;

func[2][0] = mode2\_0;

// 初始化温度为0℃

t\_passin[0] = number[tempre / 100];

t\_passin[1] = number[(tempre / 10) % 10];

t\_passin[2] = number[tempre % 10];

// 初始化频率为100.0MHz

passFreq(freq);

UARTStringPut(UART6\_BASE, "AT+VOL=15");

while (1)

{

// 采样电压

if (clock400ms\_flag == 1) // 检查400ms秒定时是否到

{

clock400ms\_flag = 0;

ui32ADC0Value = ADC\_Sample(); // 采样，一次采一个，400ms采样一次

// digit[4] = ui32ADC0Value / 1000; // 显示ADC采样值千位数

// digit[5] = ui32ADC0Value / 100 % 10; // 显示ADC采样值百位数

// digit[6] = ui32ADC0Value / 10 % 10; // 显示ADC采样值十位数

// digit[7] = ui32ADC0Value % 10; // 显示ADC采样值个位数

ui32ADC0Voltage = ui32ADC0Value \* 3300 / 4095; //采样的值线性对应电压

// digit[0] = (ui32ADC0Voltage / 1000) % 10;

// digit[1] = (ui32ADC0Voltage / 100) % 10; // 显示电压值个位数

// digit[2] = (ui32ADC0Voltage / 10) % 10; // 显示电压值十分位数

// digit[3] = ui32ADC0Voltage % 10; // 显示电压值百分位数

v\_passin[0] = number[(ui32ADC0Voltage / 1000) % 10]; // 显示电压值个位数

v\_passin[1] = number[(ui32ADC0Voltage / 100) % 10]; // 显示电压值十分位数

v\_passin[2] = number[(ui32ADC0Voltage / 10) % 10]; // 显示电压值百分位数

v\_passin[3] = number[(ui32ADC0Voltage) % 10]; // 显示电压值千分位数

if (lcd\_act == 0 && lcd\_subact == 0) if\_refresh = true; //处于模式0则刷新，为了更新频率值

}

if (clock500ms\_flag == 1) // 检查0.5秒定时是否到

{

clock500ms\_flag = 0;

// 8个指示灯以走马灯方式，每0.5秒向右（循环）移动一格

temp = led[0];

for (i = 0; i < 7; i++) led[i] = led[i + 1];

led[7] = temp;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 计时测量频率 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*clock1500ms\_flag == 1

if (start == false){ // 检查1.5秒定时是否到，启动Timer0A

TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // Timer1A 超时中断使能

IntEnable(INT\_TIMER1A); // 开启 TIMER1A 中断源

TimerEnable(TIMER1\_BASE, TIMER\_A); // TIMER1 开始计时

TimerEnable(TIMER0\_BASE, TIMER\_A); // TIMER0 开始计数

start = true;

}

if(g\_ui8INTStatus == 1) // 1s定时结束，开始计算频率

{

start = false;

g\_ui8INTStatus = 0;

// if (g\_ui32TCurCount >= g\_ui32TPreCount)

// ui32Freq = g\_ui32TCurCount - g\_ui32TPreCount;

// else {

// ui32Freq = 0xFFFFFFFF - g\_ui32TPreCount;

// ui32Freq = ui32Freq + g\_ui32TCurCount + 1;

// }

ui32Freq = g\_ui32TCurCount >= g\_ui32TPreCount ? //计算频率，正向/循环

(g\_ui32TCurCount - g\_ui32TPreCount) : (0xFFFF - g\_ui32TPreCount + g\_ui32TCurCount + 1);

ui32Freq \*= 20;

if ((ui32Freq >= FREQUENCY\_MIN) && (ui32Freq <= FREQUENCY\_MAX)) { //频率是否处于合法范围

digit[4] = ui32Freq / 1000 % 10; // 计算千位数

digit[5] = ui32Freq / 100 % 10; // 计算百位数

digit[6] = ui32Freq / 10 % 10; // 计算十位数

digit[7] = ui32Freq % 10; // 计算个位数

tempre = (ui32Freq > 299) ? (400 \* (ui32Freq - 300) / 3700) : 0; // 解码，根据发射频率计算得到温度

t\_passin[0] = number[tempre / 100]; // 传递参数至显示屏

t\_passin[1] = number[(tempre / 10) % 10];

t\_passin[2] = number[tempre % 10];

if (lcd\_act == 0 && lcd\_subact == 0) if\_refresh = true; // 若处于初始界面，则刷新显示屏

} else // 测量频率超过指定范围

{

digit[4] = 'E';

digit[5] = 'R';

digit[6] = 'R';

digit[7] = ' ';

t\_passin[0] = "N"; // 传递参数至显示屏

t\_passin[1] = "a";

t\_passin[2] = "N";

if (lcd\_act == 0 && lcd\_subact == 0) if\_refresh = true; // 若处于初始界面，则刷新显示屏

}

if(play\_record == false){

if(system\_state){

PWMStart(ui32Freq);

