Dragon Source
龍源電子

OSD 菜单说明文档

--20161104v1.0

一. 主菜单中 ICON、字符串的显示方法

1.显示 ICON 图标

（1）打开程序工程文件，进入 RTD2014OsdDisplay.c 文件，找到 void OsdDispMainMenuIconPage(BYTE ucUpDown, BYTE ucState)函数，如下图所示。

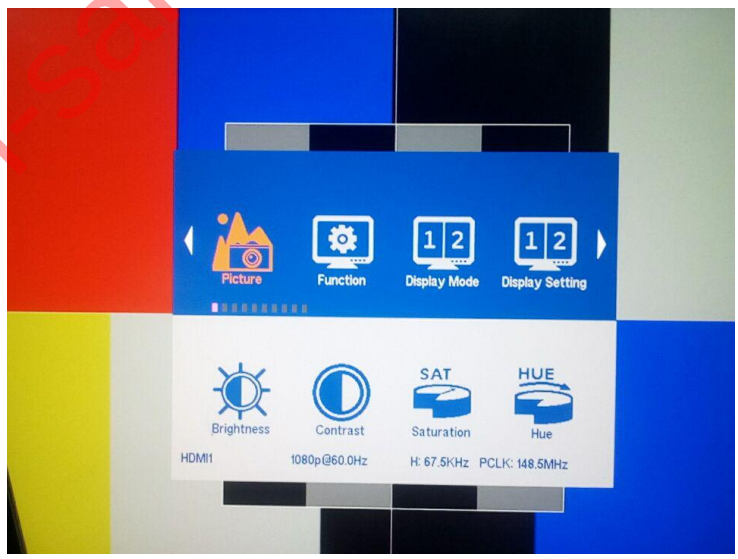
```
00879:
00880: void OsdDispMainMenuIconPage(BYTE ucUpDown, BYTE ucState)//显 示 主 菜 单 图 标 页
00881: {
00882:     BYTE pOsdItemColor[4], i;
00883:
00884:     ScalerTimerWaitForEvent(_EVENT_DEN_STOP);
00885:
00886:     // color selection-选 择 后 的 颜 色
00887:     if(ucUpDown == _UP)
00888:     {
00889:         // up
00890:         for(i=0;i<4;i++)
00891:         {
00892:             pOsdItemColor[i] = _CP_LIGHTBLUE;//浅 蓝 色
00893:         }
00894:     }
00895:     else
00896:     {
00897:         // down
00898:         for(i=0;i<4;i++)
00899:         {
00900:             pOsdItemColor[i] = _CP_BLUE;//蓝 色
00901:         }
00902:     }
00903:
00904:     //Arrow-箭 头
00905:     if(ucUpDown == _UP)
00906:     {
00907:         switch(ucState)
00908:         {
00909:             case _ICON_PAGE_CLEAR://选 择 的 左 右 箭 头
00910:                 OsdDispMainMenuArrow(_UP, _LEFT, _OSD_REJECT);
00911:                 OsdDispMainMenuArrow(_UP, _RIGHT, _OSD_REJECT);
00912:                 break;
00913:
00914:             case _ICON_PAGE_ANALOG_0:
00915:             case _ICON_PAGE_COLOR_0:
00916:             case _ICON_PAGE_ADVANCE_0:
```

（2）关于菜单图标分页排布，如下图所示：

菜单分上下两个部分，上下两部分的图标 void OsdDispMainMenuIconPage(BYTE ucUpDown, BYTE ucState)函数实现。第一个参数为实现菜单上下两部分，第二个参数为菜单功能图标排布分页状态。

1）要显示上部图标时，函数为：OsdDispMainMenuIconPage(_UP, _ICON_PAGE_DP_OPTION);

2）要显示下部图标时，函数为：OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_DP_1_DOT_X);



2.显示图标下面的字符串 string

（1）加载字符串的功能函数为 void OsdFontVLCLoadFont(BYTE ucState)；
ucState 参数表示字符加载的状态。

```
00362:
00363: void OsdFontVLCLoadFont(BYTE ucState)
00364: {
00365:     BYTE ucOsdRotateStatus = _OSD_ROTATE_DEGREE_0;
00366: #if((_OSD_ROTATE_FUNCTION == _OSD_ROTATE_SOFTWARE) || (_OSD_ROTATE_FUNCTION == _OSD_ROTATE_HARDWARE))
00367:     ucOsdRotateStatus = GET_OSD_ROTATE_STATUS();
00368: #endif
00369:
00370:     switch(ucState)
00371:     {
00372:         case _FONT1_GLOBAL:
00373:             ScalerOsdHardwareVLC(tFONT1_GLOBAL, VLC_TABLE_SIZE(tFONT1_GLOBAL), GET_CURRENT_BANK_NUMBER(), _1GLOBAL_START, g_usFontTableStart, u
00374:             break;
00375:
00376:         case _REALTEK_1BIT_LOGO0:
00377:             ScalerOsdHardwareVLC(tICON_REALTEK_1BIT_LOGO0, VLC_TABLE_SIZE(tICON_REALTEK_1BIT_LOGO0), GET_CURRENT_BANK_NUMBER(), _LOGO_START_
00378:             break;
00379:
00380:         case _REALTEK_1BIT_LOGO1:
00381:             ScalerOsdHardwareVLC(tICON_REALTEK_1BIT_LOGO1, VLC_TABLE_SIZE(tICON_REALTEK_1BIT_LOGO1), GET_CURRENT_BANK_NUMBER(), _Logo1_0x00, g
00382:             break;
00383:
00384:         default:
00385:             break;
00386:     }
00387: }
00388:
```

（2）通过调用字符数组显示字符：
void OsdWindowDrawingByFont(BYTE ucWindow, BYTE ucRow, BYTE ucCol, BYTE ucWidth, BYTE ucHeight, BYTE ucColor)；

```
00135: void OsdWindowDrawingByFont(BYTE ucWindow, BYTE ucRow, BYTE ucCol, BYTE ucWidth, BYTE ucHeight, BYTE ucColor)
00136: {
00137:     WORD hstart,hend,vstart=0,vend;
00138:
00139:     hstart = ((WORD)ucCol * 12);
00140:     hend = hstart +((WORD)ucWidth * 12);
00141:
00142:     if(ucRow > 0)
00143:     {
00144:         vstart = (ucRow * 18);
00145:     }
00146:     vend = vstart + (18 * ucHeight);
00147:
00148:     OsdWindowDrawing(ucWindow, hstart, vstart, hend, vend, ucColor);
00149: }
```

3.将图标和字符串一起显示
（1）实现函数： void OsdDispMainMenuIconString(BYTE ucIconPos, WORD usIcon, BYTE ucColor)；
参数第一个表示图标显示位置，第二个表示图标数组，第三个表示颜色。

```
00802:
00803: void OsdDispMainMenuIconString(BYTE ucIconPos, WORD usIcon, BYTE ucColor)
00804: {
00805:     BYTE ucRow = 4, ucCol = 4;
00806:     BYTE ucFontPage = _PFONT_PAGE_0;
00807:     WORD usIconLoad = 0;
00808:
00809:     ucRow = ((ucIconPos / 4) ? ROW(14) : ROW(4));
00810:     ucCol = COL(4) + ((ucIconPos % 4) * 10);
00811:
00812:     if(usIcon == _MENU_NONE)
00813:     {
00814:         OsdFuncClearOsd(ucRow, (ucCol - 2), WIDTH(10), HEIGHT(5));
00815:         return;
00816:     }
00817:
00818:     // icon
00819:     if((usIcon >= _ICON_A0_PORT) && (usIcon <= _ICON_D7_PORT))
00820:     {
00821:         switch(usIcon)
00822:         {
00823:             case _ICON_A0_PORT :
00824:                 usIconLoad = _A0_INPUT_TYPE;
00825:                 break;
00826:             case _ICON_D0_PORT :
00827:                 usIconLoad = _D0_INPUT_TYPE;
00828:                 break;
00829:             case _ICON_D1_PORT :
00830:                 usIconLoad = _D1_INPUT_TYPE;
00831:                 break;
00832:             case _ICON_D2_PORT :
00833:                 usIconLoad = _D2_INPUT_TYPE;
00834:                 break;
00835:             case _ICON_D3_PORT :
00836:                 usIconLoad = _D3_INPUT_TYPE;
00837:                 break;
00838:             case _ICON_D4_PORT :
00839:                 usIconLoad = _D4_INPUT_TYPE;
```

