打造你的个性化界面

（ 本示范教程是以一个第三方作者的认知来编写的，并非源程序作者，对于大部分功能还在摸索中，由此文中肯定存在错误的地方，敬请包涵。）

示波器的应用对于一个电子工程师来说，是家常便饭，但要是能自己定义一个自己个性化的示波器显示界面，那是很多人都没有尝试过，经济型袖珍示波器 **DIY** 套件以及开源的源程序提供了一很好的实验环境,使得没有经过专业软件培训的爱好者，亦能设计出一个出色的画面。

1. 环境

在动手之前应备好下列开发环境及器材：

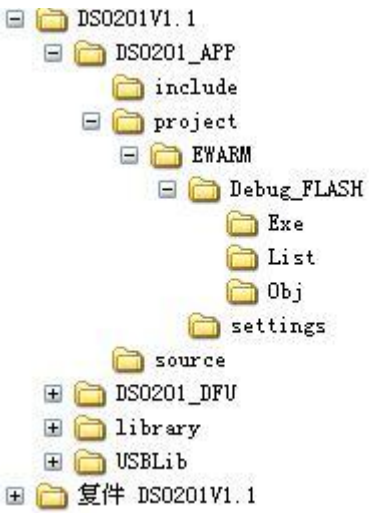
- 经济型袖珍示波器 **DIY** 套件；
- 开源的原程序；
- 编译器 **(**推荐使用 **IAR Embedded Workbench IDE 4.42** 注：**32k** 以上版本**)**；
- 固件文件管理器 （推荐使用 **DFU file Manager v2.2**）
- .

2. 基础概念

经济型袖珍示波器 **DIY** 套件提供

- **1** 个 **320x240** 的显示屏幕；
- **1** 个**<10Md** 信号输入口；
- **6** 个功能按键；
- **1** 个 **USB** 接口；

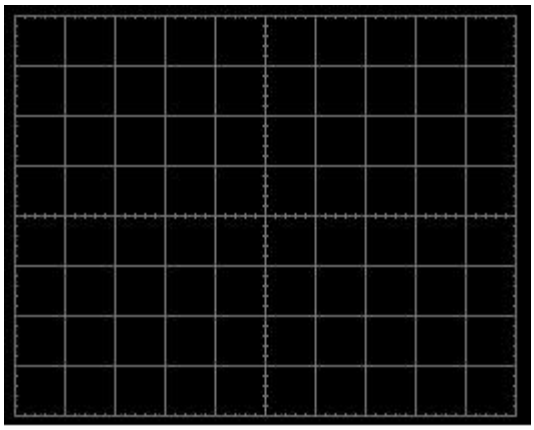
程序清单



在以后的文中，有出现☆★符号的，就是可以做修改的地方。

3. 背景网格

示波器的背景网格是由带有刻度线段组成 一般为 **8x10** 个网格，每个网格显示 **25x25** 像素，图中的示波器背景网格占用了显示屏的 **300x200** 的画面，如果为了腾出更多的地方显示其他信息，可以把示波器背景网格定为 **250x200** 或其他大小，但应该是 **50xn** 的参数；



示波器网格

☆★

在源程序的 **FUNction.h** 内可以找到下列内容,修改内容可以特定位置显示出背景网格。

```
#define X_SIZE    250
#define Y_SIZE    200
#define MIN_X     3
#define MIN_Y     24
#define MAX_X     (X_SIZE + MIN_X)
#define MAX_Y     (Y_SIZE + MIN_Y)
```

4. 显示字库

☆★

在源程序的 **Lcd.c** 内可以找到下列内容,修改内容可以显示不同的字符，亦可以画出自己喜欢的符号。

```
//=================== 字库定义 ================================
unsigned      const      short      Char_Dot[760]      =//744              12x6
0x0000,0x0000,0x0000,0x001C,0x0020,0x0040,/*" 左上角*/
0x0040,0x0040,0x0020,0x001C,0x0000,0x0000,/*# 右上角*/
0x0000,0x0000,0x0000,0xE000,0x1000,0x0800,/*$ 左下角*/
0x60C0,0x9300,0x6D80,0x3240,0xC180,0x0000,/*%    */
……
```