} else {

PWMStop();

}

} else {

PWMStart(record\_melody[ii]);

++ii;

if(ii >= 400){

ii = 0;

PWMStop();

SysCtlDelay((ui32SysClock \* 2 / 3) \* 1);

play\_record = false;

}

}

if(record){

if(ii < 400){

record\_melody[ii] = ui32Freq;

++ii;

} else {

ii = 0;

PWMStop();

SysCtlDelay((ui32SysClock \* 2 / 3) \* 1);

play\_record = true;

record = false;

}

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 刷新屏幕内容

if (if\_refresh || (if\_count\_2s && clock2s\_flag))

{

if (if\_count\_2s && clock2s\_flag){ //处于警示界面，2s计时器触发，两秒后回到ACT1

lcd\_act = 1;

lcd\_subact = 0;

if\_count\_2s = false;

}

if\_refresh = false;

for (i=0; i<4; ++i) f\_passin[i] = number[freq\_temp[i]]; // 传递显示频率

TEST\_H;

func[lcd\_act][lcd\_subact](v\_passin, t\_passin, f\_passin); // 根据不同模式显示不同界面

if (lcd\_act == 2 && lcd\_subact == 0) {if\_count\_2s = true; clock2s = 0; clock2s\_flag = false;} // 进入警示界面则开始计时

else if\_count\_2s = false;

TEST\_L;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void GPIOInit(void)

// 函数功能：GPIO初始化。使能PortK，设置PK4,PK5为输出；使能PortM，设置PM0为输出。

// （PK4连接TM1638的STB，PK5连接TM1638的DIO，PM0连接TM1638的CLK）

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GPIOInit(void)

{

//配置TM1638芯片管脚

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOK); // 使能端口 K

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOK)){}; // 等待端口 K准备完毕

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOM); // 使能端口 M

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOM)){}; // 等待端口 M准备完毕

// 设置端口 K的第4,5位（PK4,PK5）为输出引脚 PK4-STB PK5-DIO

GPIOPinTypeGPIOOutput(GPIO\_PORTK\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_5);

// 设置端口 M的第0位（PM0）为输出引脚 PM0-CLK

GPIOPinTypeGPIOOutput(GPIO\_PORTM\_BASE, GPIO\_PIN\_0);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：SysTickInit(void)

// 函数功能：设置SysTick中断

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SysTickInit(void)

{

SysTickPeriodSet(ui32SysClock/SYSTICK\_FREQUENCY); // 设置心跳节拍,定时周期20ms

SysTickEnable(); // SysTick使能

SysTickIntEnable(); // SysTick中断允许

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void DevicesInit(void)

// 函数功能：CU器件初始化，包括系统时钟设置、GPIO初始化和SysTick中断设置

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void DevicesInit(void)

{

// 使用外部25MHz主时钟源，经过PLL，然后分频为20MHz

ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ |SYSCTL\_OSC\_MAIN |

SYSCTL\_USE\_PLL |SYSCTL\_CFG\_VCO\_480),

20000000);

g\_ui32SysClock = ui32SysClock;;

GPIOInit(); // GPIO初始化

ADCInit(); // ADC初始化

Timer0Init(); // Timer0初始化

Timer1Init();

UARTInit(); // UART初始化

SysTickInit(); // 设置SysTick中断

IntMasterEnable(); // 总中断允许

IntPrioritySet(INT\_TIMER1A,0x00); // 设置INT\_TIMER0A最高优先级

IntPrioritySet(INT\_TIMER0A,0x01); // 设置INT\_TIMER0A最高优先级

IntPrioritySet(FAULT\_SYSTICK,0xe0); // 设置SYSTICK优先级低于INT\_TIMER0A的优先级

PWMInit();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void SysTick\_Handler(void)

// 函数功能：SysTick中断服务程序

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SysTick\_Handler(void) // 定时周期为20ms

{

uint8\_t key\_code\_pre = key\_code; // 保存前一次按键状态

uint8\_t i = 0;

// 0.1秒钟软定时器计数

if (++clock100ms >= V\_T100ms)

{

clock100ms\_flag = 1; // 当0.1秒到时，溢出标志置1

clock100ms = 0;

}

// 0.5秒钟软定时器计数

if (++clock500ms >= V\_T500ms)