（2）最后在 void OsdDispMainMenuIconPage(BYTE ucUpDown, BYTE ucState)函数中调用此函数，如图所示为

OsdDispMainMenuIconPage(BYTE ucUpDown, BYTE ucState)函数调用函数

OsdDispMainMenuIconString((ucUpDown + 0), _ICON_NONE, pOsdItemColor[0]);

```
01083:
01084:     switch(ucState)//修 改 菜 单 图 标 文 字
01085:     {
01086:         case _ICON_PAGE_CLEAR:
01087:             OsdDispMainMenuIconString((ucUpDown + 0), _ICON_NONE, pOsdItemColor[0]);//显 示 图 标 字 符
01088:             OsdDispMainMenuIconString((ucUpDown + 1), _ICON_NONE, pOsdItemColor[1]);
01089:             OsdDispMainMenuIconString((ucUpDown + 2), _ICON_NONE, pOsdItemColor[2]);
01090:             OsdDispMainMenuIconString((ucUpDown + 3), _ICON_NONE, pOsdItemColor[3]);
01091:             break;
01092:
01093:         case _ICON_PAGE_MAIN_0://菜 单 第 一 页
01094:             OsdDispMainMenuIconString((ucUpDown + 0), _ICON_DISPLAYMODE,    pOsdItemColor[0]);
01095:             if((GET_OSD_DISPLAY_MODE() == _OSD_DM_1P) || (GET_OSD_DISPLAY_MODE() == _OSD_DM_2P_LR) || (GET_OSD_DISPLAY_MODE() == _OSD_DM_2P_TB) ||(G
01096:             {
01097:                 OsdDispMainMenuIconString((ucUpDown + 1), _ICON_DISPLAYFUNCTION,pOsdItemColor[1]);
01098:             }
01099:             else
01100:
01101:                 OsdDispMainMenuIconString((ucUpDown + 1), _ICON_DISPLAYFUNCTION,_CP_GRAY);
01102:             }
01103:
01104:             if((GET_OSD_DISPLAY_MODE() == _OSD_DM_1P) || (GET_OSD_HLWIN_TYPE() != _HL_WIN_OFF))
01105:             {
01106:                 OsdDispMainMenuIconString((ucUpDown + 2), _ICON_SELECTREGION,   _CP_GRAY);
01107:             }
01108:             else
01109:             {
01110:                 OsdDispMainMenuIconString((ucUpDown + 2), _ICON_SELECTREGION,   pOsdItemColor[2]);
01111:             }
01112:             OsdDispMainMenuIconString((ucUpDown + 3), _ICON_PICTURE,        pOsdItemColor[3]);
01113:             break;
01114:
01115:         case _ICON_PAGE_MAIN_1://菜 单 第 二 页
01116:             OsdDispMainMenuIconString((ucUpDown + 0), _ICON_ANALOG,         pOsdItemColor[0]);
01117:             OsdDispMainMenuIconString((ucUpDown + 1), _ICON_COLOR,          pOsdItemColor[1]);
01118:             OsdDispMainMenuIconString((ucUpDown + 2), _ICON_ADVANCE,        pOsdItemColor[2]);
01119:             OsdDispMainMenuIconString((ucUpDown + 3), _ICON_INPUT,          pOsdItemColor[3]);
01120:             break;
```

通过这些函数最后达到实现图标和字符串有序的排布的目的。

二. 菜单动作的实现方法

菜单的实现离不开按键，下面先介绍下按键实现函数，再介绍按键实现菜单的功能。

1.菜单实现函数：

在 RTD2014Osd.c 中找到 void MenuDisplayMode(void)函数，如下图所示：

```
00693: void MenuDisplayMode(void)
00694: {
00695:     switch(GET_KEYMESSAGE())
00696:     {
00697:         case _MENU_KEY_MESSAGE:
00698:             SET_OSD_STATE(GET_OSD_DISPLAY_MODE() + _MENU_DISPLAYMODE_1P);
00699:             OsdDispMainMenuCursor(GET_OSD_STATE(), GET_OSD_STATE_PREVIOUS(), _INSUBSET);
00700:             OsdDispMainMenuOptionSetting(_OPTION_DISPLATMODE_TYPE,_SELECT, GET_OSD_DISPLAY_MODE());
00701:             break;
00702:
00703:         case _RIGHT_KEY_MESSAGE:
00704:             if((GET_OSD_DISPLAY_MODE() == _OSD_DM_1P) || (GET_OSD_DISPLAY_MODE() == _OSD_DM_2P_LR) || (GET_OSD_DISPLAY_MODE() == _OSD_DM_2P_
00705:             {
00706:                 SET_OSD_STATE(_MENU_DISPLAYFUNCTION);
00707:                 OsdDispMainMenuCursor(GET_OSD_STATE(), GET_OSD_STATE_PREVIOUS(),_OUTSUBSET);
00708:                 OsdDispClearSelectColor(_DOWN);
00709:                 OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_DISPLAYFUNCTION);
00710:             }
00711:             else if(GET_OSD_HLWIN_TYPE() == _HL_WIN_OFF)
00712:             {
00713:                 SET_OSD_STATE(_MENU_SELECTREGION);
00714:                 OsdDispMainMenuCursor(GET_OSD_STATE(), GET_OSD_STATE_PREVIOUS(),_OUTSUBSET);
00715:                 OsdDispClearSelectColor(_DOWN);
00716:                 OsdDispMainMenuOptionSetting(_OPTION_SELECTREGION_TYPE,_UNSELECT,GET_OSD_SELECT_REGION());
00717:                 if(GET_OSD_DISPLAY_MODE() == _OSD_DM_2P_TB)
00718:                 {
00719:                     OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_SELECTREGION_2P_TB);
00720:                 }
00721:                 else if(GET_OSD_DISPLAY_MODE() == _OSD_DM_4P)
00722:                 {
00723:                     if((GET_OSD_SELECT_REGION() >= _OSD_SR_4P_LT_INSIDE) && (GET_OSD_SELECT_REGION() <= _OSD_SR_4P_LB_OUTSIDE))
00724:                     {
00725:                         OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_SELECTREGION_4P_0);
00726:                     }
00727:                     else if((GET_OSD_SELECT_REGION() >= _OSD_SR_4P_RT_INSIDE) && (GET_OSD_SELECT_REGION() <= _OSD_SR_4P_RB_OUTSIDE))
00728:                     {
00729:                         OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_SELECTREGION_4P_1);
```

1）通过 GET_KEYMESSAGE()获取按键键值。

2）通过 SET_OSD_STATE(GET_OSD_DISPLAY_MODE() + _MENU_DISPLAYMODE_1P);设置不同的功能函数进而确定不同按键对应的功能。

2.菜单按键如下图所示：

从左往右依次是：菜单 menu，右键（VOL+），左键(VOL-)，退出键(EXIT);

3.菜单动作的实现

1.当按下菜单按键 menu 后，出现菜单首页，如图 1 所示。

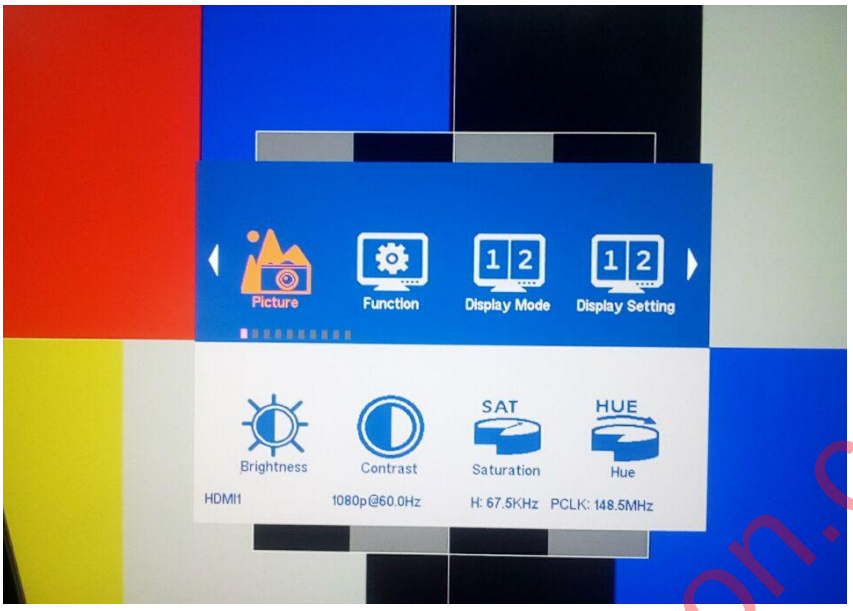


图 1　菜单首页

1）从图一可知，菜单一页共分有四个图标，选中的图标呈橘黄色，未选中的图标呈白色，中间的矩形小白点为索引提示，提示为第几个图标。另外菜单分两部分，上部分为菜单功能，下部分为菜单功能的具体实现；

