湖南大學

HUNAN UNIVERSITY

课　程　名　称：电子系统设计与创新基础训练

实验项目名称：游戏操作系统

专　业　班　级：计科 1907

姓　　名：杨杰

学　　号：201908010705

指 导 教　师：况玲

完 成 时　间：2021 年 9 月 16 日

信息科学与工程学院

摘要

近年来随着计算机在社会领域的渗透和大规模集成电路的发展，单片机的应用正在不断地走向深入，由于它具有功能强，体积小，功耗低，价格便宜,工作可靠，使用方便等特点，因此特别适合于与控制有关的系统，越来越广泛地应用于自动控制，智能化仪器，仪表，数据采集，军工产品以及家用电器等各个领域。单片机往往是作为一个核心部件来使用，在根据具体硬件结构，以及针对具体应用对象特点的软件结合，以作完善。单片机应用的重要意义在于，它从根本上改变了传统的控制系统设计思想和设计方法。从前必须由模拟电路或数字电路实现的大部分功能，现在已能用单片机通过软件方法来实现了。这种软件代替硬件的控制技术也称为微控制技术，是传统控制技术的一次革命。

游戏手柄发展至今，拥有广大使用者，配备了更多现代化、增强游戏体验的功能，如无线遥控、体感控制、一键分享等。本系统采用STC15f2k60s2 单片机作为控制平台，结合BSP库，实现游戏手柄的功能，同时附带防沉迷系统，呵护未成年人健康成长。

关键词：游戏手柄；单片机；防沉迷

目录

1. 绪论

1.1 本设计的背景及意义

游戏手柄是一种常见的电子游戏机的部件，通过操纵其按钮等，实现对游戏虚拟角色的控制。游戏手柄发展至今，已经收获了广大玩家的热爱。

网络游戏防沉迷系统（简称：防沉迷系统），是中国有关部门于2005年提出的一种技术手段，旨在解决未成年人沉迷网络游戏的现状。过度沉迷于网络游戏，对未成年人的身体和心理都有较大伤害，各游戏公司都采取了一系列行动共建防沉迷系统，本设计内置防沉迷系统也是为了未成年玩家的身心健康。

1.2 发展现状

游戏手柄经过数十年的发展，拥有了更多现代化、增强体验的功能，如无线遥控、体感控制、一键分享等，某种程度上讲，游戏手柄的发展并不是一条纯粹指向“进步”的单行线，而是不同理念、需求、应用场景、工业设计甚至阴错阳差共同作用下的产物。虽然其变革不以“狂飙突进”的形式呈现，但绝对不乏“本质上”的重要变化。

2. 设计概述

2.1 设计目的

本游戏操作系统在实现基础的游戏控制功能外，还整合了游戏成绩记录、游戏时间限制等实用功能，并提供多种接口扩展，旨在为用户提供更科学、健康、友好的游戏体验。

2.2 设计任务

9月6日-10日：阅读并学习“STC_B学习板”软件支持包使用说明，理解、掌握各模块头文件的使用方法。

9月11日-14日：购买硬件材料（蓝牙模块、惯性运动传感器、OLED显示屏），完成部分模块适配。

9月15日-17日：将已经写好的各模块合并，进行最终调试，修正Bug。

2.3 设计要求

游戏手柄的基础功能：通过导航按键控制方向，通过数字按键实现特定功能，如开始、暂停、重新开始等。

通过USB串口连接上位机作为有线手柄，通过蓝牙模块与上位机通信作为无线手柄。

通过非易失性存储器模块记录游戏成绩，搭载防沉迷系统，细心呵护玩家的身体健康。

通过音乐模块播放游戏音效。

3. 总体方案设计

通过uart1.h模块实现上位机与单片机的通信，借由单片机的导航按键和数字按键控制上位机程序，同时上位机将信息反馈给单片机并显示在单片机的数码管上。通过music.h模块实现音乐播放。通过导航按键和数字按键模块实现按键操作。