{

clock500ms\_flag = 1; // 当0.5秒到时，溢出标志置1

clock500ms = 0;

}

// 1.5秒钟软定时器计数

if (++clock1500ms >= V\_T1500ms)

{

clock1500ms\_flag = 1; // 当1.5秒到时，溢出标志置1

clock1500ms = 0;

}

// 2秒钟软定时器计数

if (++clock2s >= V\_T2s)

{

clock2s\_flag = 1; // 当2秒到时，溢出标志置1

clock2s = 0;

}

// 5秒钟软定时器计数

if (++clock5s >= V\_T5s){

clock5s\_flag = 1;

clock5s = 0;

}

// 10秒钟软定时器计数

if (++clock10s >= V\_T10s){

clock10s\_flag = 1;

clock10s = 0;

}

if (++clock400ms >= V\_T400ms)

{

clock400ms\_flag = 1; // 当40ms到时，溢出标志置1

clock400ms = 0;

}

// 刷新全部数码管和LED指示灯

TM1638\_RefreshDIGIandLED(digit, pnt, led);

// 检查当前键盘输入，0代表无键操作，1-9表示有对应按键

key\_code = TM1638\_Readkeyboard();

if(key\_code\_pre != 1 && key\_code == 1){

system\_state = !system\_state;

}

if(key\_code\_pre != 3 && key\_code == 3){

record = !record;

ii = 0;

}

// 面板参数切换

if (key\_code\_pre == 0 && key\_code != 0 && lcd\_act == 0 && lcd\_subact == 0){

lcd\_act = 1;

lcd\_subact = 0;

for (i =0; i<4; ++i){

freq\_temp[i] = freq[i];

} // 保存修改前参数

if\_refresh = true; //刷新模式

return;

}

if (key\_code\_pre != 8 && key\_code == 8){

if (lcd\_act == 1 && lcd\_subact == 0){

freq\_temp[0] = (freq\_temp[0] + 9) % 10;

if\_refresh = true;

} else if (lcd\_act == 1 && lcd\_subact == 1){

freq\_temp[1] = (freq\_temp[1] + 9) % 10;

if\_refresh = true;

} else if (lcd\_act == 1 && lcd\_subact == 2){

freq\_temp[2] = (freq\_temp[2] + 9) % 10;

if\_refresh = true;

} else if (lcd\_act == 1 && lcd\_subact == 3){

freq\_temp[3] = (freq\_temp[3] + 9) % 10;

if\_refresh = true;

}

}

if (key\_code\_pre != 2 && key\_code == 2){

if (lcd\_act == 1 && lcd\_subact == 0){

freq\_temp[0] = (freq\_temp[0] + 1) % 10;

if\_refresh = true;

} else if (lcd\_act == 1 && lcd\_subact == 1){

freq\_temp[1] = (freq\_temp[1] + 1) % 10;

if\_refresh = true;

} else if (lcd\_act == 1 && lcd\_subact == 2){

freq\_temp[2] = (freq\_temp[2] + 1) % 10;

if\_refresh = true;

} else if (lcd\_act == 1 && lcd\_subact == 3){

freq\_temp[3] = (freq\_temp[3] + 1) % 10;

if\_refresh = true;

} // 可替换成switch, 但懒得换了

}

if (key\_code\_pre != 4 && key\_code == 4){

switch (lcd\_act) {

case 1:

lcd\_subact = (lcd\_subact + 4) % 5;

if\_refresh = true;

break;

} // 可替换成if, 但懒得换了

}

if (key\_code\_pre != 6 && key\_code == 6){

switch (lcd\_act) {

case 1:

lcd\_subact = (lcd\_subact + 1) % 5;

if\_refresh = true;

break;

}

}

if (key\_code\_pre != 5 && key\_code == 5 && lcd\_act == 1 && lcd\_subact == 4){

uint16\_t f\_temp = 1000 \* freq\_temp[0] + 100 \* freq\_temp[1] + 10 \* freq\_temp[2] + freq\_temp[3];

if (f\_temp >= 880 && f\_temp <= 1080){ // 参数合法

for (i =0; i<4; ++i){

freq[i] = freq\_temp[i]; // 更新参数

}

passFreq(freq); // 参数传至FM接收模块

lcd\_act = 0;

lcd\_subact = 0;

if\_refresh = true;

clear\_screen();

}

else { // 参数不合法

for (i =0; i<4; ++i){

freq\_temp[i] = freq[i]; // 还原参数

}

lcd\_act = 2;

lcd\_subact = 0;

if\_refresh = true;

}

}

}

// UART部分

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void UARTInit(void)

