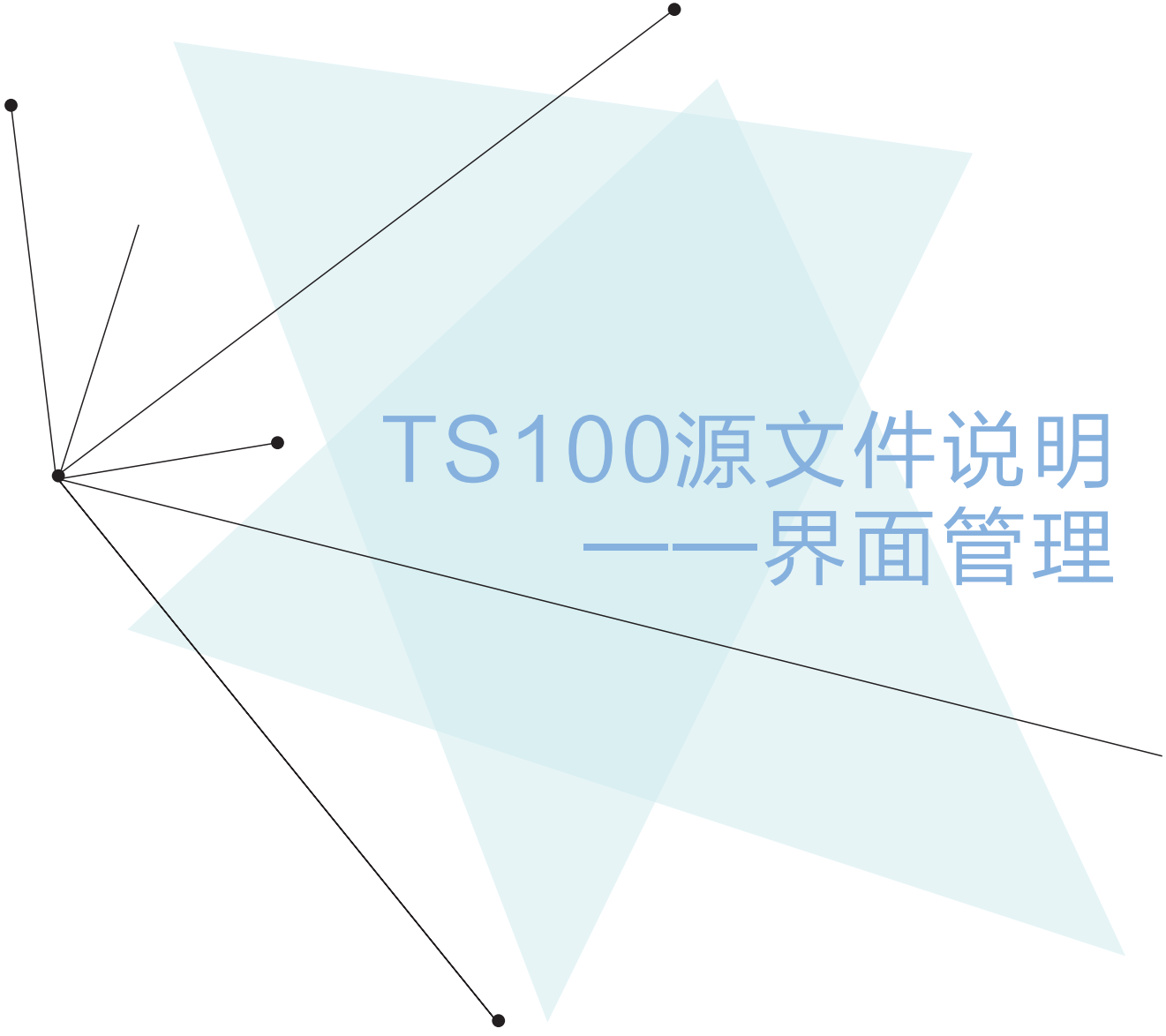


# TS100源文件说明
## ——界面管理

# 目录

# 一、显示函数

TS100采用的屏幕是16*96的分辨率，当需要显示内容当OLED上时，我们只需要按照规则把相应位点亮即可。

```
135 /*******************************************************************************
136  函数名: Oled_DrawArea
137  函数作用:显示一个区域
138  输入参数: x0:起始横坐标
139            y0:起始纵坐标(0,8,16,24)
140            wide:显示内容宽度
141            high:显示内容高度
142            ptr:显示的内容库指针
143  返回参数:下一库指针
144  *******************************************************************************/
145 u8* Oled_DrawArea(u8 x0,u8 y0,u8 wide, u8 high,u8* ptr)
146 {
147     u8 m,n,y;
148
149     n = y0 + high;
150     if(y0 % 8 == 0) m = y0 / 8;
151     else            m = y0 / 8 + 1;
152
153     if(n % 8 == 0)  y = n / 8;
154     else            y = n / 8 + 1;
155
156     for(; m < y; m++) {
157         Set_ShowPos(x0,m);
158         ptr = Data_Command(wide,ptr);
159     }
160     return ptr;
161 }
162
163 /*******************************************************************************
164  函数名: Clean_Char
165  函数作用:清除长度wide位置为k处的屏幕
166  输入参数:    清除位置wide为清除宽度
167  返回参数:NULL
168  *******************************************************************************/
169 void Clean_Char(int k,u8 wide)
170 {
```

Oled.c 文件第145行

输入参数: x0:起始横坐标 (0-96);
y0:起始纵坐标(0,8,16,24);
wide:显示内容宽度;
high:显示内容高度;
ptr:显示的内容库指针(字库);
返回参数:下一个字库指针

# 一、显示函数

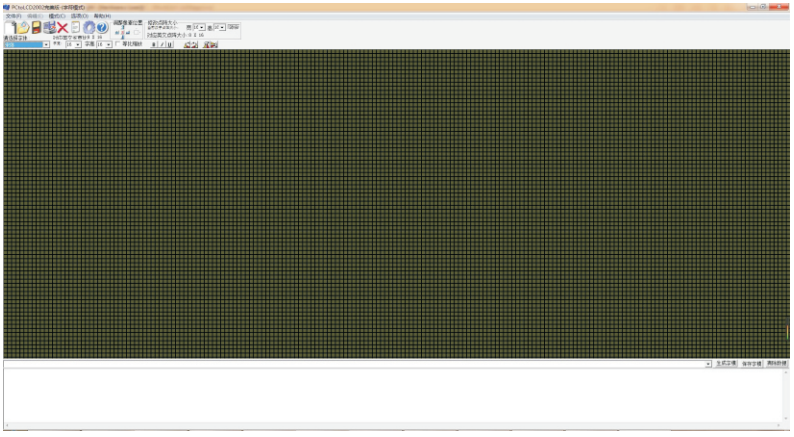