4. 硬件原理

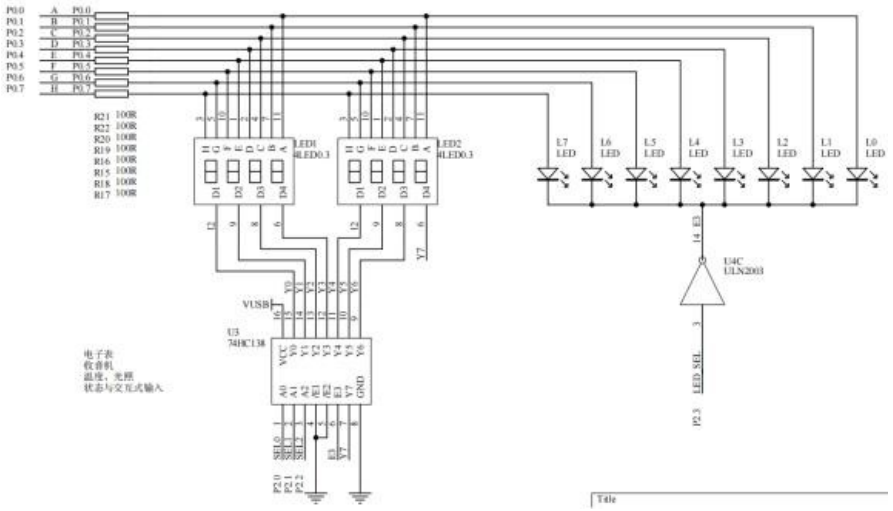
4.1 硬件组成

本设计要用到用于显示的数码管、LED 灯、数字按键、导航按键、振动传感器、无源蜂鸣器。

4.2 显示电路

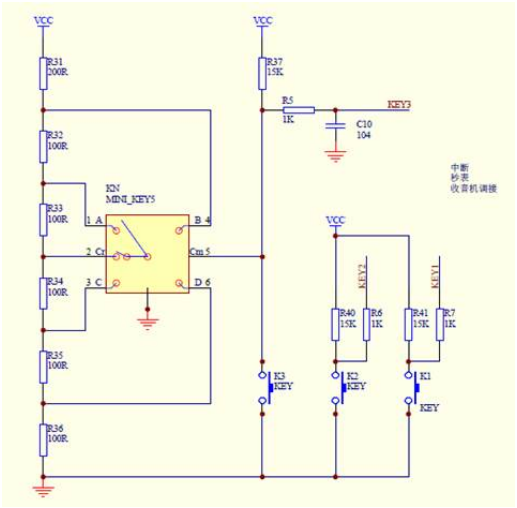
LED 数码管可以分为共阴极和共阳极两种结构:共阴极结构把 8 个发光二极管阴极连接在一起，共阳极结构是把 8 个发光二极管阳极连接在一起。本次实验采用的是共阴极的数码显示管。通过单片机引脚输出高电平，可使数码管显示相应的数字或字母，这种使数码管显示字形的数据称字形码，又称段选码。P0口的8位输出分别控制1个LED数码管的7段和一个小数点；而P2.3经反相器U4C控制74HC138的使能信号E3，结合P2.0、P2.1、P2.2这3个位选控制信号确定8个LED数码管中的哪个被点亮；电阻R15～R22为限流电阻。当段选为高、使能信号有效时，对应的LED管将会发光。通过以一定频率扫描位选信号，修改段选信号进行数码管点亮一段时间，从而给人视觉上几个数码管几乎同时显示的效果。

数码管电路：



P0口的8位输出分别控制8个发光二极管L0~L7的阳极；而P2.3经反相器U4C控制8个发光管阴极E3；当阳极为高（对应P0口位为1）、阴极为低时，对应的二极管将会发光。