// 函数功能：UART初始化。使能UART0，设置PA0,PA1为UART0 RX,TX引脚；

// 设置波特率及帧格式。

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void UARTInit(void)

{

// 引脚配置

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART6); // 使能UART6模块

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOP); // 使能端口 P

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOP)); // 等待端口 P准备完毕

GPIOPinConfigure(GPIO\_PP0\_U6RX); // 设置PP0为UART6 RX引脚，RX为接收引脚

GPIOPinConfigure(GPIO\_PP1\_U6TX); // 设置PP1为UART6 TX引脚，TX为发送引脚

// 设置端口 A的第0,1位（PA0,PA1）为UART引脚

GPIOPinTypeUART(GPIO\_PORTP\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// 波特率及帧格式设置

UARTConfigSetExpClk(UART6\_BASE,

ui32SysClock,

9600, // 波特率：9600

(UART\_CONFIG\_WLEN\_8 | // 数据位：8

UART\_CONFIG\_STOP\_ONE | // 停止位：1

UART\_CONFIG\_PAR\_NONE)); // 校验位：无

IntEnable(INT\_UART6); // UART6 中断允许

UARTIntEnable(UART6\_BASE, UART\_INT\_RX | UART\_INT\_RT);// 使能 UART6 RX,RT 中断。RX为接收中断，RT为接收超时中断

//BASE表示段基址地址。用这个地址来指向对应模块。用来发射或接受等等

// 引脚配置

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0); // 使能UART0模块

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA); // 使能端口 A

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOA)); // 等待端口 A准备完毕

GPIOPinConfigure(GPIO\_PA0\_U0RX); // 设置PA0为UART0 RX引脚

GPIOPinConfigure(GPIO\_PA1\_U0TX); // 设置PA1为UART0 TX引脚

// 设置端口 A的第0,1位（PA0,PA1）为UART引脚

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

// 波特率及帧格式设置

UARTConfigSetExpClk(UART0\_BASE,

ui32SysClock,

115200, // 波特率：115200

(UART\_CONFIG\_WLEN\_8 | // 数据位：8

UART\_CONFIG\_STOP\_ONE | // 停止位：1

UART\_CONFIG\_PAR\_NONE)); // 校验位：无

// 初始化完成后向PC端发送"Hello, 2A!"字符串

UARTStringPut(UART0\_BASE, (const char \*)"Hello, 2A!");

IntEnable(INT\_UART0); // UART0 中断允许

UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT);// 使能 UART0 RX,RT 中断

}

void UART6\_Handler(void) //UART中断服务程序

{

uint32\_t uart6\_int\_status;

uart6\_int\_status = UARTIntStatus(UART6\_BASE, true); // 取中断状态

UARTIntClear(UART6\_BASE, uart6\_int\_status); // 清中断状态

UARTStringGet(UART6\_BASE, str); //读取UART6发送的字符

UARTStringPut(UART0\_BASE, str); //发送到缓冲区等待UART0接收

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void UARTStringPut(uint32\_t ui32Base,const char \*cMessage)

// 函数功能：向UART模块发送字符串

// 函数参数：ui32Base：UART模块

// cMessage：待发送字符串

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void UARTStringPut(uint32\_t ui32Base,const char \*cMessage)

{

while(\*cMessage != '\0') {

UARTCharPut(ui32Base, \*(cMessage++));

SysCtlDelay(1 \* ( ui32SysClock / 3000)); //每读1字符延时1ms

}

// UARTCharPut(ui32Base, '\r');

// UARTCharPut(ui32Base, '\n');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void UARTStringGet(uint32\_t ui32Base,const char \*cMessage)

// 函数功能：UART模块接收字符串

// 函数参数：ui32Base：UART模块

// str：待接收字符串

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void UARTStringGet(uint32\_t ui32Base, char\* s) {

// char\* temp = s;

while (UARTCharsAvail(ui32Base)) // 重复从接收FIF0读取字符

{

\*s = (char) UARTCharGetNonBlocking(ui32Base); // 读入一个字符

++s;

SysCtlDelay(1 \* ( ui32SysClock / 3000)); //每读1字符延时1ms

}

\*s = '\0';

// UARTStringPut(UART0\_BASE, (const char \*)"Get String:");

// UARTStringPut(UART0\_BASE, temp);

}

void passFreq(uint8\_t\* f){ //根据频率值设置接收频率

UARTStringPut(UART6\_BASE, "AT+FREQ=");

if (f[0] == 0){

UARTCharPut(UART6\_BASE,f[1]+'0');

UARTCharPut(UART6\_BASE,f[2]+'0');

UARTCharPut(UART6\_BASE,f[3]+'0');

}

else {

UARTCharPut(UART6\_BASE,f[0]+'0');