例如要显示CONFIG，找到WordLib.h中的const u8 Config[]数组，
把字库指针config赋值给临时指针ptr，再逐字显示。

```
262 函数作用:显示CONFIG
263 输入参数:NULL
264 返回参数:NULL
265 ********************************************************************************/
266 void Show_Config(void)
267 {
268     u8* ptr;
269     u8 j;
270
271     ptr = (u8*)Config;
272     for(j = 0; j < 6; j++) {
273         ptr = Oled_DrawArea(j*16,0,16,16,ptr);
274     }
275 }
276
277 /********************************************************************************
278 函数名: Show_TempDown
279 函数作用:显示温度下降: 实际温度>>>目标温度
280 输入参数:temp 实际温度,dst_temp 目标温度
281 返回参数:NULL
282 ********************************************************************************/
283 void Show_TempDown(s16 temp,s16 dst_temp)
284 {
285     static u8 guide_ui = 0;
286     char str[8];
287
288     memset(str,0x0,6);
289     sprintf(str,"%d",temp);
290     if(gTemperatureshowflag == 0) {
291         if(temp > 99) str[3] = 'C';
292         else {
293             str[2] = 'C';
294             str[3] = ' ';
295         }
296     } else {
297         if(temp > 99) str[3] = 'E';
298         else {
299             str[2] = 'E';
300             str[3] = ' ';
301         }
302     }
303
304     Display_Str10(1,str);
```

UI.c文件第271-274行
ptr = Oled_DrawArea(j*16,0,16,16,ptr)的意思为：显示当前字库ptr首位开始16*16的字符，显示的位置在(j*16,0)处；从0，16，32，48，64，80四个位置分别显示'C','O','N','F','I','G'，Oled_DrawArea();每次返回下一个显示字符的首字库地址。