4.3 数字按键电路

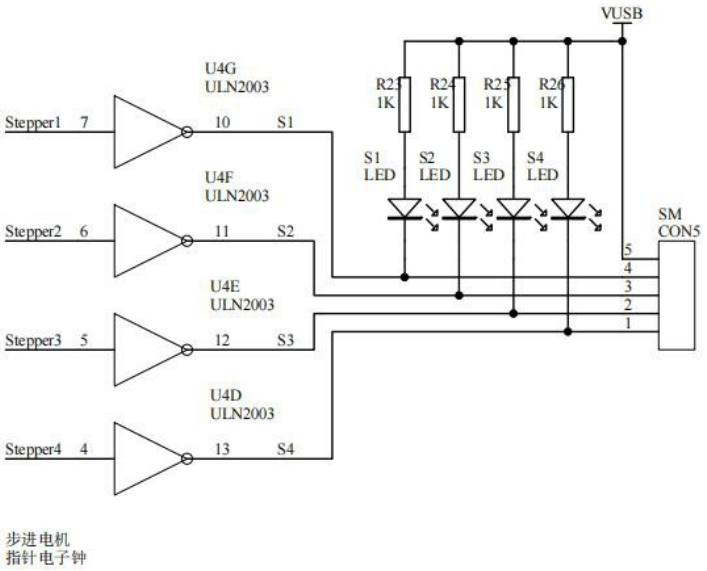


按键电路示意图（三个按键分别是K1、K2、K3）

当按键被按下的时候，电路导通接地，I/O口为低电平；当按键未被按下时，电路断开，I/O口保持高电平。

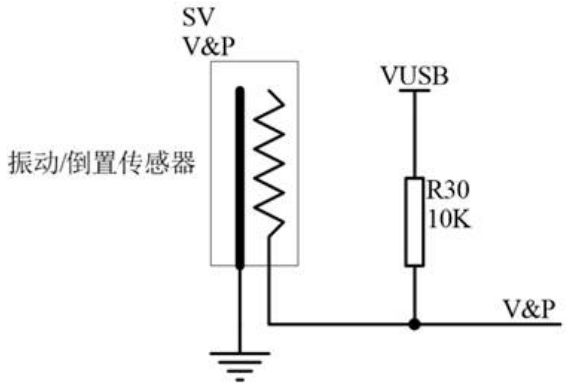
4.4 导航按键电路

导航按键的每一个方向被按下，都会引起实际电压的改变，从而可以根据这个原理，与 A/D 转换器配合，可以判断哪个方位被按下，获取按下后 A/D 转换的结果。



4.5 振动传感器

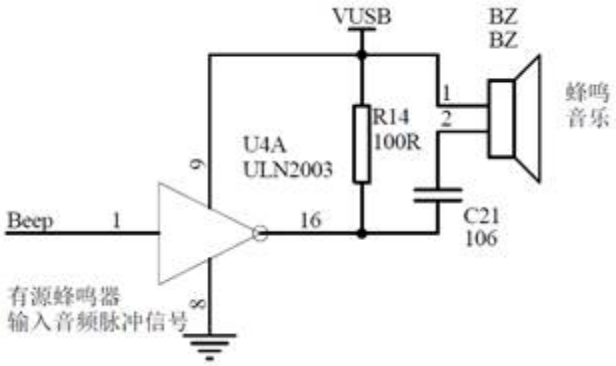
振动传感器电路及原理



振动传感器电路原理图

本实验板中使用的振动传感器是一种简单的器件，管内有一跟固定的导线，在这根导线的周围有另一根较细的导线以螺旋状环绕它。可以想象为一个弹簧旁边有一跟导线。在不振动时，两根导线不会相碰，一旦振动发生，两根导线就会短接。所以我们只需判断导线是否短接了，就可以知道振动是否发生。

4.6 无源蜂鸣器



无源蜂鸣器电路原理图

本实验利用无源蜂鸣器与按键key1两外接模块实现蜂鸣器的发声功能。

蜂鸣器分为有源蜂鸣器和无源蜂鸣器，这里的源特指振荡源；有源蜂鸣器直接加电就可以响起，无源蜂鸣器需要我们给提供振荡源。理想的振荡源为一定频率的方波。

相比与有源蜂鸣器，无源蜂鸣器的优点在于价格便宜，可以通过控制其振动频率来改

变发出的声音，因此，无源蜂鸣器可以用于音乐的播放。而有源蜂鸣器的优点在于使用简单，不需要编写“乐谱”。本实验板使用的无源蜂鸣器是电磁式蜂鸣器，电磁式蜂鸣器由振荡器、电磁线圈、磁铁、振动膜片及外壳等组成。接通电源后，接收到的音频信号电流通过电磁线圈，使电磁线圈产生磁场。振动膜片在电磁线圈和磁铁的相互作用下，周期性地振动发声。