2）第一页菜单上部分的功能图标分别是：Piture；Fuction；Display Mode；Display Setting。

3）菜单上部分第一个图标为 Picture，菜单下半部分显示的是 Picture 图标的具体功能内容，分别表示的内容是：亮度（brightness），对比度（contrast），饱和度（saturation），色调（hue）。

2.按下右键后第二个图标呈橘黄色，而第一个变成看白色，依次类推，每次按下右键，选中的图标右移一个，到了第一页第四个图标后进入菜单第二页。如图 2 所示。

图 2　菜单第二页

菜单第二页的功能图标分别是：Select Region, Input,Audio,VGA;

值得注意的是当看到图标呈灰色时，表示此图标无功能设置项。

3.第二页翻完之后进入菜单最后一页，索引提示倒数第二个矩形高亮。如图 3 所示。

（1）最后一页的第一个图标是系统设置；从下部分的功能也可以看出系统设置功能包括：复位（Rest），菜单定时（Menu Time），菜单横向扩展和菜单纵向扩展。

（2）最后一页第二个图标为信息提示；信息主要显示视频分辨率大小，视频输入的频率，视频接入的端口号等视频重要参数。

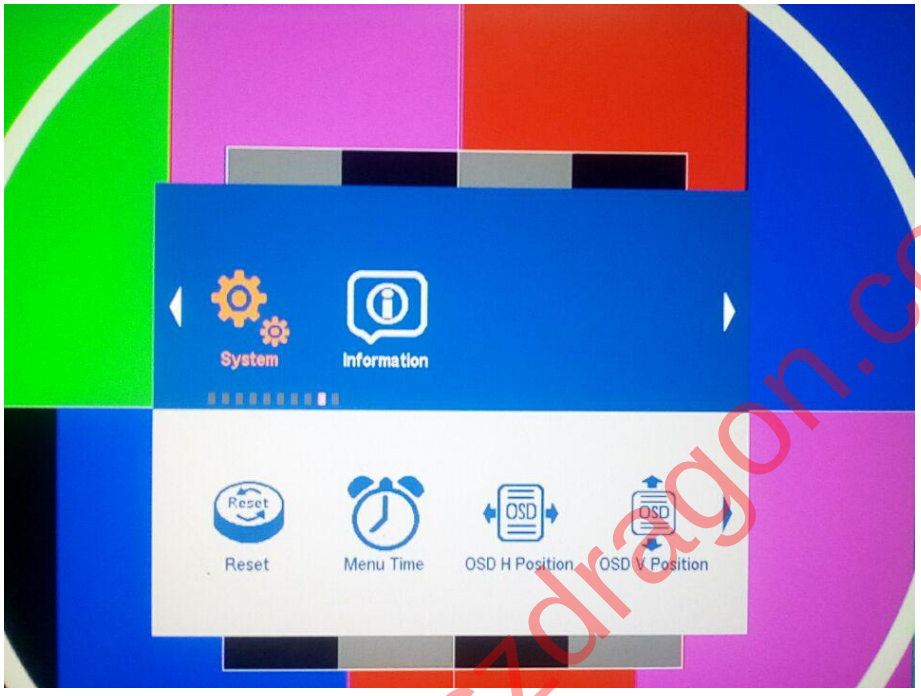
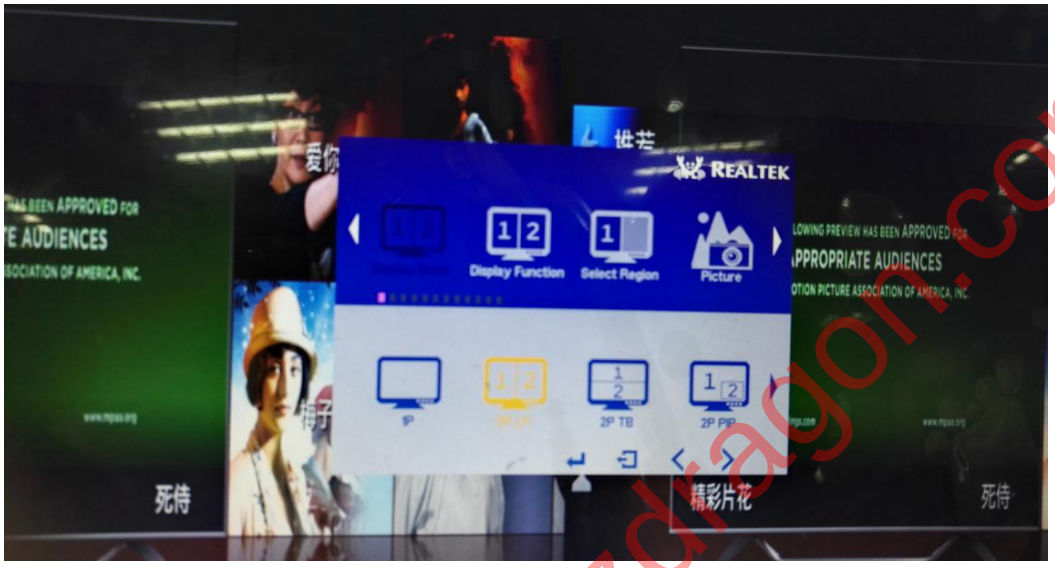


图 2 菜单第三页

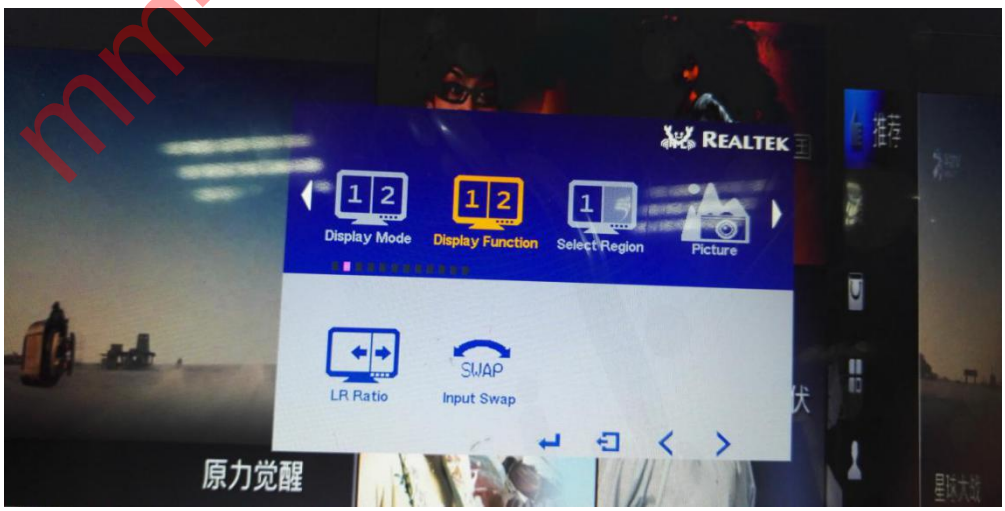
三．主菜单的整体及各个菜单组成部分介绍

主菜单的整体由 12 个菜单组成如下：

1. MenuDisplayMode 的子菜单有 5 个（1P, 2P_LR（左右）, 2P_TB（上下）, 2P_PIP（画中画）, 4P（四画面），）如图 1 中 2P_LR

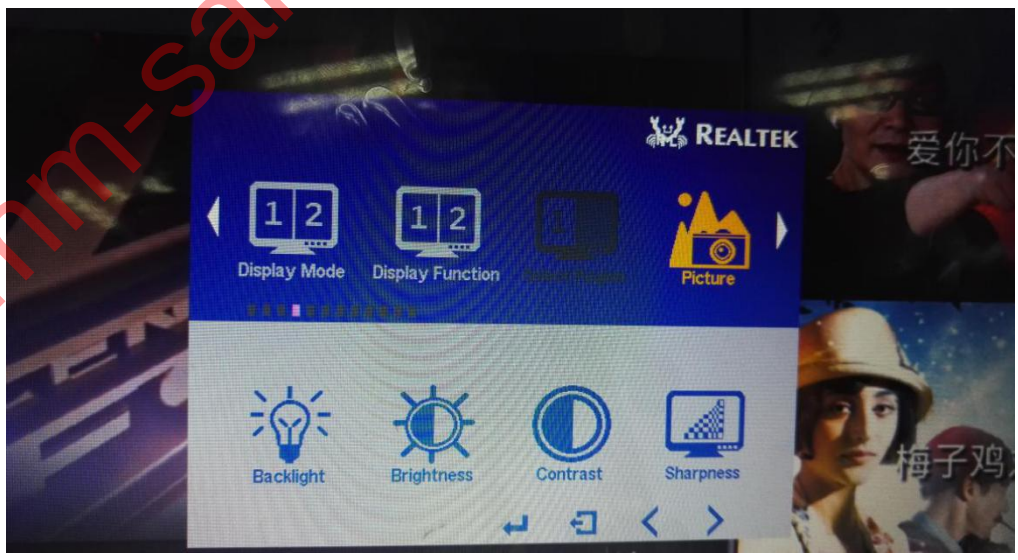


2. MenuDisplayFunction, 子菜单在 1p 模式下主要子菜单是 DispRotate（画面旋转 0.90.180.270）; 2p_LR，2p_TB 模式下主要子菜单是 LRratio（左右调整）,InputSwap（对调），2P_PIP PipPosition(pip 位置),PipTransparency(PIP 透明度),PipSize（pip 的大小）,InputSwap（pip 对调），