# 二、字库制作

字幕是屏幕上显示的文字,它不仅是屏幕画面的重要组成部分,更是必要的补充、装饰等作用。下面详细介绍如何制作字库。



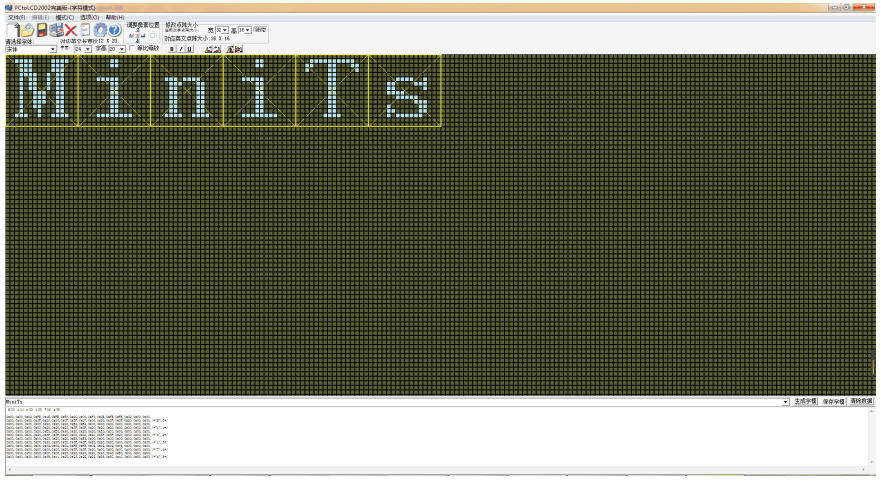
打开Pctolcd2002软件，点开设置，把图片设置成如图所示(阴码，列行式)；


接下来我们就可以开始制作字库。

测试制作的字串为："MiniTs"，总共6位，每位宽度、高度都为16;

在字模选项框内输入："MiniTs"，此时显示字串为 8*16格式，点击设置栏可以修改字体大小位置。

# 二、字库制作




做完上面操作后点击生成字模，得到下列数据；

0x00,0x00,0x02,0xFE,0x1E,0xFE,0xF0,0x00,0x00,0xF0,0x0E,0xFE,0xFE,0x02,0x00,0x00,
0x00,0x00,0x20,0x3F,0x20,0x00,0x0F,0x3F,0x07,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,/*"M",0*/
0x00,0x00,0x00,0x00,0x20,0x20,0x20,0xE6,0xE6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x00,0x00,0x00,0x00,/*"i",1*/
0x00,0x00,0x00,0x20,0xE0,0xE0,0x40,0x20,0x20,0x20,0xE0,0xC0,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,0x00,/*"n",2*/
0x00,0x00,0x00,0x00,0x20,0x20,0x20,0xE6,0xE6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x00,0x00,0x00,0x00,/*"i",3*/
0x00,0x00,0x0C,0x06,0x02,0x02,0x02,0xFE,0xFE,0x02,0x02,0x02,0x06,0x0C,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,0x00,0x00,0x00,0x00,/*"T",4*/
0x00,0x00,0x00,0x00,0xC0,0xC0,0x20,0x20,0x20,0x20,0x40,0x40,0xE0,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x39,0x11,0x23,0x23,0x22,0x22,0x26,0x3C,0x1C,0x00,0x00,0x00,/*"s",5*/

# 二、字库制作

```
const u8 MiniTs[] = { /*16*16*/
  0x00,0x00,0x02,0xFE,0x1E,0xFE,0xF0,0x00,0x00,0xF0,0x0E,0xFE,0xFE,0x02,0x00,0x00,
  0x00,0x00,0x20,0x3F,0x20,0x00,0x0F,0x3F,0x07,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,/*"M",0*/
  0x00,0x00,0x00,0x00,0x20,0x20,0x20,0xE6,0xE6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x00,0x00,0x00,0x00,/*"i",1*/
  0x00,0x00,0x00,0x20,0xE0,0xE0,0x40,0x20,0x20,0x20,0xE0,0xC0,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,0x00,/*"n",2*/
  0x00,0x00,0x00,0x00,0x20,0x20,0x20,0xE6,0xE6,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x00,0x00,0x00,0x00,/*"i",3*/
  0x00,0x00,0x0C,0x06,0x02,0x02,0x02,0xFE,0xFE,0x02,0x02,0x02,0x06,0x0C,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x3F,0x3F,0x20,0x00,0x00,0x00,0x00,0x00,0x00,/*"T",4*/
  0x00,0x00,0x00,0x00,0xC0,0xC0,0x20,0x20,0x20,0x20,0x40,0x40,0xE0,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x39,0x11,0x23,0x23,0x22,0x22,0x26,0x3C,0x1C,0x00,0x00,0x00,/*"s",5*/
};
```