无源蜂鸣器只需改变Beep端口的电平，产生一个周期性的方波即可使蜂鸣器发生声音，不同的频率发出的声音不同。其中，ULN2003是一个功放，用于放大电流。电阻R14和电容C21是用来保护电路的。若人为将Beep端口的电平一直置为高电平，在没有保护电路的情况下，容易烧毁电路，但即使有保护电路也应该注意不要将Beep端口长时间置于高电平，这对器件也是有一定损害的。

程序烧入单片机后，需要按下按键key1才会进行演奏。

5. 软件设计与实现

使用老师所提供的 BSP，在 main.h 中加载所需要的头文件。本次游戏操作系统设计所需的头文件有：系统模块：sys.h；芯片：STC15F2K60S2；显示模块：displayer.h；按键模块：Key.h；ADC 模块：adc.h;音乐模块：music.h 模块；振动传感器模块：Vib.h； Uart1串行通信模块：uart1.h；无源蜂鸣器模块：beep.h。加载完成之后在 main.c 中调用这些头文件中所包含的函数，来实现对用的功能。具体的实现方式如下：

5.1 显示模块：

```
void dealwithDisp()
{
    unsigned char d0, d1, d2, d3, d4, d5, d6, d7;
    //Seg7Print(d0, d1, d2, d3, d4, d5, d6, d7);
    d6=*(rtx + 1) >> 4;
    d7=*(rtx + 1) & 0x0f;
    Seg7Print(20, 27, 32, 20, 31, 12, d6, d7);
}
```

5.2 数字按键模块：

```
void dealwithmykey()
{
    if (GetKeyAct(enumKey1) == enumKeyPress) //示例。按键1:（按下时）当前关卡重新开始
    {
```

```
        *(rxd + 0) = 6;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
        SetBeep(1000, 20);
    }
    if (GetKeyAct(enumKey2) == enumKeyPress) //示例。按键2:（按下时）开始/暂停游戏
    {
        *(rxd + 0) = 7;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
        SetBeep(1000, 20);
    }
}
```

5.3 导航按键模块：

```
void dealwithmyKN()
{
    if (GetAdcNavAct(enumAdcNavKeyUp) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 1;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyDown) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 2;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyLeft) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 3;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyRight) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 4;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyCenter) == enumKeyPress) //回退到上一步
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 5;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKey3) == enumKeyPress) //示例。按键3:(按下时）控制音乐播放"暂停/继续"
    {
```

```
        SetMusic(Music_PM, Music_tone, &song, sizeof(song), enumMscDrvLed); //调整播放节凑和音调
        if (GetPlayerMode() == enumModePlay)
            SetPlayerMode(enumModePause);
        else
            SetPlayerMode(enumModePlay);
    }
}
```

5.4 振动传感器模块：

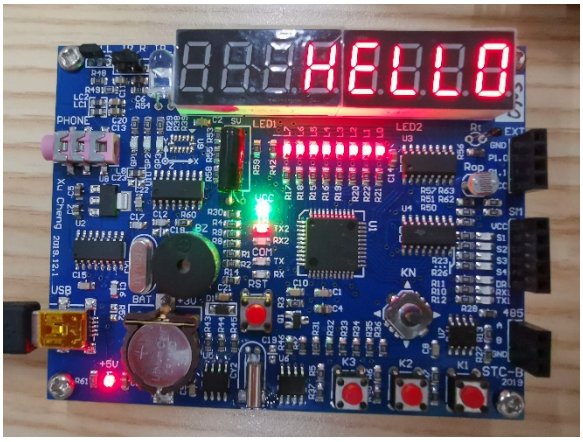
```
void mySV_callback() //示例: 振动事件回调函数：控制音乐播放/暂停
{
    if (GetVibAct())
        if (GetPlayerMode() == enumModePause)
            SetPlayerMode(enumModePlay);
        else
            SetPlayerMode(enumModePause);
}
```