UARTCharPut(UART6\_BASE,f[1]+'0');

UARTCharPut(UART6\_BASE,f[2]+'0');

UARTCharPut(UART6\_BASE,f[3]+'0');

}

}

// 频率测量部分

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void TIMER1A\_Handler(void)

// 函数功能：Timer1A中断服务程序，记录捕获方波上升沿时定时器（TIMER0）的计数值

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void TIMER1A\_Handler(void)

{

TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // 清除中断标志

g\_ui32TPreCount = g\_ui32TCurCount; // 保存上一次TIMER0边沿计数值

g\_ui32TCurCount = TimerValueGet(TIMER0\_BASE, TIMER\_A); // 读取TIMER0边沿计数值

TimerDisable(TIMER0\_BASE, TIMER\_A); // 停止TIMER0边沿计数

g\_ui8INTStatus = 1; // 50ms计数完成

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void Timer1Init(void)

// 函数功能：设置Timer1为一次性定时器，定时周期为1s

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Timer1Init(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1); // TIMER1 使能

TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_ONE\_SHOT); // 设置为 32 位 1次 定时器

TimerLoadSet(TIMER1\_BASE, TIMER\_A, (ui32SysClock / 20) - 1); // TIMER1A装载计数值50ms

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// 函数原型：void Timer0Init(void)

// 函数功能：设置Timer0为输入边沿（上升沿）计数模式，T0CCP0(PL4)为捕获引脚

// 函数参数：无

// 函数返回值：无

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Timer0Init(void)

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0); // 使能TIMER0

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOL); // 使能GPIOL

GPIOPinConfigure(GPIO\_PL4\_T0CCP0); // 配置引脚复用

GPIOPinTypeTimer(GPIO\_PORTL\_BASE, GPIO\_PIN\_4); // 引脚映射

GPIOPadConfigSet(GPIO\_PORTL\_BASE, GPIO\_PIN\_4, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU); // 将引脚弱上拉

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_SPLIT\_PAIR | TIMER\_CFG\_A\_CAP\_COUNT\_UP); // 半长定时器，增计数

//TimerPrescaleSet(TIMER0\_BASE, TIMER\_A, 255); // 预分频256

TimerControlEvent(TIMER0\_BASE, TIMER\_A, TIMER\_EVENT\_POS\_EDGE); // 初始化配置为捕捉上升沿

}

unsigned char temp1[] = "1";

unsigned char temp2[] = "2";

unsigned char temp3[] = "3";

unsigned char freq\_1[] = "1";

unsigned char freq\_2[] = "1";

unsigned char freq\_3[] = "1";

unsigned char freq\_4[] = "1";

int freq1 = 1;

int freq2 = 1;

int freq3 = 1;

int freq4 = 1;

struct struct\_act

{

    unsigned char num;

    unsigned char \*str[20];

    unsigned char x[20],y[20],inverse[20];

} ;

//////////////////////////////

//State0

struct struct\_act a0;

unsigned char a0\_s0[] = "温度:";

unsigned char \*a0\_s1;

unsigned char \*a0\_s2;

unsigned char a0\_s3[] = ".";

unsigned char \*a0\_s4;

unsigned char a0\_s5[] = "℃";

//State1

struct struct\_act a1;

unsigned char a1\_s0[] = "载频:";

unsigned char \*a1\_s1;

unsigned char \*a1\_s2;

unsigned char \*a1\_s3;

unsigned char a1\_s4[] = ".";

unsigned char \*a1\_s5;

unsigned char a1\_s6[] = "MHz";

unsigned char a1\_s7[] = "确定";

//State2

struct struct\_act a2;

unsigned char a2\_s0[] = "载频超范围！";

struct struct\_act \*act[1];

//////////////////////////////

unsigned int act\_state = 0;

unsigned int ui\_state=0;  //状态号

unsigned int key\_ENTER\_state=0;

unsigned int key\_ENTER\_prestate=0;

unsigned int ENTER\_key\_timer=0;

unsigned int key\_ENTER\_flag=0;

unsigned int key\_DOWN\_state=0;

unsigned int key\_DOWN\_prestate=0;

unsigned int key\_DOWN\_timer=0;

unsigned int key\_DOWN\_flag=0;

unsigned int key\_UP\_state=0;

unsigned int key\_UP\_prestate=0;

unsigned int key\_UP\_timer=0;

unsigned int key\_UP\_flag=0;

unsigned int key\_INCREASE\_state=0;

unsigned int key\_INCREASE\_prestate=0;

unsigned int key\_INCREASE\_timer=0;

unsigned int key\_INCREASE\_flag=0;

unsigned int key\_DECREASE\_state=0;

unsigned int key\_DECREASE\_prestate=0;

unsigned int key\_DECREASE\_timer=0;

unsigned int key\_DECREASE\_flag=0;

void frequency\_init(uint16\_t work\_freq){

    freq1 = work\_freq / 1000;

    freq2 = (work\_freq / 100) % 10;

    freq3 = (work\_freq / 10) % 10;

    freq4 = work\_freq % 10;