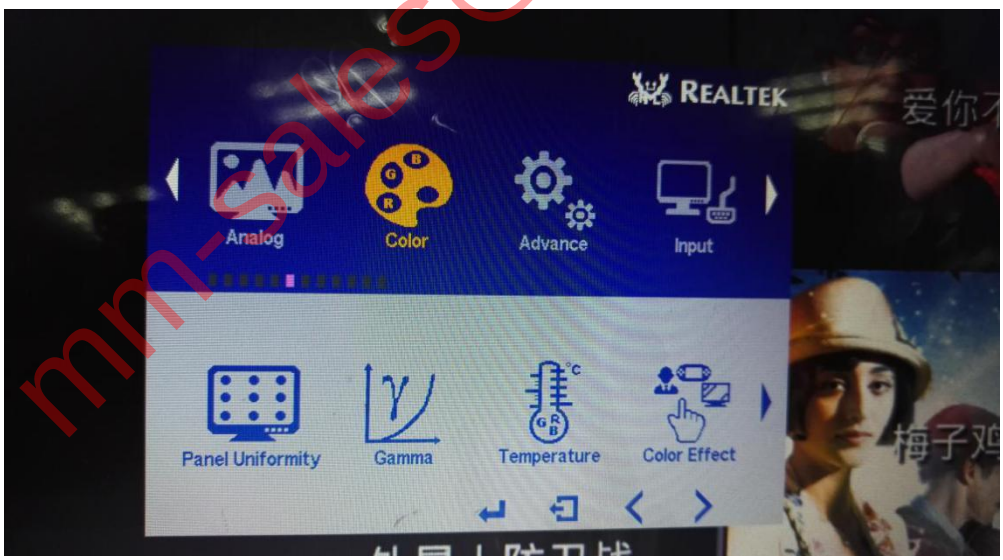
3. MenuSelectRegion 这个功能只有在 2P 及以上才有功能，举 2P_PIP 为例子菜单有 sub, main, full, 这里要说下如下图选的是 main 那么调节亮度. 对比度. 锐利度等时都是调 main 画面，pip 子画面亮度对比度的都不变，这样子. 主画面就有一个对比，有的客户选择这个功能没看到现象就说没功能，并不知道是这样用，特别说下；

4. MenuPicture 的子菜单主要有 4 个 Backlight（背光）, Brightness（亮度）,Contrast（对比度）,Sharpness（锐利度）,如下图：



5. MenuAnalog, 子菜单是在 vga 信号下有功能 Auto（自动调整）,HPos（水平位置）,VPos（垂直位置）,Clock（时钟）,Phase（相位）；如下图（在 hdmi 信号下）

6.MenuColor, 子菜单组成由 PanelUniformity（板均匀性）,Gamma,Temperature（色温）,ColorEffect（彩色效应）,Demo, Hue（色调）,Saturation（饱和度）,如下图

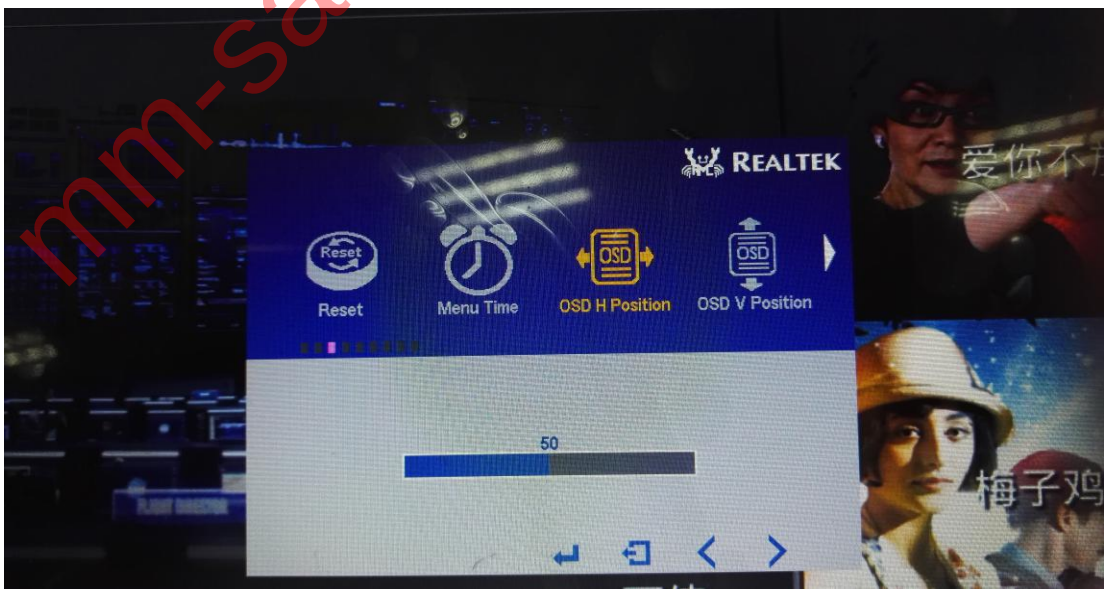


7.MenuAdvance 主要由 Aspect Ratio（屏幕高宽比有 full，16：9，4：3，5：4，1：1）, OverScan（关和开）,DDCCI, UltraVivi（高亮度镜面宽屏）,DCR, DpOption（dp 通道选择）, DpEDID（单 dp edid,双 dp edid）, Clone mode, FreeSync, 如下部分图：

8.MenuInput,如图：

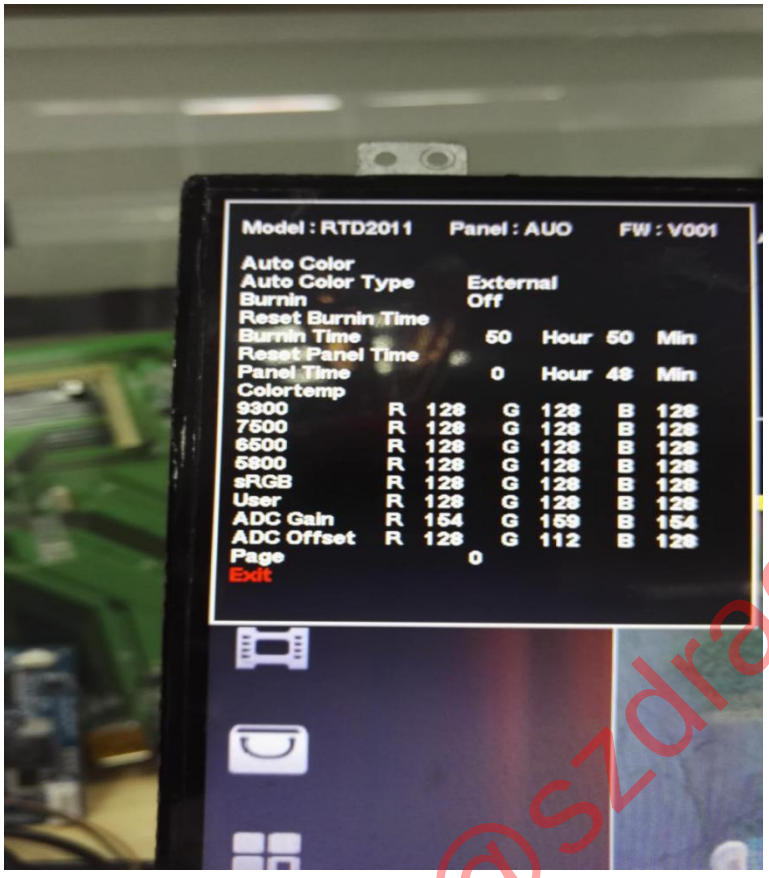
9. MenuAudio, 由 4 个子菜单组成 Volume（音量）,Mute（静音）,StandAlong,AudioSource（analog（模拟）和 digital（数学））,如下图：