把这些数据放进数组，然后就可以显示了，同理，可以修改字符大小如：12*16，8*16，选择其他大小的字符，而在显示上，把字符大小改成对应的便可以显示；


实物显示图

显示的代码为：

```
ptr = (u8*)MiniTs;
for(j = 0; j < 6; j++) {
    ptr = Oled_DrawArea(j*16,0,16,16,ptr);
}
```

# 三、横向滚动显示

在系统提示按键，设置状态初始等等，都有横向滚动显示的字符。其实就是不断的改变各个字符的所处的位置，便可以实现滚动的显示。下面就简单的分析一下提示按键press的滚动显示代码。

```
202         ptr0 = Oled_DrawArea(0,0,96,16,(u8*)Maplib);
203     } else if(i == 3) {//第4  步
204         for(j = 0 ; j < 6; j++) {
205             k = 84;
206             while(k >= posi) {
207                 ptr0 = (u8*)Maplib1 + j*28;
208                 Clean_Char(k+7,14);
209                 ptr0 = Oled_DrawArea(k,0,14,16,ptr0);
210                 k-=7;
211                 Delay_Ms(10);
212             }
213             posi += 14;
214         }
215         posi = 0;
216     }
217     i++;
218     if(i == 4)  i = 0;
219 }
220 /*******************************************************************************
221 函数名: Show_Ver
222 函数作用:显示版本
223 输入参数:ver 版本号flag (0 :滚动显示 )(1不滚动)
224 返回参数:NULL
225 *******************************************************************************/
226 void Show_Ver(u8 ver[],u8 flag)
227 {
228     u8 *ptr;
229     int k,i;
230     u8 temp0,temp1,temp2;
231
232     if(ver[2] >= 0x30 && ver[2] < 0x3a)   temp1 = ver[2] - 0x30;
233     if(ver[3] >= 0x30 && ver[3] < 0x3a)   temp2 = ver[3] - 0x30;
234     if(ver[0] >= 0x30 && ver[0] < 0x3a)   temp0 = ver[0] - 0x30;
235
```

UI.c 文件第204行：

遍历6个字符，计算每个字符以7个像素点为步进数到达指定位置所需要的显示的位置以及清除上一位置的数；

当遍历与步进协同合作一起显示起来的时候，在人眼看来就是连续的横向滚动显示；

对于其他字符的横向显示，也是同理，只需要把ptr0 = (u8*)Maplib1 + j*28;中的字库Maplib1换成需要显示字库即可。

# 四、竖向滚动显示

TS100中开机竖向出现的版本信号，设置温度时的温度滚动，以及温控状态下的温度指示图标，都有竖向滚动显示；竖向滚动比横向稍微复杂。

UI.c 文件第558行

输入参数: ptr:字节库；
num:个数；
width:宽度；
direction :方向 (0 up, 1 down可以自定义其他，只需改变po_j,po_m的初始化值)。
如：Show_ReverseChar((u8*)Ver_s,8,12,2);
表示向上滚动(direction = 2),
宽度(width = 12)为
12的8个字符(num = 8),
字库为(ptr = Ver_s)Ver_s;

```
550  }
551 /*****************************************************************************
552  函数名: Show_ReverseChar
553  函数作用:竖向动态显示字符
554  输入参数: ptr:字节库    num:个数
555                width:宽度    direction :方向 (0 up, 1 down)
556  返回参数:NULL
557  *****************************************************************************/
558  void Show_ReverseChar(u8* ptr,u8 num,u8 width,u8 direction)
559  {
560      static u32 j = 0,m = 0,po_j[3] = {0,0,0},po_m[3] = {0,0,16};
561      u32 i,k;
562
563      if(direction == 0) { //up
564          if(gUp_flag == 0) { //前一状态不是加热
565              j = 0;
566              m = 0;
567              gUp_flag     = 1;
568              gDown_flag   = 0;
569              gLevel_flag  = 0;
570          } else {
571              j = po_j[0];
```

