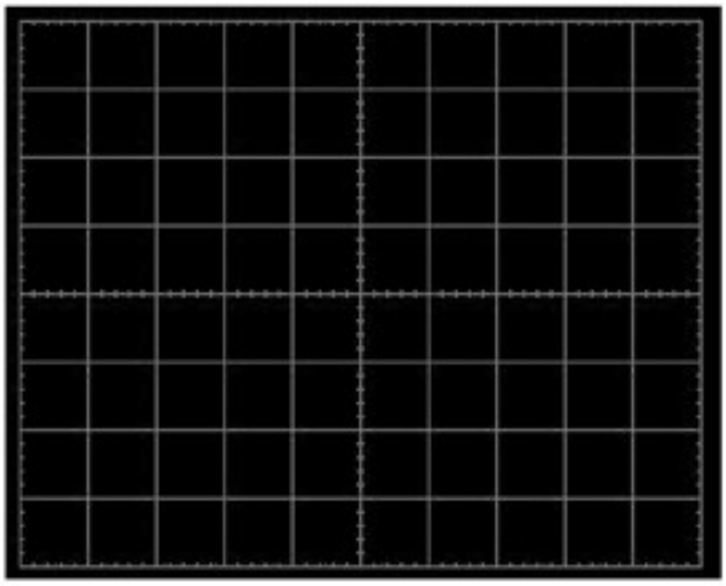
# Customizing DSO Nano User Interface

DSO Nano is open source based, customizing the user interface is much easier than you thought. By linking the new UI to default variables and constants, you can build a individual interface yourself.

## Grid

The background of the DSO is grid , it help user to measure the waveform on the screen.Now the DSO have 8X10 grids, and each grid is 25X25 pixel. The entire grid is 300X200 pixels, here is where the screen display the waveform. To make more space for display another information, you can modify the grid into smaller size, but it need to be the times of 50 pixel.



You can modify the grid in the FUNction.h . Just change the definition of the below you can move the grid to any position and change its size:

```
#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)
```

## Character

DSO has a font library in Lcd.c , just modify the font library you can change another typeface of character to display , and you can add more figure into the library to display , like ”→”.

```
    //===================          font          library
=====================================
    unsigned const short Char_Dot[760] =//744  12x6
    0x0000,0x0000,0x0000,0x001C,0x0020,0x0040,/*" */
```

```
0x0040,0x0040,0x0020,0x001C,0x0000,0x0000,/*# */
0x0000,0x0000,0x0000,0xE000,0x1000,0x0800,/*$ */
0x60C0,0x9300,0x6D80,0x3240,0xC180,0x0000,/*% */
```

## Character string display function

You can find this function in the Lcd.c and use it you can show a string on the screen with any color and on any position.

```
/***********************************************************************
 Display_Str: display the string in assigned position
 Input: X,Y,Color,Displaymode,string
***********************************************************************/
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);
    for(i=0;i<6;++i){
 ……..
      for(j=0;j<12;++j){
        if(b&16) {
 ……….
             if(*s==0x21) x0 +=3;
        else  x0 += 6;
 ……..
```

unsigned short x0：X coordinate （0~319）
unsigned short y0：Y coordinate，(0~239)
unsigned short Color：color （will be reference in next section）
unsigned char Mode：display mode （PRN: normal; INV :Invert）
unsigned const char *s：String

## Dot

You can use the two functions to drop a dot with any color in any position.

```
Point_SCR(x, y);    // X coordinate （0~319） Y coordinate，(0~239)
Set_Pixel(color);   //drop a color in the pixel set above
```

## Menu

The entire button signal is put into the Key_Buffer. And you can read the key value to judge which key be pressed.

- KEYCODE_PLAY
- KEYCODE_LEFT
- KEYCODE_RIGHT
- KEYCODE_DOWN
- KEYCODE_UP
- KEYCODE_MANU

In the main.c , you can fine the code below. And modify the code here you can define a new menu for your DSO. The menu is made up by a loop.

```
Switch(Item) {
  case SYNC_MODE:  //Menu name
```

```
   if(Key_Buffer==KEYCODE_LEFT)  Item=Y_VERNIER_2;  //to forward menu
   if(Key_Buffer==KEYCODE_RIGHT) Item=Y_SENSITIVITY  //to next menu
   if(Key_Buffer==KEYCODE_DOWN){          //the operation
           if  ……
   if(Key_Buffer==KEYCODE_UP){ ……//the operation
   ……        }
      break;

     case Y_SENSITIVITY:
      break;
}
```

In the Function.h you can fine the define for every menu, the operation function is in the Function.c. You can add the new menu to create new function for you DSO .

```
#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
```

## Color

In the Ldc.h you can fine the definition for every color, just modify the definition you can change the color, and you can add the color define here to make your interface more colorful.

```
#define WHITE    0xFFFF  //wite: B = F800, G = 07E0, R = 001F
#define PANEL    0xFFE0  //panel: B = F800, G = 07E0, R = 0000
#define RED    0x001F  //red: B = 0000, G = 0000, R = 001F
#define GRN    0x07E0  //green: B = 0000, G = 07E0, R = 0000
#define YEL    0x07FF  //yellow: B = 0000, G = 07E0, R = 001F

#define GRID    0x738E  //gray: B = 7000, G = 0380, R = 000E
#define CURVE    0x0F8E  // green: B = 0001, G = 0780, R = 000E
#define MODEL  0xC05E  // purple: B = C000, G = 0040, R = 001E
#define LINE    0xE79F  // white: B = E000, G = 0780, R = 001F
#define BLACK    0x0000  //black: B = 0000, G = 0000, R = 0000
```

For example:

```
Point_SCR(2, 1);
Set_Pixel(YEL);
```

Will drop a yellow dot in the (2,1) pixel .
And modify the

```
#define GRID 0x07E0 //green
```

The grid will be change to green color.
Ok , now you can start working with your individual interface now. Have fun!

Regards
FreeZinG
01.06.2010