10. MenuOther 由 4 个子菜单组成 Reset（复位）,MenuTime（菜单显示时间）,OsdHPos（菜单水平位置）,OsdVPos（菜单垂直位置）,Language（语言 english，Chinese）,Transparency（菜单透明度）,Rotate（菜单翻转 0，90，270）,BorderWidth（菜单边框宽）,BorderColor（菜单边框颜色），如下部分图：



11.MenuInformation 主要显示当前的信息

12. MenuFactory 工厂菜单；



四. 画 2796 demo 菜单的基本步骤

1.先介绍程序中添加菜单的位置及各函数意义；

首先在程序中查找 OperationTable[],包含菜单的所有项及其调节，在 MenuNone 函数中执行按键（一般是五键式 menu,left,right,exit,power）的 mask，在没有任何操作下，menu_key 主要做的动作是画出主菜单的框架，平时改菜单的大小，结构，都在这个函数中更改，left，right，exit，主要是快捷键的添加，例如 information osd 显示，切换 source，vga 自动调整，亮度，对比度，音量等等。

```
void OsdDispMainMenu(void)
{
    BYTE ucTest = 0;

    g_ucFontPointer0 = _OSD_PAGE_0_START;
    g_ucFontPointer1 = _OSD_PAGE_1_START;
    g_ucFontPointer2 = _OSD_PAGE_2_START;

    OsdFuncApplyMap(WIDTH(_OSD_MAIN_MENU_WIDTH), HEIGHT(_OSD_MAIN_MENU_HEIGHT), COLOR(_CP_BLACK, _CP_BG));

//20140304 Abel
#if(_OSD_ROTATE_FUNCTION == _OSD_ROTATE_HARDWARE)
    ScalerOsdMapRotation(GET_OSD_ROTATE_STATUS(), _ENABLE, HEIGHT(_OSD_MAIN_MENU_HEIGHT), _DISABLE, 0, _ENABLE);
#endif

    OsdFuncBlending(_OSD_TRANSPARENCY_ONLY_WINDOW);
    OsdFuncTransparency(GET_OSD_TRANSPARENCY_STATUS());

    // Adjust Color Palette
    OsdPaletteSelectPalette(_PALETTE_MAIN_MENU);

    // Load Font & Icon
    OsdFuncSet2BitIconOffset(_2BIT_ICON_OFFSET);

    OsdFontVLCLoadFont(_FONT1_GLOBAL);
    //OsdFontVLCLoadFont(_FONT2_ICON_MENU);

   // Background window
   //20140210 Abel Modify
#if(_OSD_ROTATE_FUNCTION == _OSD_ROTATE_SOFTWARE)
    if((GET_OSD_ROTATE_STATUS() == _OSD_ROTATE_DEGREE_90)||(GET_OSD_ROTATE_STATUS() == _OSD_ROTATE_DEGREE_270))
    {
```

```
   // Realtek Mark
   OsdFontPut1BitTable( ROW(0), COL(32), tOSD_iREALTEK, COLOR(_CP_WHITE, _CP_BG));

   // Main Menu Icon & String

   if(GET_OSD_STATE() == _MENU_DISPLAYMODE)
   {
       OsdDispMainMenuIconPage(_UP, _ICON_PAGE_MAIN_0);
       OsdDispMainMenuOptionSetting(_OPTION_DISPLAYMODE_TYPE,_UNSELECT,GET_OSD_DISPLAY_MODE());
       if(GET_OSD_DISPLAY_MODE() < _OSD_DM_4P)
       {
           OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_DISPLAYMODE_0);
       }
       else
       {
           OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_DISPLAYMODE_1);
       }

       // Selection Arrow Indication
       OsdDispMainMenuCursor(GET_OSD_STATE(), GET_OSD_STATE_PREVIOUS(),_OUTSUBSET);
   }
   // key info
   OsdDispMainMenuKeyInfo(_KEY_INFO_ALL, _OSD_UNSELECT);

   OsdDispSetPosition(_POS_PERCENT, _OSD_POSITION_GLOBAL_A, GET_OSD_HPOS(), GET_OSD_VPOS());

   // Osd Enable
   ScalerTimerWaitForEvent(_EVENT_DEN_STOP);
   OsdFuncEnableOsd();
```

注 1.菜单有二页，定义每页字符的起始地址；2.菜单的 map，菜单的长宽颜色等；3.菜单的透明度；4.菜单的图标及右上角 logo 的字库；5.右上角 logo 的 table；6.整个菜单的生成；7.图标及字体的颜色状态；8.菜单在画面的位置；

2. 主要介绍上面 6：整个菜单的画法，及图标及字体选择与选中的颜色变化;

这个函数 OsdDispMainMenuIconPage(_UP, _ICON_PAGE_MAIN_0);主要是画菜单的上面如图中 1 的主菜单的图标及字符，


```
case _ICON_PAGE_MAIN_0:
    OsdDispMainMenuIconString((ucUpDown + 0), _ICON_DISPLAYMODE,    pOsdItemColor[0]);
    if((GET_OSD_DISPLAY_MODE() == _OSD_DM_1P) || (GET_OSD_DISPLAY_MODE() == _OSD_DM_2P_LR) || (GET_OSD_
    {
        OsdDispMainMenuIconString((ucUpDown + 1), _ICON_DISPLAYFUNCTION,pOsdItemColor[1]);
    }
    else
    {
        OsdDispMainMenuIconString((ucUpDown + 1), _ICON_DISPLAYFUNCTION,_CP_GRAY);
    }

    if((GET_OSD_DISPLAY_MODE() == _OSD_DM_1P) || (GET_OSD_HLWIN_TYPE() != _HL_WIN_OFF))
    {
        OsdDispMainMenuIconString((ucUpDown + 2), _ICON_SELECTREGION,   _CP_GRAY);
    }
    else
    {
        OsdDispMainMenuIconString((ucUpDown + 2), _ICON_SELECTREGION,   pOsdItemColor[2]);
    }
    OsdDispMainMenuIconString((ucUpDown + 3), _ICON_PICTURE,        pOsdItemColor[3]);
```

这个函数 OsdDispMainMenuIconPage(_DOWN, _ICON_PAGE_DISPLAYMODE_0);主要是画图片中 2 的部分，它是_MENU_DISPLAYMODE 的第一页子菜单，同上面一样，我就不附图了。提下这个函数 OsdDispMainMenuArrow(_UP, _LEFT, _OSD_UNSELECT);主要是画左箭头右箭头；OsdDispMainMenuCursor(GET_OSD_STATE(),GET_OSD_STATE_PREVIOUS(),_OUTSUBSET);看这个函数的参量 osd 状态，与 osd 先前状态，大概就知道是未选中前 osd 颜色，与选中后的颜色，颜色变化肉眼就更加直观当前什么状态。

3.主要大概介绍图标及字符的添加，主要这个函数 OsdDispMainMenuIconString（）

```
void OsdDispMainMenuIconString(BYTE ucIconPos, WORD usIcon, BYTE ucColor)
{
    BYTE ucRow = 4, ucCol = 4;
    BYTE ucFontPage = _PFONT_PAGE_0;
    WORD usIconLoad = 0;

    ucRow = ((ucIconPos / 4) ? ROW(14) : ROW(4));
    ucCol = COL(4) + ((ucIconPos % 4) * 10);

    if(usIcon == _MENU_NONE)
    {
        OsdFuncClearOsd(ucRow, (ucCol - 2), WIDTH(10), HEIGHT(5));
        return;
    }

    // icon
    if((usIcon >= _ICON_A0_PORT) && (usIcon <= _ICON_D7_PORT))
    {
        switch(usIcon)
        {
            case _ICON_A0_PORT :
                usIconLoad = _A0_INPUT_TYPE;
                break;
            case _ICON_D0_PORT :
                usIconLoad = _D0_INPUT_TYPE;
                break;
            case _ICON_D1_PORT :
                usIconLoad = _D1_INPUT_TYPE;
                break;
            case _ICON_D2_PORT :
                usIconLoad = _D2_INPUT_TYPE;
                break;
            case _ICON_D3_PORT :
```