简单举例说明正常显示在OLED上的字型有着相应的倒序关系，如下列表1表2所示，0表示未点亮，1表示点亮：

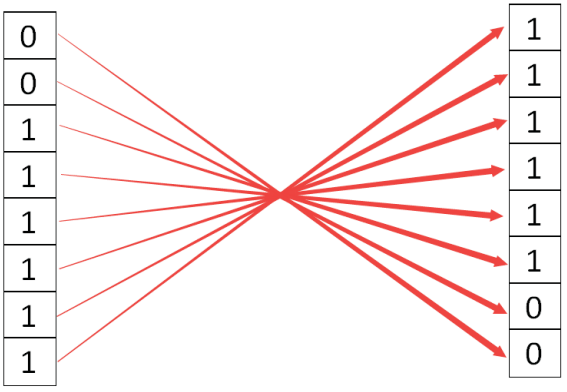



表 1 :在OLED中显示的顺序　　表 2 :在字库保存的顺序

# 四、竖向滚动显示

为了能正常显示，按照以下步骤操作：

UI.c 文件第589行

```
584         }
585     } else {
586         j = po_j[2];
587         m = po_m[2];
588     }
589     for(i = 0; i < width * 2 * num; i++)    gTemp_array[i] = Reverse_Bin8(*(ptr + i)); //逆向8位
590
591     for(k = 0; k < width * 2 * num; k += width * 2)
592         for(i = 0; i < width ; i++) {
593             gTemp_array_u16[i + k] = ((gTemp_array[i + k] & 0x00FF) << 8) | gTemp_array[i + k + width] ;//上半部下半部与成u16 便于移位
594             if(direction == 1) {
595                 if(j == 0)  gTemp_array_u16[i + k] <<= m;//下面空，上面显示
596                 else        gTemp_array_u16[i + k] >>= j;//上面空，下面显示
597             } else { //上
598                 if(m == 0)  gTemp_array_u16[i + k] <<= j;//下面空，上面显示
599                 else        gTemp_array_u16[i + k] >>= m;//上面空，下面显示
600             }
```

第一步：在字库中的表示却是表2所示，所以在Show_ReverseChar()函数中，首先把字库中上8位与下8位都从表2转换为表1中正常显示的格式。

第二步：再把上8位与下8位正常显示的数据，合成16位数据。

UI.c 文件第593行

UI.c 文件第594-603行

第三步：根据滚动方向，把16位数据进行位移。

UI.c 文件第601-602行

第四步：再把位移完成后的16位数据分为两个8位。

```
591     for(k = 0; k < width * 2 * num; k += width * 2)
592         for(i = 0; i < width ; i++) {
593             gTemp_array_u16[i + k] = ((gTemp_array[i + k] & 0x00FF) << 8) | gTemp_array[i + k + width] ;//上半部下半部与成u16 便于移位
594             if(direction == 1) {
595                 if(j == 0)  gTemp_array_u16[i + k] <<= m;//下面空，上面显示
596                 else        gTemp_array_u16[i + k] >>= j;//上面空，下面显示
597             } else { //上
598                 if(m == 0)  gTemp_array_u16[i + k] <<= j;//下面空，上面显示
599                 else        gTemp_array_u16[i + k] >>= m;//上面空，下面显示
600             }
601             gTemp_array[i + k] = (gTemp_array_u16[i + k] & 0xFF00) >> 8;
602             gTemp_array[i + k + width] = gTemp_array_u16[i + k] & 0x00FF;
603         }
604
605     for(i = 0; i < width * 2 * num; i++)    gTemp_array[i] = Reverse_Bin8(gTemp_array[i]); //移位后再逆向
606
607     if(m == 0 && j == 16) { //全显示，换显示'头数'
608         j = 0;
609         m = 16;
610     }
611     if(m == 0)  j++;
612     else        m--;
613
614     if(direction == 0) { //up
615         po_j[0] = j;
616         po_m[0] = m;
617     } else if(direction == 1) {
618         po_j[1] = j;
619         po_m[1] = m;
620     } else {
621         po_j[2] = j;
622         po_m[2] = m;
623     }
624 }
625
626 /*******************************************************************************
627  Show_TempReverse 竖向动态显示温度字符
628  word_num:     个数
629  word_width:   宽度
630  direction :   方向 (0 up, 1 down)
631  *******************************************************************************/
```