}

void frequency\_set(uint16\_t \*work\_freq, int freq1, int freq2, int freq3, int freq4){

    \*work\_freq = freq1 \* 1000 + freq2 \* 100 + freq3 \* 10 + freq4;

}

bool checkFreq(int freq1, int freq2, int freq3, int freq4){

    int freq = freq1 \* 1000 + freq2 \* 100 + freq3 \* 10 + freq4;

    if(freq >= 880 && freq <= 1080) return false;

    else return true;

}

void itoafrequency (uint8\_t freq1, int freq2, int freq3, int freq4, unsigned char \*\*instrfrequency1, unsigned char \*\*instrfrequency2, unsigned char \*\*instrfrequency3, unsigned char \*\*instrfrequency4)//将后台opmode工作模式转换为lcd屏幕数据

{

    freq\_1[0] = '0' + freq1;

    freq\_2[0] = '0' + freq2;

    freq\_3[0] = '0' + freq3;

    freq\_4[0] = '0' + freq4;

    \*instrfrequency1 = freq\_1;

    \*instrfrequency2 = freq\_2;

    \*instrfrequency3 = freq\_3;

    \*instrfrequency4 = freq\_4;

}

void itoatempreture(uint8\_t tempreture, unsigned char \*\*instrfrequency1, unsigned char \*\*instrfrequency2, unsigned char \*\*instrfrequency3){

    temp1[0] = tempreture / 100 + '0';

    \*instrfrequency1 = temp1;

    temp2[0] = (tempreture / 10) % 10 + '0';

    \*instrfrequency2 = temp2;

    temp3[0] = tempreture % 10 + '0';

    \*instrfrequency3 = temp3;

}

void ENTER\_detect(void)

{

    switch(key\_ENTER\_state)

    {

        case 0:

            if(key\_code==5)

            {key\_ENTER\_state=1; key\_ENTER\_flag=1;} break;

        case 1:

            if (key\_code!=5)

            {key\_ENTER\_state=0;} break;

        default: {key\_ENTER\_state=0;} break;

    }

}

void DOWN\_detect(void)

{

    if (key\_code==6) ///////////////////     DOWN

    {

        key\_DOWN\_prestate=key\_DOWN\_state;

        key\_DOWN\_state=0;

        if (key\_DOWN\_prestate==1) key\_DOWN\_flag=1;

    }

    else

    {

        key\_DOWN\_prestate = key\_DOWN\_state;

        key\_DOWN\_state=1;

    }

}

void UP\_detect(void)

{

    switch(key\_UP\_state)

    {

        case 0:

            if(key\_code==4)

            {key\_UP\_state=1; key\_UP\_flag=1;} break;

        case 1:

            if (key\_code!=4)

            {key\_UP\_state=0;} break;

        default: {key\_UP\_state=0;} break;

    }

}

void INCREASE\_detect(void)

{

    if (key\_code==2) ///////////////////     INCREASE

    {

        key\_INCREASE\_prestate=key\_INCREASE\_state;

        key\_INCREASE\_state=0;

        if (key\_INCREASE\_prestate==1)

        {   key\_INCREASE\_flag=1;    key\_INCREASE\_timer =0;  }

        else if (key\_INCREASE\_prestate==0)

        {

            if  (++key\_INCREASE\_timer>=KEYTMR\_OF)

            { key\_INCREASE\_flag=1; key\_INCREASE\_timer=0;}

        }

    }

    else

    {

        key\_INCREASE\_prestate = key\_INCREASE\_state;

        key\_INCREASE\_state=1;

        key\_INCREASE\_timer=0;

    }

}

void DECREASE\_detect(void)

{

    if (key\_code==8) ///////////////////     DECREASE

    {

        key\_DECREASE\_prestate=key\_DECREASE\_state;

        key\_DECREASE\_state=0;

        if (key\_DECREASE\_prestate==1)

        {   key\_DECREASE\_flag=1;    key\_DECREASE\_timer =0;  }

        else if (key\_DECREASE\_prestate==0)

        {

            if  (++key\_DECREASE\_timer>=KEYTMR\_OF)