这里是分页，每一页左边到右边最多四个图标；

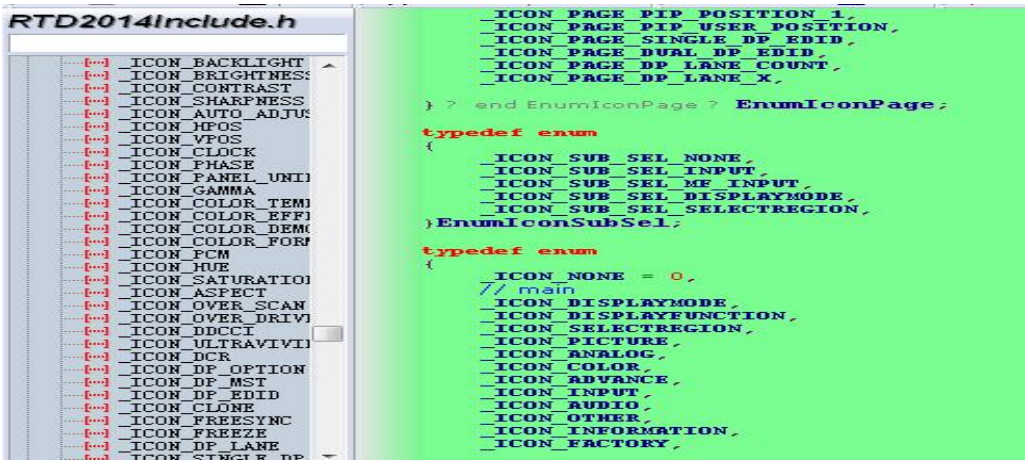
```
            case _ICON_D7_PORT :
                usIconLoad = _D7_INPUT_TYPE;
                break;
        } ? end switch usIcon ?

        OsdFontVLCDynamicLoadIcon(ucIconPos, usIconL...);
    } ? end if (usIcon>=_ICON_A0_POR... ?
    else
    {
        OsdFontVLCDynamicLoadIcon(ucIconPos, usIcon);
    }
    OsdFontPut1BitMainMenuIcon(ucRow, ucCol, (_1DYNAMIC_ICON_START + (ucIconPos * 24)), ucColor, _CP_BG);

    // string
    ucRow += 4;
    ucCol -= 2;
    OsdFuncClearOsd(ucRow, ucCol, WIDTH(10), HEIGHT(1));

    if((ucIconPos / 4) > 0)
    {
        ucFontPage = _FONT_PAGE_1;
    }

    if((usIcon >= _ICON_A0_PORT) && (usIcon <= _ICON_D7_PORT))
    {
        OsdPropPutStringCenter( ucRow, ucCol, WIDTH(10), ucFontPage, (usIcon-_ICON_INPUT_END +_STRIN_END), COLOR(ucColo
    }
    else
    {
        OsdPropPutStringCenter( ucRow, ucCol, WIDTH(10), ucFontPage, usIcon-1, COLOR(ucColor, _CP_BG), _ENGLISH);
    }
} ? end OsdDispMainMenuIconString ?
```

注，1.load 的是 source 的图标，2.是 load 菜单的主要图标；3.主要是定义图标的大小，6x4，因为所有图标都一致大小，所以定义一个函数直接调用；4.主要是字符的调用；

4. 主要细加介绍图标的添加字库的添加，函数 OsdFontVLCDynamicLoadIcon（），及 OsdPropPutStringCenter（）：

首先在 tFONT1_MAIN_ICON_TABLE_1[]中添加定义比如：tFONT1_ICON_DISPLAYMODE（显示模式的图标），在这个函数之前定义如下字库，在相应的头文件中定义相应 icon，定义要按顺序，不然菜单显示的字符容易错，（字符怎么来的可以查看另一个文档，很详细），

```
BYTE code tFONT1_ICON_DISPLAYMODE[]=
{
    0x0f,0x81,0x3c,0x7e,0x6d,0x24,0x59,0xab,0x01,0x6d,0x00,0x90,
    0x40,0x92,0xe9,0x24,0xc9,0xce,0x92,0x24,0x45,0x52,0x45,0x52,
    0x45,0x52,0x00,0x90,0x08,0x49,0x92,0x24,0x49,0x42,0x24,0x00,
    0x00,0x00,0x40,0x22,0x24,0x49,0x92,0x24,0x89,0x8e,0x44,0x37,
    0x74,0x43,0x37,0x00,0x20,0x11,0x92,0x24,0x49,0x92,0x84,0x54,
    0x82,0x2a,0xa8,0x82,0x2a,0x00,0x90,0x08,0x49,0x92,0x24,0x49,
    0x42,0x24,0x00,0x00,0x00,0x40,0x08,0x59,0x95,0x24,0x39,0x49,
    0x26,0x91,0x74,0x42,0x77,0x42,0x77,0x42,0x17,0x49,0x15,0x49,
    0x15,0x49,0x15,0x49,0x15,0x49,0x15,0x49,0x15,0x49,0x15,0x49,
    0x15,0x49,0x01,0x40,0xaa,0xda,0x4c,0xd5,0x4c,0xa6,0x8a,0x99,
    0x2a,0x66,0xaa,0x98,0xa9,0x62,0xa6,0x8a,0xd1,0x0d,0xdd,0xd0,
    0x0d,0xdd,0xd0,0x0d,0xdd,0xd0,0x0d,0xdd,0xd0,0x0d,0x54,0x41,
    0x15,0x54,0x31,0x53,0x85,0xa9,0x82,0x2a,0xa8,0x82,0x2a,0xa8,
    0x02,0x00,0x29,0xd9,0x67,0xd6,0xaa,0xa4,0xd7,0xa2,0x7b,0x2d,
    0x7a,0xd7,0x62,0x0e,0x2d,0xc5,0x9e,0xd5,0x09,0xdd,0x09,0xdd,
    0x09,0xdd,0x09,0xdd,0x09,0xdd,0x09,0xdd,0x09,0xdd,0x09,0xdd,
    0x09,0x5d,0x24,0x55,0x24,0x55,0x24,0x55,0x24,0x55,0x24,0x55,
    0x24,0x55,0x24,0x55,0x24,0xb5,0x48,0x6a,0xaa,0x98,0xa4,0xb6,
    0x49,0xb2,0x77,0x00,0x00,0x00,0x00,0x40,0x37,0x74,0x2f,0x74,
    0xaf,0x45,0x37,0x74,0x43,0x37,0x74,0x43,0x37,0x74,0x43,0xab,
    0x62,0xa8,0x8a,0x99,0x2a,0xa8,0x82,0x2a,0xa8,0x82,0x2a,0xa8,
```



字符的添加方式与图标的类似，在这个函数 OsdPropPutStringCenter(ucRow, ucCol, WIDTH(10), ucFontPage, usIcon-1, COLOR(ucColor, _CP_BG), _ENGLISH);行，列，字符串的宽度（有时候字符串长度太长了也会出现字符显示错误），页数，string，颜色，语言。OsdPropGetStringTableAddress(ucString)在这个函数中添加字符的地址，首先也要相应的定义如下 1 图。例如：tSTRING_DISPLAYMODE[] = { _D_, _i_, _s_, _p_, _l_, _a_, _y_, ___, _M_, _o_, _d_, _e_, _END_,}，每个字符也要定义宽度，如下 2 图。tFONT_EUROPE[],这是 load 英文的字库，一般包括 a-z,及一些符号，画了多少个字符，定义字符宽度是也要相应的数量，不然菜单字符串出错。