第五步：最后再进行一次逆向，把正常显示的表1数据，转换为OLED显示的表2。

UI.c 文件第605行

最后得到移位后的全局变量gTemp_array[],当移位一步步累加起来的时候，显示出gTemp_array[]字库中的内容，也会呈现出竖向滚动的画面。

其他样式的竖向滚动也是同理，不同的是，滚动字符的大小，长度，还有字符初始所处位置也可以不同，滚动特效看起来也就不同了。

# 五、各个界面对应函数

根据上面介绍的字库制作，已经三种不同方式的显示效果，就可以完成TS100界面的改写以及自己独立去制作界面了。下面就相关界面在程序中的调用位置做一个说明，使大家能够快速的定位到需要修改的界面。

```
936     case IDLE:
937         if(gCont == 1) {
938             gCont = 0;
939             Clear_Screen();
940         }
941
942         if(ver_flag == 0) {
943             Display_BG();
944             Show_Ver(device_info.ver,0);
945             ver_flag = 1;
946         } else if(UI_TIMER == 0 && G6_TIMER != 0) {
947             Show_Notice();
948             UI_TIMER = 50;
949         }
950         if(G6_TIMER == 0) { //屏保
951             id_cnt++;
952             if(id_cnt == 50) Sc_Pt(bk--);
953             if(bk == 0) Oled_DisplayOff();
954         }
955         if((Get_MmaShift() == 1) || (Get_gKey() != NO_KEY)) {
956             G6_TIMER = device_info.idle_time;
957             bk = 0x33;
958             Sc_Pt(bk);
959             Oled_DisplayOn();
960         }
961         break;
962     case TEMP_CTR:
963         if(gCont == 0) {
964             gCont = 1;
965             Set_LongKeyFlag(1);
966             Clear_Screen();
967         }
968
969         ht_flag = Get_HtFlag();
970
971         if(ht_flag != 1) {
972             if(td_flag == 1)  Clear_Screen();
973             td_cnt  = 0;
974             td_flag = 0;
975         }
976
977         if(td_cnt == 75 && td_flag == 0) {
978             Clear_Screen();
979             td_flag = 1;
```

UI.c 文件第943行

开机显示的logo
在此函数显示；
logo显示为普通显示，只需要改名字库(ptr = (u8*)Mini)就可以改变需要显示的logo了。

显示固件版本信息
固件版本可以竖向滚动显示，
也可以普通显示，显示此函数，device_info.ver为版本号数组，后一位参数为显示方式选择。

UI.c 文件第944行

UI.c 文件第947行

显示提示按键信息
横向刷过烙铁指示图标以及按键提示press字符，显示此函数，Show_Notice函数中分为4部显示整个的过程。每个过程很清晰，都可以动手改。

# 五、各个界面对应函数

```
994             if((temp_val > device_info.t_work) && (temp_val - device_info.t_work < 18))        te
995             else if((temp_val <= device_info.t_work) && (device_info.t_work - temp_val < 18)) te
996             if(Get_TemperatureShowFlag() == 1) {
997                 temp_val = TemperatureShow_Change(0,temp_val);
998             }
999
1000            Display_Temp(1,temp_val/10);
1001            Show_HeatingIcon(ht_flag,Get_MmaActive());//0升温1降温2恒温
1002            td_cnt++;
1003        }
1004        break;
1005    case TEMP_SET:
1006        Temp_SetProc();
1007        break;
1008    case CONFIG:
1009        if(gCont == 1) {
1010            gCont = 0;
1011            Clear_Screen();
1012        }
1013        switch(Get_gKey()) {
1014        case KEY_CN|KEY_V1:
1015            config_show = 1;
1016            break;
1017        case KEY_CN|KEY_V2:
1018            config_show = 2;
1019            break;
1020        case KEY_CN|KEY_V3:
1021            config_show = 0;
1022            break;
1023        default:
1024            break;
1025        }
1026        if(config_show == 0) {
1027            Show_Config();
1028        } else if(config_show == 1) {
1029            Display_BG();
1030        } else if(config_show == 2) {
1031            Show_Ver(device_info.ver,1);
1032        }
1033        if(config_show != 3) {
1034            Set_gKey(NO_KEY);
1035            config_show = 3;
1036        }
1037        break;
1038    case THERMOMETER:
1039        if(gCont == 0) {
1040            gCont = 1;
1041            Clear_Screen();
1042        }
1043        if(gCalib_flag != 0) {
```