当然，你可以更换一个不同点阵的字符

5. 字符串显示函数

☆★

在源程序的 **Lcd.c** 内可以找到下列内容,使用不同的点阵的字库要修改下列内容。

```
/*****************************************************************************
 Display_Str: 在指定位置显示字符串    输入: X、Y 坐标，颜色值，显示模式，字符串
*****************************************************************************/
void Display_Str(unsigned short x0, unsigned short y0, unsigned short Color,
unsigned char Mode, unsigned const char *s)
{ 。。。。。

  while (*s!=0) {
    unsigned const short *scanline=Char_Dot+((*s-0x22)*6);  // 蓝色字为字符像素
宽度
    for(i=0;i<6;++i){ // 蓝色字为字符像素宽度
。。。。

      for(j=0;j<12;++j){     // 蓝色字为字符像素高度
        if(b&16) {// 蓝色字为去掉字符的 0X0000 部分（1，2，4，8，16）
          。。。
 if(*s==0x21) x0 +=3;  //显示位置水平方向+3 蓝色字为字符像素宽度/2,实现中置显示
效果
    else  x0 += 6;       //显示位置水平方向+6   蓝色字为字符像素宽度
 。。。。。。
```

☆★

字符的显示函数 ，熟悉这项内容可以在屏幕打印你喜欢的内容和符号。

```
Display_Str(unsigned short x0, unsigned short y0, unsigned short Color, unsigned
char Mode, unsigned const char *s)
```

- unsigned short x0：X 坐标 （0~319）
- unsigned short y0：Y 坐标，(0~239)
- unsigned short Color：颜色值 （请参考下文的颜色系统）
- unsigned char Mode：显示模式 （PRN：正常，INV 反显，Type 交替）
- unsigned const char *s：字符串 （参照字库的位置）

☆★

画点函数 ，熟悉这项内容可以在屏幕打印你喜欢的线段和点。

```
Point_SCR(x, y);        //x:轴坐标，y：y 轴坐标
Set_Pixel(color);       //在上述坐标内设置该像素的颜色
```

6. 菜单显示

在系统内已将实在收到的按键信号放到 Key_Buffer 内。按不同的按键产生不同的键值。

- KEYCODE_PLAY
- KEYCODE_LEFT
- KEYCODE_RIGHT
- KEYCODE_DOWN
- KEYCODE_UP

I KEYCODE_MANU

☆★

在源程序的 main.c 内可以找到下列内容,修改内容可以定义不同的菜单风格
菜单是由一个条件循环组成

```
Switch(Item) {
   case SYNC_MODE:  //菜单名
       if(Key_Buffer==KEYCODE_LEFT)  Item=Y_VERNIER_2;  //指向上一菜单
       if(Key_Buffer==KEYCODE_RIGHT) Item=Y_SENSITIVITY  //指向下一菜单
       if(Key_Buffer==KEYCODE_DOWN){            //要实现的操作
                if  ……
       if(Key_Buffer==KEYCODE_UP){ ……//要实现的操作
       ……          }
       break;

     case Y_SENSITIVITY:
       break;
}
```

☆★

在源程序的 Function.h 内可以找到下列菜单项声明,具体的函数在 Function.c 内，但不建议修改，除非你对系统和硬件结构非常熟悉，但建议动手做些新的功能菜单追加上去，领悟自己动手的乐趣。

```
    #define SYNC_MODE           0
    #define Y_SENSITIVITY           1
    #define X_SENSITIVITY           2
    #define Y_POSITION              3
    #define MEASUR_KIND         4
    #define POWER_INFOMATION 5
    #define TRIG_SENSITIVITY    6
    #define TRIG_SLOPE              7
    #define INPUT_ATTENUATOR    8
    #define SAVE_WAVE_CURVE     9
    #define LOAD_WAVE_CURVE     10
    #define OUTPUT_FREQUENCY    11
    #define X_VERNIER_2         12
    #define X_VERNIER_1         13
    #define X_POSITION              14
    #define RUNNING_STATUS      15
    #define DELTA_T             16
    #define Y_VERNIER_2         17
    #define Y_VERNIER_1         18
    #define TRIG_LEVEL          19
```

```
#define VERNIERS                20
#define WINDOW_AREA         21
```