5.5 音乐模块：

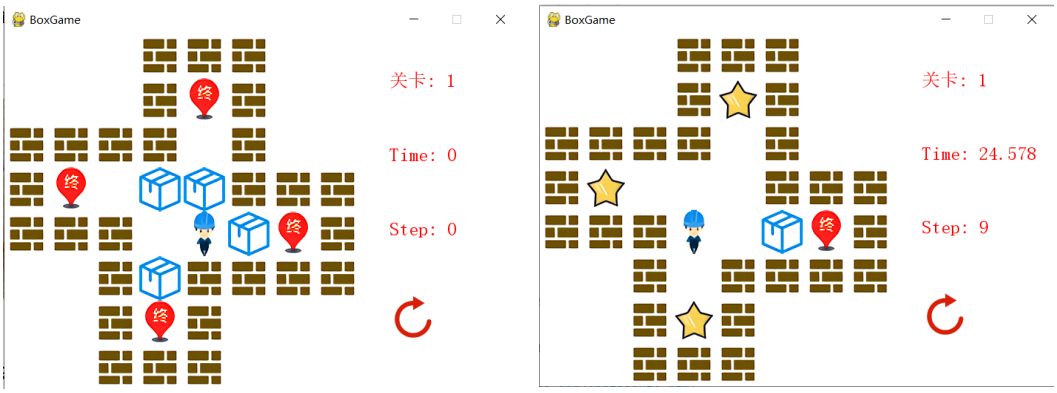
```
if (GetAdcNavAct(enumAdcNavKey3) == enumKeyPress) //示例。按键 3:(按下时）控制音乐播放"暂停/继续"
{
      SetMusic(Music_PM, Music_tone, &song, sizeof(song), enumMscDrvLed); //调整播放节凑和音调
      if (GetPlayerMode() == enumModePlay)
          SetPlayerMode(enumModePause);
      else
          SetPlayerMode(enumModePlay);
}
```

6. 实验过程与测试

使用老师所给出的 BSP 在 keil 中编写程序。用 STC_ISP 打开并下载 hex 文件。下载完成之后数码管最右边五位显示HELLO。



接着上位机运行box_game_main.py程序。游戏起始界面如下：



导航按键上键人物向上移动，导航按键下键人物向下移动，导航按键左键人物向左移动，导航按键右键人物向右移动，导航按键中键人物回退一步。

K2键控制游戏暂停和开始，K1键重新开始当前关卡，K3键控制音乐播放和暂停，同时也可以通过振动传感器控制音乐播放。

7. 设计总结

采用 BSP 在 keil 编程的程序能够编译成功，且 STC_ISP 下载验证能够实现想要实现的功能。本次设计的系统以单片机为控制核心，通过USB实现单片机与上位机的相互通信，并能通过导航按键和数字按键操纵游戏。基于单片机的游戏操作系统设计可推广到各种电子控制系统中。综上所述，该系统的设计和研究在社会生产和生活中具有重要地位。

整个创新设计历时两个星期，直到16号才终于画下句号，在此感慨良多。因为我的设计是自己设计实现的，所以在编程与调试上的时间不短。遇到过很多的Bug也解决了很多的难题，到现在终于可以松一口气说我完成了。

单片机的学习尤其是编程确实非常的难，因为有太多的寄存器需要去学习如何赋值、都代表了什么，有时候看STC的数据手册上的寄存器功能都会感到一头雾水，大段大段的文字、非常多的图片往往让人感到困扰，想要完全理解这太难了。还好有徐老师编写的BSP，大大降低了单片机编程的难度。遇到不懂的地方，在板子上下载看一下，忽略掉那些无关紧要的东西，慢慢就渐入佳境了。