UI.c 文件第1000-1001行

//设置 温度显示、加热指示状态显示
在加热恒温状态下，温度显示，以及
加热指示状态显示，在调用此两个函数temp_val
为放大10倍的温度数，ht_flag为加热指示状态，
Get_MmaActive()是烙铁移动指示。

```
1043        if(gCalib_flag != 0) {
1044            //显示校准完成
1045            Show_Cal(gCalib_flag);
1046            gCalib_flag = 0;
1047            Clear_Screen();
1048        }
1049        if(UI_TIMER == 0) {
1050            temp_val = Get_Temp(0);
1051            if(Get_TemperatureShowFlag() == 1) {
1052                temp_val = TemperatureShow_Change(0,temp_v
1053            }
1054            Display_Temp(1,temp_val/10);
1055            Display_Str(6,"F");
1056            UI_TIMER = 20;
```

如上同理在温度温度计模式、休眠模式下也是
如此显示温度的，温度计模式下的温度显
示在UI.c第1054行；校准结果显示函数在
UI.c第1045行的Show_Cal()，根
据当前显示结果标志，显示校准结果。

显示CONFIG
在配置模式下，有显示CONFIG
(UI.c 第1027行Show_Config())，
有LOGO(UI.c 第1029行Display_BG())，
有版本号(UI.c 第1031行Show_Ver())，
显示方法同上面所说，不同的是调用位置
不同，还是参数不同。

# 五、各个界面对应函数

```
1056            UI_TIMER = 20;
1057        }
1058        break;
1059    case ALARM:
1060        if(gCont == 0) {
1061            gCont = 1;
1062            Clear_Screen();
1063        }
1064        if(gCont == 1 && UI_TIMER == 0) {
1065            Show_Warning();
1066            UI_TIMER = 50;
1067        }
1068        break;
1069    case WAIT:
1070        temp_val = Get_TempVal();
1071        if((temp_val > device_info.t_standby) && (temp_val - device_info.t_standby <= 18))
1072        else if((temp_val <= device_info.t_standby) && (device_info.t_standby - temp_val <
1073
1074        ht_flag = Get_HtFlag();
```

UI.c 文件第1065行

警告界面调用函数
此函数为警告界面调用函数，根据警告类型不同调用不同的字库，显示不同的警告内容。

```
976
977        if(td_cnt == 75 && td_flag == 0) {
978            Clear_Screen();
979            td_flag = 1;
980        }
981
982        if(td_flag && UI_TIMER == 0) {
983            temp_val = Get_TempVal();
984            dst_temp = device_info.t_work;
985            if(Get_TemperatureShowFlag() == 1) {
986                temp_val = TemperatureShow_Change(0,temp_val);
987                dst_temp = TemperatureShow_Change(0,dst_temp);
988            }
989            Show_TempDown(temp_val,dst_temp);
990            UI_TIMER = 50;
991        }
992        if((TEMPSHOW_TIMER == 0) && (!td_flag)) {
993            temp_val = Get_TempVal();
994            if((temp_val > device_info.t_work) && (temp_val - device_info.t_work < 18))
995            else if((temp_val <= device_info.t_work) && (device_info.t_work - temp_val < 1
996            if(Get_TemperatureShowFlag() == 1) {
997                temp_val = TemperatureShow_Change(0,temp_val);
998            }
```

UI.c 文件第989行

目标温度的动态图
函数显示的是当前温度下降到目标温度的动态图；当温度下降比较大，时间比较长时，该函数就会调用。