Cika Electrónica

Nota de Aplicación: CAN-030
Título: **Generación de bitmaps para SSD1332 y similares**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	08/06/05	

Les presentamos ahora una forma rápida de generar bitmaps para los nuevos displays gráficos color con tecnología OLED. En este caso, nos referimos al UG9664GFD, display de 96x64 pixels, basado en chips controladores SSD1332.

Software

Recordemos que la estructura interna de memoria del SSD1332 asigna un formato 3:3:2 para 8bpp y 5:6:5 para 16bpp. En este caso, utilizaremos solamente 16bpp, dado que la granularidad de color de 8bpp resulta muy limitada para bitmaps, ya que no se trata de 256 colores tomados de una paleta de 24-bits, como en VGA (8 bits para R, 8 para G, y 8 para B), sino de 256 colores fijos, con 3 bits para B y G y 2 para R.

Como adelantáramos en la CAN-029, si comparamos la estructura de memoria del display con la forma de guardar imágenes de 24 bits en formato BMP, veríamos que son muy similares, por ejemplo: BMP va de abajo a arriba y el display de arriba a abajo, por lo que la imagen se ve espejada verticalmente; BMP usa tres bytes (B, G, R)para la información de color y el display utiliza dos bytes o uno, según el formato. Además, BMP incluye un encabezado de 54 bytes.

Por consiguiente, para adaptar una imagen, debemos llevarla a la resolución deseada, espejarla verticalmente, salvarla en formato BMP y luego de descartar los 54 bytes del comienzo, procesar la paleta al formato del SSD1332 en el modo de color que queramos utilizar.

Esto es lo que haremos a continuación.

Desarrollo

El algoritmo que proponemos es simple, antes que eficiente, y lee un archivo en formato BMP, guardando otro sin el header con la corrección de colores. Recordemos que el BMP debe ser en 24 bits, una utilidad sin costo para convertir estos archivos es *Gimp* (GNU Image Manipulation Program).

```
fread(buffer,1,14+40,filein);                // descarta header, no hay paleta
for(i=0;i<64;i++){
        fread(buffer,1,3*96,filein);         // obtiene RGB para una línea
        for(j=0;j<96;j++){
                b=buffer[3*j];                // obtiene RGB para un pixel
                g=buffer[3*j+1];
                r=buffer[3*j+2];
                bb=b&0xF8;                    // recorta a 5B 6G 5R
                rr=(r>>3)&0x1F;
                gg1=(g>>5)&0x07;
                gg2=(g<<3)&0xE0;
                buffer[2*j]=bb|gg1;          // mueve al formato del SSD en 16-bit
                buffer[2*j+1]=gg2|rr;        // bbbbbggg gggrrrrr
        }
        fwrite(buffer,1,2*96,fileout);       // escribe una línea
}
```

Realizamos el procesamiento por líneas, para de alguna forma estructurar la operación y mejorar el acceso a archivos, simplemente. Los 16-bits de información de color corresponden a 5 bits de B en el primer byte que se envía al display (que asociamos al LSB) , 3 bits de G, y luego 3 bits más de G en el segundo byte, seguidos de 5 bits de R. El orden de los bits de color coincide con el esperado, es decir, el bit más significativo de color está en la misma dirección que el bit más significativo del byte. Simplemente leemos los 24-bits del BMP y los convertimos a los 16-bits del SSD, pixel a pixel, línea a línea

El programa terminado, es el siguiente:

```
#include <stdio.h>

int main (int argc, char *argv[])
{

FILE *filein,*fileout;
unsigned char buffer[3*96],r,g,b,rr,gg1,gg2,bb;
int i,j;

    if(argc!=3){
      fprintf(stderr,"usage: %s filein fileout\n",argv[0]);
      exit (-1);
    }

    if (!(filein=fopen(argv[1],"r"))) {
      fprintf(stderr,"%s: can not open %s\n",argv[0],argv[1]);
      exit (-1);
    }

    if (!(fileout=fopen(argv[2],"w"))) {
      fprintf(stderr,"%s: can not open %s\n",argv[0],argv[2]);
      exit (-1);
    }

        fread(buffer,1,14+40,filein);               // descarta header, no hay paleta
        for(i=0;i<64;i++){
                fread(buffer,1,3*96,filein);        // obtiene RGB para una línea
                for(j=0;j<96;j++){
                        b=buffer[3*j];               // obtiene RGB para un pixel
                        g=buffer[3*j+1];
                        r=buffer[3*j+2];
                        bb=b&0xF8;                   // recorta a 5B 6G 5R
                        rr=(r>>3)&0x1F;
                        gg1=(g>>5)&0x07;
                        gg2=(g<<3)&0xE0;
                        buffer[2*j]=bb|gg1;         // mueve al formato del SSD en 16-bit
                        buffer[2*j+1]=gg2|rr;       // bbbbbggg gggrrrrr
                }
                fwrite(buffer,1,2*96,fileout);      // escribe una línea
        }

    fclose(filein);
    fclose(fileout);
}
```

Si bien hemos desarrollado esto sobre Linux y compilado con *gcc*, no debería haber inconvenientes en portarlo a cualquier otro sistema