这次小学期的实训我感觉非常有意义，不仅学到了很多的新知识并将之运用到了实践上，而且了解到了这些简单的硬件的工作方式，真是受益匪浅。

附录：

main.c

```
 //******* 用户程序段1：用户程序包含文件 *************//
#include "main.H" //必须。编写应用程序时，仅需改写 main.h 和 main.c文件
#include "song.c" //举例。song.c中编写了音乐（同一首歌）编码
//******* 用户程序段2：用户自定义函数声明 *************//

//******* 用户程序段3：用户程序全局变量定义 *************//
unsigned char Music_tone, Music_PM; //举例。 音乐播放音调、节凑（每分钟节拍数）
unsigned int count;
unsigned char rxd[1] = {0x01};
unsigned char rtx[2];
unsigned char rxdhead[1] = {0xaa};

//******* 用户程序段4：用户自定义函数原型 *************//
#include "function.c"

void my1S_callback() //举例。1S事件回调函数
{
    // count++;
    // if (count == 60)
    // {
    //  count = 0;
    //      SetDisplayerArea(0, 7);
    //  Seg7Print(10, 10, 10, 10, 31, 20, 5, 32);

    // }
}

void myKN_callback() //举例。导航按键事件回调函数
{
    dealwithmyKN();
}

void mykey_callback() // 按键（Key1、Key2）事件回调函数
{
    dealwithmykey();
}

void myADC_callback() //举例。ADC事件回调函数
{
}

void mySV_callback() //示例：振动事件回调函数：控制音乐播放/暂停
{
    if (GetVibAct())
        if (GetPlayerMode() == enumModePause)
            SetPlayerMode(enumModePlay);
        else
            SetPlayerMode(enumModePause);
}

void myUart1_callback()
```

```
{
    dealwithDisp();
}

//******* main()函数 *************//
void main()
{ //主函数 main() 开始          //此行必须！！！

    //******* 用户程序段5：用户main()函数内部局部变量定义 *************//

    //******* 用户程序段6：用户main()函数（初始化类程序） *************//
    //1,加载需要用的模块(由各模块提供加载函数)
    Key_Init();       //举例，需要用到的模块及其函数、方法，必须对其初始化（让其准备、就
绪和响应做后台服务）
    HallInit();       //举例
    VibInit();        //举例
    DisplayerInit(); //举例
    // BeepInit();           //举例
    // MusicPlayerInit();    //举例
    AdcInit(ADCexpEXT); //举例，ADC模块初始化，有参数，选择扩展接口EXT上P1.0、P1.1是
否也做ADC功能
    Uart1Init(1200);     //举例，串口1初始化，有参数，设置Uart1通信波特率

    //2,设置事件回调函数(由sys提供设置函数SetEventCallBack())
    SetEventCallBack(enumEventKey, mykey_callback);  //举例
    SetEventCallBack(enumEventSys1S, my1S_callback); //举例
    SetEventCallBack(enumEventNav, myKN_callback);   //举例，设置导航按键回调函数
    SetEventCallBack(enumEventVib, mySV_callback);
    SetEventCallBack(enumEventXADC, myADC_callback); //扩展接口上新的AD值事件
    SetEventCallBack(enumEventUart1Rxd, myUart1_callback);

    //3,用户程序状态初始化
    SetDisplayerArea(0, 7);
    Seg7Print(10, 10, 10, 23, 20, 26, 26, 0);
    SetUart1Rxd(&rtx, sizeof(rtx), rxdhead, sizeof(rxdhead));
    //设置串口接收方式：数据包条件：接收数据包放置在rxd中，数据包大小rxd大小，数据包头需
要与rxdhead匹配，匹配数量rxdhead大小

    //4,用户程序变量初始化
    Music_PM = 90;
    Music_tone = 0xFC;

    /*****************  MySTC_OS 初始化与加载开
始  ********************************/
    MySTC_Init(); // MySTC_OS 初始化          //此行必须！！！
    while (1)     // 系统主循环               //此行必须！！！
    {
        MySTC_OS(); // MySTC_OS 加载           //此行必须！！！
        /*****************  MySTC_OS 初始化与加载结
束  ********************************/

        //******* 用户程序段7：用户main()函数（主循环程序） *************//

    } //主循环while(1)结束          //此行必须！！！
```

```
} //主函数 main() 结束          //此行必须！！！
```