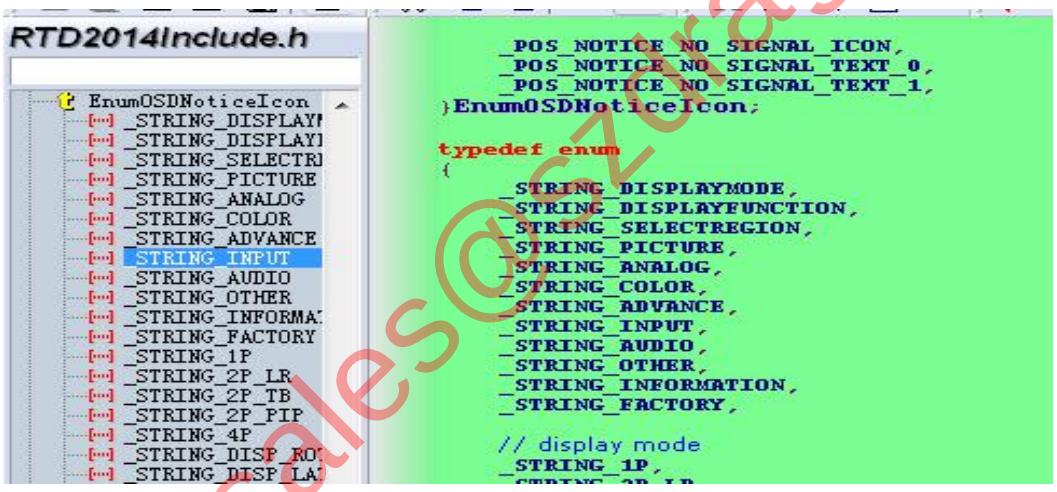


图 1

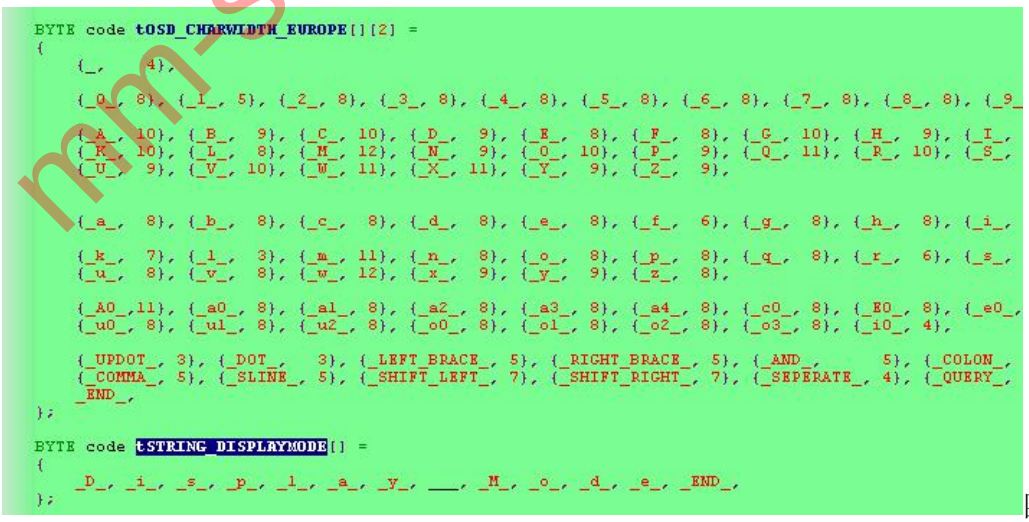


图 2

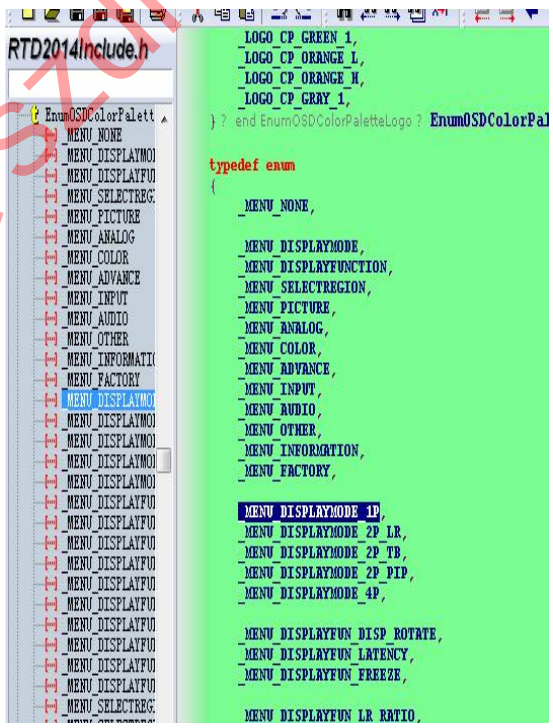
```
//
BYTE code tFONT_EUROPE[] =
{
    //=========Address -- (00),
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,
    0x00,0x00,0x00,

    //=========Address -- (01),
    0x00,0x00,0x00,
    0x03,0x08,0x00,
    0x7e,0xce,0x00,
    0xcc,0x66,0x00,
    0xcc,0x66,0x00,
    0xcc,0x66,0x00,
    0xe7,0xec,0x00,
    0x30,0x80,0x00,
    0x00,0x00,0x00,

    //=========Address -- (02),
    0x00 0x00 0x00
```

图 3

5. 相应的其他的各项一样的步骤，再介绍一些刚接触菜单时减少错误的细节：

1），OperationTable[]左边对应菜单的每一项，右边定义是要按着左边的顺序定义，要一一定义，不能少项,还有各个函数添加时也要进行相应的定义等等不然都会导致出错，具体可以慢慢摸索；

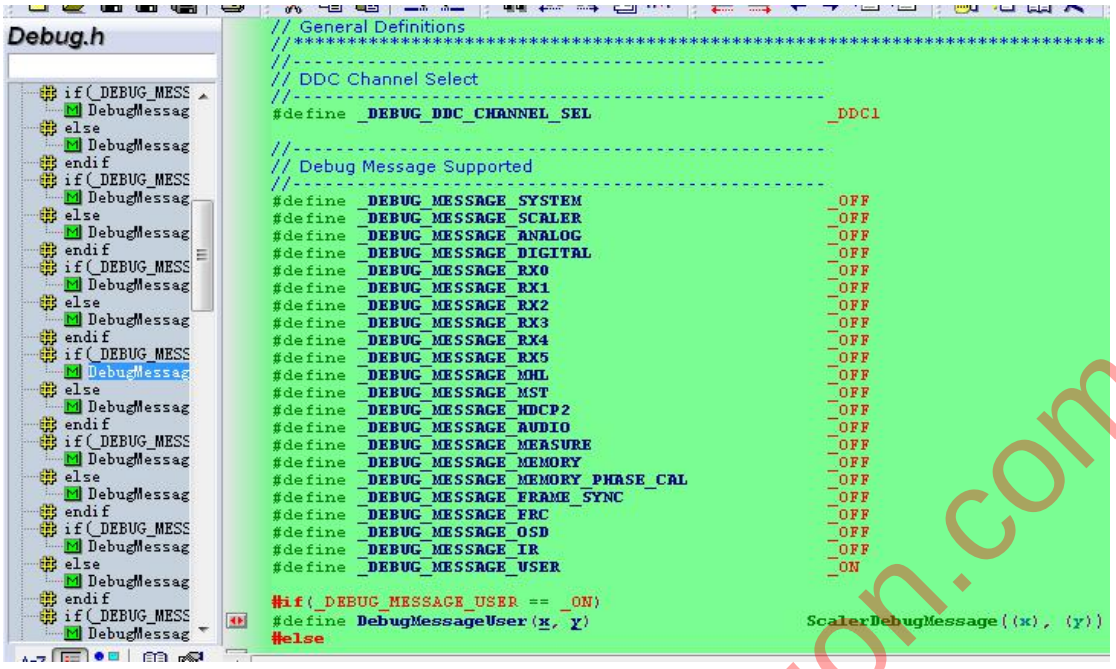
```
code void (*OperationTable[])(void) =
{
    MenuNone,

    // main
    MenuDisplayMode,
    MenuDisplayFunction,
    MenuSelectRegion,
    MenuPicture,
    MenuAnalog,
    MenuColor,
    MenuAdvance,
    MenuInput,
    MenuAudio,
    MenuOther,
    MenuInformation,
    MenuFactory,

    // display mode
    MenuDisplayMode1P,
    MenuDisplayMode2P_LR,
    MenuDisplayMode2P_TB,
    MenuDisplayMode2P_PIP,
    MenuDisplayMode4P,

    // display function
    MenuDisplayFunDispRotate,
    MenuDisplayFunLatency,
    MenuDisplayFunFreeze,
```



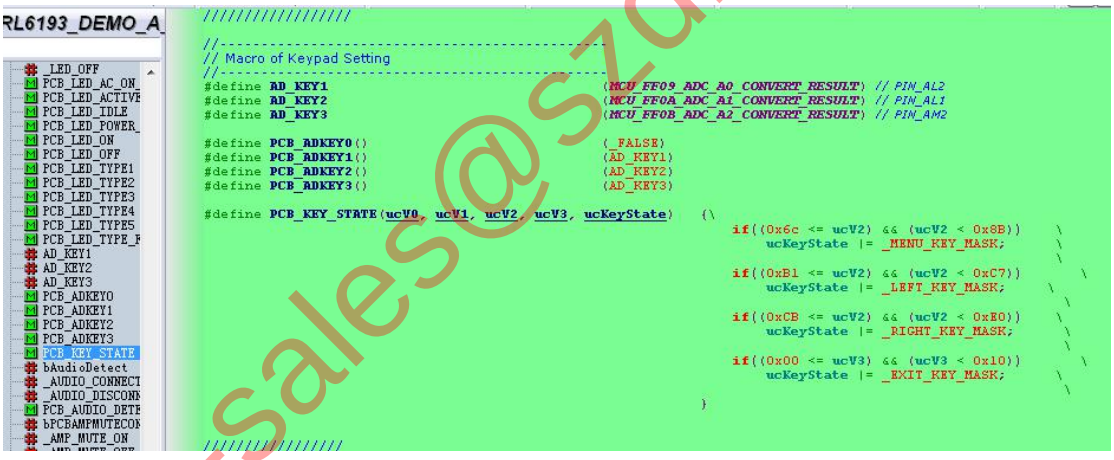
2）一般 用 SET_OSD_STATE（）来改变上一个变量传递到下一个变量，例如； menu none 时设的是这个状态 SET_OSD_STATE(_MENU_DISPLAYMODE);按右键设的是这个状态：SET_OSD_STATE(_MENU_PICTURE);

