Cika Electrónica

Nota de Aplicación: CAN-034
Título: **Generación de textos en displays LCD color con controladores S1D13705 y Rabbit**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	24/06/05	

Veremos en esta nota una forma de generar caracteres al utilizar un controlador de Epson, el S1D13705, para que se encargue de todo lo relacionado con el manejo del display. Para una descripción, conexionado y rutinas de soporte del controlador, remítase a las Notas de Aplicación CAN-032 y CAN-033.

Algoritmo

Si trabajamos en 8bpp, para ubicar un punto en pantalla, calculamos su posición en memoria sabiendo que alojamos un pixel por byte, es decir: $mem = x + 320 \cdot y$.

Si en cambio trabajamos en 4bpp, para ubicar un punto en pantalla, calculamos su posición en memoria sabiendo que alojamos dos pixels por byte, es decir: $mem = \frac{x + 320 \cdot y}{2}$. El resto de esa operación (el bit menos significativo de $x + 320 \cdot y$) nos dirá qué nibble utilizar.

La forma más común (y bastante eficiente) de almacenar tipografías en memoria, consiste en agrupar los pixels "pintados" del caracter en bytes, en el sentido horizontal, es decir, un byte aloja ocho pixels que corresponden a la parte superior del caracter, de izquierda a derecha, de MSB a LSB. Si el ancho del caracter es mayor a dieciséis pixels, entonces se utilizarán grupos de dos bytes. Ésta es la forma en la que se alojan los fonts provistos con las libraries de Dynamic C, y con un simple algoritmo los podemos convertir para su uso con un controlador de displays color. Simplemente, chequearemos el estado de cada pixel, y si éste está pintado, lo coloreamos en el display. De igual modo, si no lo está, podemos utilizar un color de fondo, o dejarlo sin modificar.

Aprovechando que las fonts de Dynamic C están definidas como libraries, las podemos incluir automáticamente en xmem. Definiremos una simple estructura para guardar algunos parámetros de cada tipografía que nos permitan acelerar su impresión, estos datos los obtenemos observando el archivo que contiene la tipografía, que no es otra cosa que código fuente:

```
#use "6X8L.lib"
#use "12X16L.lib"

typedef struct {
       unsigned long *font;
       unsigned char  lpc;          // líneas por character
       unsigned char  Bpcl;         // bytes por línea de caracter (1 ó 2)
       unsigned char  bpcl;         // bits por línea de caracter
} FontInfo;

const static FontInfo fontinfo[]={
{&Font6x8,8,1,6},
{&Font12x16,16,2,12}
};
```

A continuación, veremos una rutina para imprimir un caracter en 8bpp:

```
void LCD_putchr ( int font, unsigned long daddress, char chr , int color, int bcolor)
{
int i,j,ii,jj;
unsigned long address,dispaddress;
int data,aux;
```

```
        i=fontinfo[font].lpc;                                   // líneas por character
        ii=fontinfo[font].Bpcl;                                 // bytes por línea de caracter
        jj=fontinfo[font].bpcl;                                 // bits por línea de caracter
        address=*(fontinfo[font].font)+i*ii*(chr-0x20);         // ubica caracter en tipografía
        dispaddress=daddress;                                   // apunta a display
        while(i--){
                data=xgetint(address);                          // obtiene 2 bytes de xmem
                aux=(data&0xFF)<<8;                             // swap bytes (MSB - LSB)
                data=((data>>8)&0xFF)+aux;
                address+=ii;
                j=jj;
                while(j--){                                     // rota a izquierda y chequea
                        if(data&0x8000)
                                write13705(dispaddress,color);        // pinta
                        else
                                if(bcolor>=0)                   // bcolor =-1 => transparente
                                        write13705(dispaddress,bcolor);
                        data<<=1;
                        dispaddress++;
                }
                dispaddress=daddress+=320;                      // apunta a línea de abajo
        }
}
```

A continuación, veremos una rutina para imprimir un caracter en 4bpp:

```
void LCD_putchr ( int font, unsigned long daddress, char chr , int color, int bcolor)
{
int i,j,ii,jj;
unsigned long address,dispaddress;
int data,ddata,aux;

        i=fontinfo[font].lpc;                                   // líneas por character
        ii=fontinfo[font].Bpcl;                                 // bytes por línea de caracter
        jj=fontinfo[font].bpcl;                                 // bits por línea de caracter
        address=*(fontinfo[font].font)+i*ii*(chr-0x20);         // ubica caracter en tipografía
        dispaddress=daddress;                                   // apunta a display
        while(i--){
                data=xgetint(address);
                aux=(data&0xFF)<<8;                             // swap bytes (MSB - LSB)
                data=((data>>8)&0xFF)+aux;
                address+=ii;
                j=jj;
                while(j--){
                        ddata=read13705((unsigned int)(dispaddress/2)); // lee display
                        aux=-1;
                        if(data&0x8000)                         // chequea si pinta
                                aux=color;
                        else
                                if(bcolor>=0)                   // bcolor =-1 => transparente
                                        aux=bcolor;
                        if(aux != -1){
                                if(dispaddress&1){              // ubica nibble
                                        ddata&=0xF0;            // borra bits a escribir
                                        ddata|=aux;             // setea color del pixel derecho
                                }
                                else {
                                      ddata&=0x0F;
                                      ddata|=(aux<<4);          // setea color del izquierdo
                                }
                        }
                        write13705((unsigned int)(dispaddress/2),ddata);
                        data<<=1;
                        dispaddress++;
                }
                dispaddress=daddress+=320;                      // línea siguiente
        }
}
```

En ambos casos, el cálculo de la dirección inicial en pantalla para cada caracter de un string, la realizamos de la siguiente forma:

```
void LCD_printat (int font,unsigned int row,unsigned int col,char *ptr,int color,int bcolor)
```

```
{
unsigned long address;

        address=320L*row;                                       // ubica dirección en display
        do {
                address+=col;
                LCD_putchr (font,address,*ptr++,color,bcolor);      // pone caracter
                col=fontinfo[font].bpcl;                            // siguiente
        }       while (*ptr);
}
```

Para escribir un texto, simplemente llamamos a esta rutina, teniendo cuidado de no excedernos en los límites útiles:

```
        LCD_printat(0,20,20,"Cika Electronica",15,6);
```

El software que acompaña a esta nota, incluye todas las rutinas de soporte para que pueda observar rápidamente estos ejemplos.