function.c

```
void dealwithDisp()
{
    unsigned char d0, d1, d2, d3, d4, d5, d6, d7;
    //Seg7Print(d0, d1, d2, d3, d4, d5, d6, d7);
    d6=*(rtx + 1) >> 4;
    d7=*(rtx + 1) & 0x0f;
    Seg7Print(20, 27, 32, 20, 31, 12, d6, d7);
}

void dealwithmyKN()
{
    if (GetAdcNavAct(enumAdcNavKeyUp) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 1;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyDown) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 2;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyLeft) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 3;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyRight) == enumKeyPress)
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 4;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKeyCenter) == enumKeyPress) //回退到上一步
    {
        SetBeep(1000, 20);
        *(rxd + 0) = 5;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
    }
    if (GetAdcNavAct(enumAdcNavKey3) == enumKeyPress) //示例。按键3:(按下时）控制音
乐播放"暂停/继续"
    {
        SetMusic(Music_PM, Music_tone, &song, sizeof(song), enumMscDrvLed); //调整
播放节凑和音调
        if (GetPlayerMode() == enumModePlay)
            SetPlayerMode(enumModePause);
        else
            SetPlayerMode(enumModePlay);
    }
}
```

```
void dealwithmykey()
{
    if (GetKeyAct(enumKey1) == enumKeyPress) //示例。按键1:（按下时）当前关卡重新开始
    {
        *(rxd + 0) = 6;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
        SetBeep(1000, 20);
    }
    if (GetKeyAct(enumKey2) == enumKeyPress) //示例。按键2:（按下时）开始/暂停游戏
    {
        *(rxd + 0) = 7;
        Uart1Print(&rxd, sizeof(rxd)); //从串口1发送出去
        SetBeep(1000, 20);
    }
}
```

main.H

```
/********************************  Ver3.3 说明 ********************************
  (1) 系统工作时钟频率可以在main.c中修改 SysClock赋值（单位Hz）。
      如：code long SysClock=11059200; 定义系统工作时钟频率为11059200Hz（也即11.0592MHz）
      系统工作频率必须与实际工作频率（下载时选择的）一致，以免与定时相关的所有功能出现误差或错误。
    (2) 使用方法:
            1，在工程中加载main.c文件和STC_BSP.lib库文件
                    2，在main.c中选择包含以下头文件（如果要使用可选模块提供的函数和方法，就必须包含其头文件）    :
                                    #include "STC15F2K60S2.H"        //必须，"STC-B学习板"使用MCU指定的头文件
                                    #include "sys.H"                 //必须，sys（MySTC_OS调度程序）头文件
                                    #include "display.H"             //可选，display（显示模块）头文件。
                                    #include "key.H"                 //可选，key（按键模块）头文件。
                                    #include "hall.H"                //可选，hall（霍尔传感器模块）头文件。
                                    #include "Vib.h"                   //可选，Vib（振动传感器模块）头文件。
                                    #include "beep.H"                //可选，beep（蜂鸣器模块）头文件。
                                    #include "music.h"               //可选，music（音乐播放）头文件。
                                    #include "adc.h"                 //可选，adc（热敏、光敏、导航按键、扩展接口ADC功能）头文件。
                                    #include "uart1.h"               //可选，uart1（串口1通信）头文件。
                                    #include "uart2.h"               //可选，uart2（串口2通信）头文件。
                                  #include "stepmotor.h"           //可选，步进电
```

机

```
#include "DS1302.h"                //可选，DS1302实时时钟
#include "M24C02.h"                //可选，24C02非易失性存储器
#include "FM_Radio.h"                //可选，FM收音机
#include "EXT.h"                   //可选，EXT扩展接口（电子秤、超声波测距、旋转编码器、PWM输出控制电机快慢和正反转）
#include "IR.h"                    //可选，38KHz红外通信
```

3，MySTC_Init()是sys初始化函数，必须执行一次；

MySTC_OS()是sys调度函数，应置于while（1）循环中；

4，各可选模块如果选用，必须在使用模块其它函数和方法前执行一次模块所提供的驱动函数（设置相关硬件、并在sys中加载其功能调度）：

```
DisplayerInit();        //显示模块驱动
    Key_Init();             //按键模块驱动
    BeepInit();             //蜂鸣器模块驱动
    MusicPlayerInit();      //蜂鸣器播放音乐驱动
HallInit();             //霍尔传感器模块驱动
VibInit();              //振动传感器模块驱动
AdcInit();              //模数转换ADC模块驱动（含温度、光照、导航按键与按键Key3、EXT扩展接口上的ADC）
StepMotorInit();        //步进电机模块驱动
DS1302Init();           //DS1302实时时钟驱动
FMRadioInit();            //FM收音机驱动
EXTInit();              //扩展接口驱动（含电子秤、超声波测距、旋转编码器、PWM输出，但不含EXT上Uart2和与之相关应用）
Uart1Init();            //Uart1（串口1）驱动：USB上（与计算机通信）
Uart2Init();            //Uart2（串口2）驱动：485接口、或EXT扩展接口（多机通信、Uart方式模块如蓝牙模块）
IrInit();               //38KHz红外通信模块驱动
```