            { key\_DECREASE\_flag=1; key\_DECREASE\_timer=0;}

        }

    }

    else

    {

        key\_DECREASE\_prestate = key\_DECREASE\_state;

        key\_DECREASE\_state=1;

        key\_DECREASE\_timer=0;

    }

}

void display\_ui\_act(unsigned int i)

{

    unsigned int j=0;

    clear\_screen();

    for (j=0;j<act[i]->num;j++)

    {

        display\_GB2312\_string(act[i]->x[j],(act[i]->y[j]-1)\*8+1,act[i]->str[j],act[i]->inverse[j]);

    }

}

void initial\_act(void)

{

//State: a0

    a0.num=6;

    a0.str[0]=a0\_s0; a0.x[0]=3;  a0.y[0]=1;  a0.inverse[0]=0;

    itoatempreture(tempreture, &a0\_s1, &a0\_s2, &a0\_s4);

    a0.str[1]=a0\_s1; a0.x[1]=3;  a0.y[1]=11;  a0.inverse[1]=0;

    a0.str[2]=a0\_s2; a0.x[2]=3;  a0.y[2]=12;  a0.inverse[2]=0;

    a0.str[3]=a0\_s3; a0.x[3]=3;  a0.y[3]=13;  a0.inverse[3]=0;

    a0.str[4]=a0\_s4; a0.x[4]=3;  a0.y[4]=14;  a0.inverse[4]=0;

    a0.str[5]=a0\_s5; a0.x[5]=3;  a0.y[5]=15;  a0.inverse[5]=0;

//State: 1

    a1.num=8;

    frequency\_init(work\_freq);

    itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

    a1.str[0]=a1\_s0; a1.x[0]=1;  a1.y[0]=1;  a1.inverse[0]=0;

    a1.str[1]=a1\_s1; a1.x[1]=1;  a1.y[1]=6;  a1.inverse[1]=1;

    a1.str[2]=a1\_s2; a1.x[2]=1;  a1.y[2]=7;  a1.inverse[2]=0;

    a1.str[3]=a1\_s3; a1.x[3]=1;  a1.y[3]=8;  a1.inverse[3]=0;

    a1.str[4]=a1\_s4; a1.x[4]=1;  a1.y[4]=9;  a1.inverse[4]=0;

    a1.str[5]=a1\_s5; a1.x[5]=1;  a1.y[5]=10;  a1.inverse[5]=0;

    a1.str[6]=a1\_s6; a1.x[6]=1;  a1.y[6]=11;  a1.inverse[6]=0;

    a1.str[7]=a1\_s7; a1.x[7]=7;  a1.y[7]=13;  a1.inverse[7]=0;

//State: 2

    a2.num=1;

    a2.str[0]=a2\_s0; a2.x[0]=1;  a2.y[0]=1;  a2.inverse[0]=0;

//Output with act[0]

    act[0]=&a0;

    display\_ui\_act(0);

}

void ui\_proc0\_0(void)

{

    if(key\_code!=0)

    {

        key\_UP\_flag=0;key\_DOWN\_flag=0;key\_INCREASE\_flag=0;key\_DECREASE\_flag=0;key\_ENTER\_flag=0;

        act[0] = &a1;

        display\_ui\_act(0);

        act\_state = 1;

        ui\_state=001;

    }

    itoatempreture(tempreture, &a0\_s1, &a0\_s2, &a0\_s4);

    display\_ui\_act(0);

}

void ui\_proc1\_001(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[1]=0;

        a1.inverse[7]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 007;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[1]=0;

        a1.inverse[2]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 002;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq1++;

        if (freq1>9) freq1=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[1] = a1\_s1;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq1--;

        if (freq1<0) freq1=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[1] = a1\_s1;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_002(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[2]=0;

        a1.inverse[1]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 001;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[2]=0;

        a1.inverse[3]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 003;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq2++;

        if (freq2>9) freq2=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[2] = a1\_s2;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq2--;

        if (freq2<0) freq2=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[2] = a1\_s2;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_003(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[3]=0;

        a1.inverse[2]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 002;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[3]=0;

        a1.inverse[5]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 005;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq3++;

        if (freq3>9) freq3=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[3] = a1\_s3;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq3--;

        if (freq3<0) freq3=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[3] = a1\_s3;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_005(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[5]=0;

        a1.inverse[3]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 003;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[5]=0;

        a1.inverse[7]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 007;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

        freq4++;

        if (freq4>9) freq4=0;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[5] = a1\_s5;

        display\_ui\_act(0);

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

        freq4--;

        if (freq4<0) freq4=9;

        itoafrequency(freq1, freq2, freq3, freq4, &a1\_s1, &a1\_s2, &a1\_s3, &a1\_s5);

        a1.str[5] = a1\_s5;

        display\_ui\_act(0);