7. 颜色系统

☆★

在源程序的 **Lcd.h** 内可以找到下列显示颜色定义声明,修改内容定义，可以产生不同的颜色，这样你的个性化界面会显得与众不同、绚丽多彩!

```
#define WHITE       0xFFFF  //白色：B = F800, G = 07E0, R = 001F
#define PANEL       0xFFE0  //青色：B = F800, G = 07E0, R = 0000
#define RED     0x001F  //红色：B = 0000, G = 0000, R = 001F
#define GRN     0x07E0  //绿色：B = 0000, G = 07E0, R = 0000
#define YEL     0x07FF  //黄色：B = 0000, G = 07E0, R = 001F
#define GRID    0x738E  //灰色：B = 7000, G = 0380, R = 000E
#define CURVE       0x0F8E  //亮绿：B = 0001, G = 0780, R = 000E
#define MODEL   0xC05E  //紫色：B = C000, G = 0040, R = 001E
#define LINE    0xE79F  //白色：B = E000, G = 0780, R = 001F
#define BLACK       0x0000  //黑色：B = 0000, G = 0000, R = 0000

Point_SCR(ti-2, MAX_Y-1);Set_Pixel(BLACK);
```

8. 初始值定义

☆★

在源程序的 **Function.h** 内可以找到下列显示颜色定义声明,修改内容定义,可以产生不同的颜色，这样你的个性化界面会显得与众不同、绚丽多彩!

```
//------------------------------开机时的初始值定义
-----------------------------------------------
//                          X 档 -- 极性 保存 读取 Fout -- -- X.POS -- --
-- -- --
unsigned short Item_Index[20]={  0,  //模式: 0: AUTO, 1: NORM, 2: SIGN, 3:
NONE
                                 6,  //Y 档: 从 10mV~ 100V
                                 8,  //x 档: 从 1uS~10S
                                 0,  // (未知)
                                 0,  //计算
                                 4,  //电源模式
                                 8,  //触发灵敏度
                                 0,  //触发边缘 0:上升，1：下降
                                 0,  //探头衰减 0：x1 ,1:x10
                                 1,  //当前保存文件指针 从 file000~File255
                                 1,  //当前读取文件指针 从 file000~File255
```

```
                                9,  //输出参考波形频率 从10Hz~1MHz
                                0,  //(未知)
                                0,  //(未知)
                                4096,  //(未知)
                                0,  //(未知)
                                0,  //(未知)
                                0,  //(未知)
                                0,  //(未知)
                                0};  //(未知)

unsigned short
        vt=140,   //触发电平
        v0=69,   //基线电平
        v1=199,  //上游标线
        v2=40,   //下游标线
        t0=150,  //同步触发点
        t1=68,   //左游标线
        t2=233;  //右游标线

unsigned char
        Hide_vs=1,  //触发灵敏度参考线 0=显示，1=消隐
        Hide_vt=0,  //触发电平参考线 0=显示，1=消隐
        Hide_v0=1,  //基线电平参考线 0=显示，1=消隐
        Hide_v1=0,  //上游标参考线 0=显示，1=消隐
        Hide_v2=0;  //下游标参考线 0=显示，1=消隐
 unsigned char
        Hide_t1=0,  //左游标参考线 0=显示，1=消隐
        Hide_t2=0,  //右游标参考线 0=显示，1=消隐
        Hide_Ref=1, //参考波形曲线 0=显示，1=消隐
        Item=2;    //开机时菜单位置
```