说明：有部分模块不需要驱动；驱动函数有些有参数。（具体见各模块头文件说明）

5，sys和各模块共提供以下事件：

```
numEventSys1mS:             1mS事件（"1毫秒时间间隔到"事件）
enumEventSys10mS:           10mS事件（"10毫秒时间间隔到"事件）
enumEventSys100mS:          100mS事件（"100毫秒时间间隔到"事件）
    enumEventSys1S:             1S事件（"1秒时间间隔到"事件）
    enumEventKey:               按键事件（K1、K2、K3 三个按键有"按下"或"抬起"操作）
    enumEventHall:              霍尔传感器事件（霍尔传感器有"磁场接近"或"磁场离开"事件）
    enumEventVib:               振动传感器事件（振动感器检测到"振动"事件）
    enumEventNav:               导航按键事件（导航按键5个方向、或按键K3 有"按下"或"抬起"操作）
    enumEventXADC:              扩展接口上完成一次ADC
```

```
转换事件   （P1.0、P1.1采取到一组新数据）
                                    enumEventUart1Rxd:          Uart1收到了一个符合指
定要求（数据包头匹配、数据包大小一致）的数据包
                    enumEventUart2Rxd:          Uart2收到了一个符合指定要求（数据包头
匹配、数据包大小一致）的数据包
                    enumEventIrRxd:             红外接收器Ir上收到一个数据
包

                                    对这些事件，应采用"回调函数"方法响应（即用sys提供的
SetEventCallBack()设置用户回调函数），以提高系统性能。

                        6，各可选模块提供的其它函数和具体使用方法请参见：
                            各模块头文件中的说明；
                                main.c提供的推荐程序框架和部分示例；
                                其它可能技术文档或应用示例

编写：徐成（电话18008400450）    2021年2月26日设计，2021年9月1日更新
*****************************************************************************/

#include "STC15F2K60S2.H" //必须。
#include "sys.H"          //必须。
#include "displayer.H"
#include "key.h"
#include "hall.h"
#include "Vib.h"
#include "beep.h"
#include "music.h"
#include "adc.h"
#include "uart1.h"
#include "uart2.h"
#include "stepmotor.h"
#include "DS1302.h"
#include "M24C02.h"
#include "FM_Radio.h"
#include "EXT.h"
#include "IR.h"

code unsigned long SysClock = 11059200; //必须。定义系统工作时钟频率(Hz)，用户必须修
改成与实际工作频率（下载时选择的）一致
#ifdef _displayer_H_                    //显示模块选用时必须。（数码管显示译码表，用
户可修改、增加等）
code char decode_table[] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f,
 0x6f, 0x00, 0x08, 0x40, 0x01, 0x41, 0x48,
/* 序
号:   0   1    2  3    4     5    6    7   8    9   10    11   12   13    14
15     */
/* 显示:   0   1    2    3    4     5    6    7   8    9  (无)   下-  中-  上-  上
中-  中下-   */
                            0x77, 0x7c, 0x39, 0x5e, 0x79, 0x71, 0x3d, 0x76, 0x30,
 0x0e, 0x38, 0x54, 0x5c, 0x73, 0x67, 0x50, 0x78, 0x3e, 0x1c, 0x6e,
                            /* 序
号:   16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
 35  */
                            /* 显
```

```
示:   A   b   C   d   E   F   G   H   I   J   L   n   o   P   q   r   t   U   v
 y   */
                                0x3f | 0x80, 0x06 | 0x80, 0x5b | 0x80, 0x4f | 0x80, 0
x66 | 0x80, 0x6d | 0x80, 0x7d | 0x80, 0x07 | 0x80, 0x7f | 0x80, 0x6f | 0x80};
                              /* 序号:   36  37  38  39  40  41  42  43  44  45*/
                     /* 带小数点显示:   0   1   2   3   4   5   6   7   8   9 */
#endif
```