一、整体框架

1.头文件声明区域：所有底层驱动声明的一个地方

```c
/* 头文件声明区 */
#include <STC15F2K60S2.H>//单片机寄存器专用头文件
#include <Init.h>//初始化底层驱动专用头文件
#include <Led.h>//Led底层驱动专用头文件
#include <Key.h>//按键底层驱动专用头文件
#include <Seg.h>//数码管底层驱动专用头文件
#include "onewire.h"//温度底层驱动专用头文件
#include "iic.h"//DAC底层驱动专用头文件

```

2.变量声明区域：所有用户变量声明的一个地方

```c
/* 变量声明区 */
unsigned char Key_Val,Key_Down,Key_Old,Key_Up;//按键专用变量
unsigned char Key_Slow_Down;//按键减速专用变量
unsigned char Seg_Buf[8] = {10,10,10,10,10,10,10,10};//数码管显示数据存放数组
unsigned char Seg_Point[8] = {0,0,0,0,0,0,0,0};//数码管小数点数据存放数组
unsigned char Seg_Pos;//数码管扫描专用变量
unsigned int Seg_Slow_Down;//数码管减速专用变量
unsigned char ucLed[8] = {0,0,0,0,0,0,0,0};//Led显示数据存放数组
unsigned char Seg_Disp_Mode;//数码管显示模式标志位 0-温度显示界面 1-参数设置界面 2-DAC输出界面
unsigned char Temperature_Params;//温度参数变量（用于设置显示）
unsigned char Temperature_Params_Ctrol = 25;//温度参数控制变量（用于实际控制） 初始值：25
float Voltage_Output;//实时输出电压
float Temperature;//实时温度变量
bit Ouput_Mode;//DAC输出模式 0-DAC输出电压与温度相关 1-DAC给出的关系输出电压
```

3.按键处理区域：所有按键相关的操作都在这个函数内编写

```c

/* 键盘处理函数 */
void Key_Proc()
{
  if(Key_Slow_Down) return;
  Key_Slow_Down = 1;//键盘减速程序

  Key_Val = Key_Read();//实时读取键码值
  Key_Down = Key_Val & (Key_Old ^ Key_Val);//捕捉按键下降沿
  Key_Up = ~Key_Val & (Key_Old ^ Key_Val);//捕捉按键上降沿
  Key_Old = Key_Val;//辅助扫描变量

  switch(Key_Down)
  {
    case 4://界面切换按键
      if(++Seg_Disp_Mode == 3)
        Seg_Disp_Mode = 0;//数码管显示模式在0-2之间循环切换
      if(Seg_Disp_Mode == 1) //当前处于温度参数设置界面
```

4.信息处理区域：分为两大部分 信息采集和数据显示

```c
}

/* 信息处理函数 */
void Seg_Proc()
{
  if(Seg_Slow_Down) return;
  Seg_Slow_Down = 1;//数码管减速程序

  /* 信息采集区域 */
  Temperature = rd_temperature();//实时采集温度数据

  /* 数据显示区域 */
  switch(Seg_Disp_Mode)
  {
    case 0://温度显示界面
      Seg_Buf[0] = 11;//标识符C
      Seg_Buf[4] = (unsigned char)Temperature / 10 % 10;
      Seg_Buf[5] = (unsigned char)Temperature % 10;
      Seg_Buf[6] = (unsigned int)(Temperature * 100) / 10 % 10;
      Seg_Buf[7] = (unsigned int)(Temperature * 100) % 10;
      Seg_Point[5] = 1;//使能小数点
```

5.其他显示区域：Led、蜂鸣器、继电器等其他外设存放的地方

```c

/* 其他显示函数 */
void Led_Proc()
{
  unsigned char i;//用于For循环
  /* DAC相关 */
  if(Ouput_Mode == 0)//DAC输出电压与温度相关
  {
    if(Temperature > Temperature_Params_Ctrol) //当实时温度大于温度参数时
      Voltage_Output = 5;//DAC输出5V
    else//当实时温度小于参数设置时
      Voltage_Output = 0;//DAC输出0V
    //Voltage_Output = (bit)((unsigned char)Temperature / Temperature_Params_Ctrol) * 5;
  }
  else//DAC给出的关系输出电压
```