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

    }

}

void ui\_proc1\_007(void)

{

    if(key\_UP\_flag)

    {

        a1.inverse[7]=0;

        a1.inverse[5]=1;

        key\_UP\_flag=0;

        display\_ui\_act(0);

        ui\_state = 005;

    }

    else if (key\_DOWN\_flag)

    {

        a1.inverse[7]=0;

        a1.inverse[1]=1;

        key\_DOWN\_flag=0;

        display\_ui\_act(0);

        ui\_state = 001;

    }

    else if (key\_INCREASE\_flag)

    {

        key\_INCREASE\_flag=0;

    }

        else if (key\_DECREASE\_flag)

    {

        key\_DECREASE\_flag=0;

    }

    else if(key\_ENTER\_flag)

    {

        key\_ENTER\_flag=0;

        a1.inverse[7]=0;

        a1.inverse[1]=1;

        if(checkFreq(freq1, freq2, freq3, freq4)){

            act\_state = 2;

            ui\_state = 0;

            NOKEY\_clock5s\_flag = 0;

            act[0] = &a2;

            display\_ui\_act(0);

        } else {

            frequency\_set(&work\_freq, freq1, freq2, freq3, freq4);

            if(freq1 == 0){

                UARTCharPut(UART6\_BASE, 'A');

                UARTCharPut(UART6\_BASE, 'T');

                UARTCharPut(UART6\_BASE, '+');

                UARTCharPut(UART6\_BASE, 'F');

                UARTCharPut(UART6\_BASE, 'R');

                UARTCharPut(UART6\_BASE, 'E');

                UARTCharPut(UART6\_BASE, 'Q');

                UARTCharPut(UART6\_BASE, '=');

                UARTCharPut(UART6\_BASE, freq\_1[0]);

                UARTCharPut(UART6\_BASE, freq\_2[0]);

                UARTCharPut(UART6\_BASE, freq\_3[0]);

                UARTCharPut(UART6\_BASE, freq\_4[0]);

            } else {

                UARTCharPut(UART6\_BASE, 'A');

                UARTCharPut(UART6\_BASE, 'T');

                UARTCharPut(UART6\_BASE, '+');

                UARTCharPut(UART6\_BASE, 'F');

                UARTCharPut(UART6\_BASE, 'R');

                UARTCharPut(UART6\_BASE, 'E');

                UARTCharPut(UART6\_BASE, 'Q');

                UARTCharPut(UART6\_BASE, '=');

                UARTCharPut(UART6\_BASE, freq\_2[0]);

                UARTCharPut(UART6\_BASE, freq\_3[0]);

                UARTCharPut(UART6\_BASE, freq\_4[0]);

            }

            itoatempreture(tempreture, &a0\_s1, &a0\_s2, &a0\_s4);

            act[0] = &a0;

            display\_ui\_act(0);

            act\_state = 10;

            ui\_state = 0;

        }

    }

}

void ui\_proc2\_0(void)

{

    if(key\_code!=0)

    {

        key\_UP\_flag=0;key\_DOWN\_flag=0;key\_INCREASE\_flag=0;key\_DECREASE\_flag=0;key\_ENTER\_flag=0;

    }

    if(NOKEY\_clock5s\_flag==1)

    {

        act[0] = &a1;

        display\_ui\_act(0);

        act\_state = 1;

        ui\_state = 001;

    }

}

void delay60ms()

{

    if(key\_code!=0)

    {

        key\_UP\_flag=0;key\_DOWN\_flag=0;key\_INCREASE\_flag=0;key\_DECREASE\_flag=0;key\_ENTER\_flag=0;

    }

    if(clock100ms\_flag==1)

    {

        act[0] = &a0;

        display\_ui\_act(0);

        act\_state = 0;

        ui\_state = 0;

    }

}

void ui\_state\_proc(unsigned int ui\_state)

{

    switch (act\_state)

        {

            case 0:

                switch (ui\_state){

                    case 0: ui\_proc0\_0(); break;

                    default: break;

                }

            break;

            case 1: switch (ui\_state){

                    case 001: ui\_proc1\_001(); break;

                    case 002: ui\_proc1\_002(); break;

                    case 003: ui\_proc1\_003(); break;

                    case 005: ui\_proc1\_005(); break;

                    case 007: ui\_proc1\_007(); break;

                    default: break;

                }

            break;

            case 2: switch (ui\_state){

                    case 0: ui\_proc2\_0(); break;

                    default: break;

                }

            break;

            case 10: delay60ms(); break;

            default: break;

        }

}