3）按键的配置：UserInterfaceKeyHandler();首先在这个函数中找到 RTDKeyScan()这个函数，PCB_KEY_STATE(ucVoltage0, ucVoltage1, ucVoltage2, ucVoltage3, ucKeyState);这个就可以配置每个按键的值，首先打开打印口，通过打印消息来确定每一个按键的值，一般我们可以自己

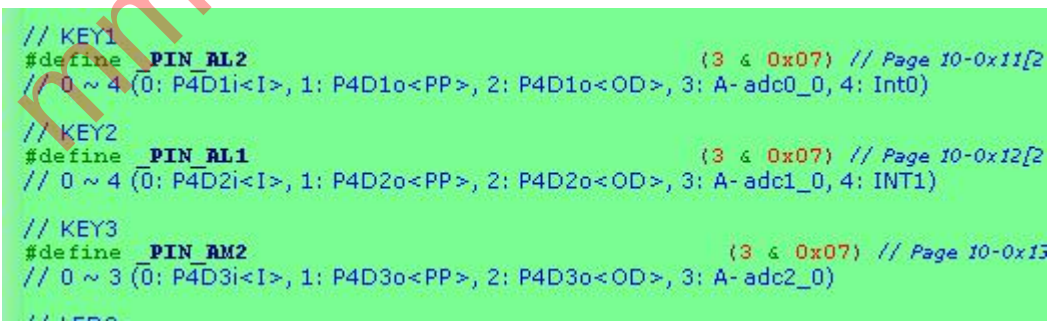
添加一个打印_DEBUG_MESSAGE_USER ,打印起来会比较直观，



在选的 pcb 中添加相应按键值的范围：比如按键 menu 的打印出来值为 ox70,如下，只要这个字在你给的范围就会执行 menu 的命令，但是同一电压如 ucv2 下，范围不能互相嵌套，比如 menu 范围是(0x6c <= ucV2) && (ucV2 < 0x8B)， left 为（0x80 <= ucV2) && (ucV2 < 0xC7) 就不行了。



芯片的引脚也要内部设为对应的 adc0,1,2.如下图;



大致的菜单画法就这些步骤，个人觉得菜单不是一接触就能不出问题的画好，只有不断出问题不断解问题才能完成的，我相信你够聪明的脑袋能解决。

五．RTD20140sdFunc.c 中各函数的简介

RTD20140sdFunc.c 中共 20 个函数组成：

1. OsdFuncApplyMap(BYTE ucWidth,BYTE ucHeight, BYTE ucColor)

自己在程序中搜索下会发现在画菜单，画 osd information，画 logo 等都会出现这个函数，这个函数就是画的 osd 的 map，比如：OsdFuncApplyMap(WIDTH(36),HEIGHT(8),COLOR(_CP_WHITE,_CP_BG));宽为 36，高为 8.颜色为白色

2. VoidOsdFuncSetPosition(EnumOsdPositionType ,enumOsdPositionType, WORD usX, WORD usY);

看命名，顾名思义，set position,就是画菜单，画 osd information，画 logo 放在屏的哪个位置；

3. void OsdFuncTransparency(BYTE ucTrans);

同理顾名思义 Transparency 透明度，就是菜单，information 等 osd 的透明度；

4. void OsdFuncBlending(BYTE ucType);

uctype 类型如下{_OSD_TRANSPARENCY_DISABLE = 0x00,
_OSD_TRANSPARENCY_ONLY_WINDOW = _BIT2,
_OSD_TRANSPARENCY_ALL = _BIT3,
_OSD_TRANSPARENCY_WINDOW_AND_CHARACTER_BACKGROUND = (_BIT3 | _BIT2)}根据字面意思应该就能知道是透明的类型，所有的都透明，还是只是 window 透明等。

5. BYTE OsdFuncGetOsdFontPointer(BYTE ucPointerSelect)与 BYTEOsdFuncSetOsdFontPointer(BYTEucPointerSelect,BYTEucPointe

r);是一起的，Get 与 Set 获得与设置，主要是设置菜单 page start(比如 diaplay mode 有 page1,page2,),item start,(每一页又有很多项)，每个 item 占有若干字符，第一项占用 3 个字节，那么第 2 项开始就至少要为 4，这样来防止乱码；

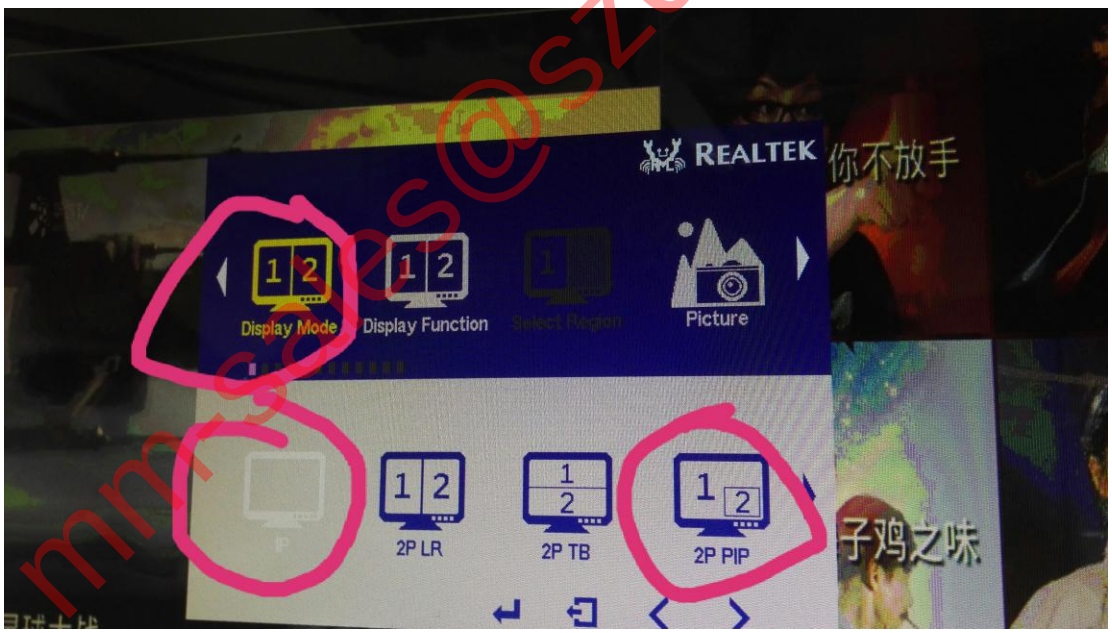
6. voidOsdFuncCloseWindow(EnumOsdWindowsType enumWinIndex);

7. void OsdFuncDisableOsd(void)和 void OsdFuncEnableOsd(void);

按字面意思理解即可；

8. void OsdFuncChangeIconColor1Bit(BYTE ucRow, BYTE ucItem, BYTE ucWidth , BYTE ucHeight, BYTE ucColor);

同样按字面意思改变 icon（图标为 1bit）如下图圈起来的各个选中与没选中的的图标颜色；



9. void OsdFuncChangeColor1Bit(BYTE ucRow, BYTE ucCol, BYTE ucWidth , BYTE ucHeight, BYTE ucColor, BYTE ucFontSelectFrom);

这个函数包含在上一个函数里面，真正改变颜色的这个函数；

10. void OsdFuncSet2BitIconOffset(BYTE ucOffset);

字面意思 2bit icon，但目前在我们程序中用不到，我们一般图标为 1bit;

11. void OsdFuncClearOsd(BYTE ucRow, BYTE ucCol, BYTE ucWidth , BYTE ucHeight);

这个主要用来清理某行某列的字符或图标，去除替换前的字节或图标，

12. void OsdFuncSixColorAdjust(void);

主要是色彩效果（color effect）的六种颜色（红橙黄绿蓝紫）的调节；

13. void OsdFuncColorFormatAdjust(void);

主要是 color format（色彩格式）如下：

_COLOR_SPACE_RGB,
_COLOR_SPACE_YCBCR422,
_COLOR_SPACE_YCBCR444,
_COLOR_SPACE_YCBCR420,
_COLOR_SPACE_YPBPR,
_COLOR_SPACE_DP_YONLY,