6.定时器初始化函数：一般无需做任何更改

```c

/* 定时器0中断初始化函数 */
void Timer0Init(void)   //1毫秒@12.000MHz
{
  AUXR &= 0x7F;   //定时器时钟12T模式
  TMOD &= 0xF0;   //设置定时器模式
  TL0 = 0x18;   //设置定时初始值
  TH0 = 0xFC;   //设置定时初始值
  TF0 = 0;    //清除TF0标志
  TR0 = 1;    //定时器0开始计时
  ET0 = 1;    //定时器中断0打开
  EA = 1;     //总中断打开
}
```

7.中断服务函数：一般是用于计时和计数用 一毫秒执行一次中断服务

```
/* 定时器0中断服务函数 */
void Timer0Server() interrupt 1
{
  if(++Key_Slow_Down == 10) Key_Slow_Down = 0;//键盘减速专用
  if(++Seg_Slow_Down == 500) Seg_Slow_Down = 0;//数码管减速专用
  if(++Seg_Pos == 8) Seg_Pos = 0;//数码管显示专用
  Seg_Disp(Seg_Pos,Seg_Buf[Seg_Pos],Seg_Point[Seg_Pos]);
  Led_Disp(Seg_Pos,ucLed[Seg_Pos]);
}
```

8.主函数：一般固定不变

```
/* Main */
void main()
{
  rd_temperature();//上电读取一次温度并且延时750MS避免数据
  Delay750ms();
  System_Init();
  Timer0Init();
  while (1)
  {
    Key_Proc();
    Seg_Proc();
    Led_Proc();
  }
}
```

二、Led 模块的基本使用

1.点亮一个 Led

ucLed[需要点亮的位（0-7）] = 0（灭）/ 1（亮）

```
/* 其他显示函数 */
void Led_Proc()
{
  ucLed[0] = 1;//点亮第一个Led
}
```

2.闪烁一个 Led

第一步：定义两个变量 一个用于计时 一个充当标志位

```
7
8  /* 变量声明区 */
9  unsigned char Key_Val,Key_Down,Key_Old,Key_Up;//按键专用变量
10 unsigned char Key_Slow_Down;//按键减速专用变量
11 unsigned char Seg_Buf[8] = {10,10,10,10,10,10,10,10};//数码管显示数据存放数
12 unsigned char Seg_Point[8] = {0,0,0,0,0,0,0,0};//数码管小数点数据存放数组
13 unsigned char Seg_Pos;//数码管扫描专用变量
14 unsigned int Seg_Slow_Down;//数码管减速专用变量
15 unsigned char ucLed[8] = {0,0,0,0,0,0,0,0};//Led显示数据存放数组
16 unsigned int Timer_500Ms;//计时专用变量
17 bit Led_Flag;//LED标志位
18
```

常见定义类型:

Unsigned char:0-255

Unsigned int:0-65535

Float:小数

Bit:不是 0 就是 1

定义原则：尽量用占用小的变量类型 不要太浪费资源

定义方式：[变量类型]【空格】[变量名称（自己自定义）] （【=】【初始值】）【;】（如果没有给初值 那么默认为 0）

第二步：在中断服务函数中编写代码

```
59 /* 定时器0中断服务函数 */
60 void Timer0Server() interrupt 1
61 {
62   if(++Key_Slow_Down == 10) Key_Slow_Down = 0;//键盘减速专用
63   if(++Seg_Slow_Down == 500) Seg_Slow_Down = 0;//数码管减速专用
64   if(++Seg_Pos == 8) Seg_Pos = 0;//数码管显示专用
65   Seg_Disp(Seg_Pos,Seg_Buf[Seg_Pos],Seg_Point[Seg_Pos]);
66   Led_Disp(Seg_Pos,ucLed[Seg_Pos]);
67   Timer_500Ms++;//中断服务函数1毫秒执行一次 那么这个变量将会一毫秒加一
68   if(Timer_500Ms == 500)//如果这个变量等于五百 那么说明此时已经过了五百毫秒
69   {
70     Timer_500Ms = 0;//复位计时变量 用于下一次计时
71     Led_Flag ^= 1;//异或1就是取反的意思 从0变1 从1变0
72   }
73 }
```

If 语句:

```
If（判断条件）
{
    //条件成立执行这个括号的内容
}
Else
{
    //条件不成立执行这个括号的内容
}
```

第三步：在 Led_Proc 内编写代码

```
/* 其他显示函数 */
void Led_Proc()
{
  ucLed[0] = 1;//点亮第一个Led
  ucLed[1] = Led_Flag;//闪烁第二个Led
}
```

三、Seg 模块的基本使用

1.显示固定数字

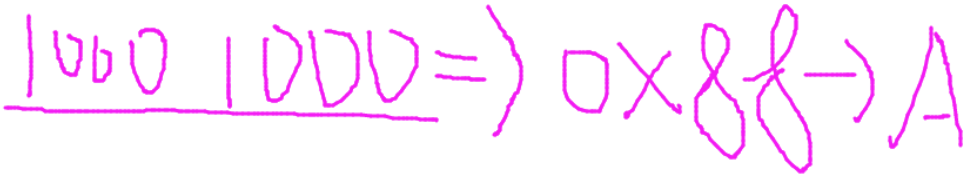
Seg_Buf[需要显示的数码管（0-7）] = 0-9（数字）

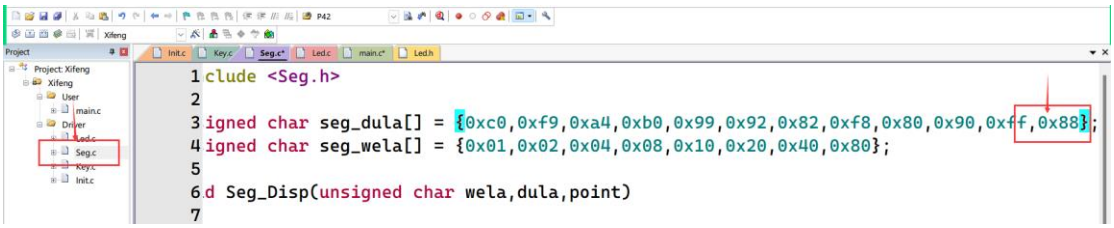
```
/* 信息处理函数 */
void Seg_Proc()
{
  if(Seg_Slow_Down) return;
  Seg_Slow_Down = 1;//数码管减速程序

  Seg_Buf[0] = 5;//第一个数码管显示数字5
  Seg_Buf[1] = 2;//第二个数码管显示数字2
  Seg_Buf[2] = 0;//第三个数码管显示数字0
}
```

2.显示固定字母

第一步：推断字母段码后在底层内添加





第二步：显示字母

```
31
32 /* 信息处理函数 */
33 void Seg_Proc()
34 {
35   if(Seg_Slow_Down) return;
36   Seg_Slow_Down = 1;//数码管减速程序
37
38   Seg_Buf[0] = 5;//第一个数码管显示数字5
39   Seg_Buf[1] = 2;//第二个数码管显示数字2
40   Seg_Buf[2] = 0;//第三个数码管显示数字0
41   Seg_Buf[3] = 11;//第四个数码管显示字母A
42 }
43
44 /* 其他显示函数 */
```

3.显示变量

第一步：定义变量

```
15 unsigned char ucLed[8] = {
16 unsigned int Num = 4321;
```

第二步：显示变量

```
  Seg_Buf[4] = Num / 1000 % 10;//第五个数码管显示Num的千位
  Seg_Buf[5] = Num / 100 % 10;//第六个数码管显示Num的百位
  Seg_Buf[6] = Num / 10 % 10;//第六个数码管显示Num的十位
  Seg_Buf[7] = Num % 10;//第六个数码管显示Num的个位
}
```

Num 的第 X 位 = Num / 1（后面 X 个 0） % 10

个位是 /1 故可以省略不写

四、Key 模块的基本使用

1.按下某个按键执行某个功能

```
31
32   switch(Key_Down)//Key_Down是按键的下降沿 我们对下降沿进行判断
33   {
34     case 19://如果是S19按下
35       Num++;//Num+1
36     break;//固定不变
37   }
```

Switch（Key_Down(下降沿-按下) / Key_Up(上升沿-松手)）

{

Case 某一个按键数字：

//执行语句

Break;

}