Cika Electrónica

Nota de Aplicación: CAN-076
Título: **Utilización de displays LCD gráficos (SED1335) con Ramtron VRS51L3074**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	19/03/07	port de CAN-005 con modificaciones

Nos interiorizaremos ahora en el desarrollo de una interfaz para conectar un módulo LCD gráfico inteligente Powertip PG320240, a un módulo Rabbit 2000. Se trata de un display de 320x240 pixels basado en chips controladores compatibles con el SED1335, de S-MOS, y su clon de Epson. Analizaremos más tarde el software de control y un simple programa demostración, que sirve para comprobar el correcto funcionamiento de los módulos LCD que tengamos en stock, y de paso, demostrar sus capacidades. A fin de probar la mayor parte posible del hardware, la interfaz será de 8 bits y realizará lectura y escritura del controlador LCD.

Hardware

El SED1335 presenta una interfaz con dos posibles modos de trabajo: tipo Motorola (E, RS, R/W) o tipo Intel (RD, WR, A0). El PG320240FRS de Powertip lo utiliza en esta última modalidad.

Para la interfaz con el micro, deberemos tener en cuenta el controlador que posee el display. Dado que el micro es de 3,3V y 5V-tolerant, podemos usar displays basados en S1D13305 y SED1335 sin problemas, dado que éstos toleran los niveles de tensión del VRS51L3074.

El port P0, hace las veces de bus de datos, mientras que algunos pines del port P2 generarán, por software, las señales de control.

El circuito de contraste de este display es totalmente interno, el mismo puede ajustarse mediante un preset ubicado en la parte posterior del display.

El display dispone, además, de un pin de reset, el cual podemos controlar a voluntad o conectar al reset del circuito. Para el desarrollo de esta nota de aplicación, simplemente lo conectamos mediante un pull-up a la tensión de alimentación.

Ramtron		LCD
P0.0	----------	D0
P0.1	----------	D1
P0.2	----------	D2
P0.3	----------	D3
P0.4	----------	D4
P0.5	----------	D5
P0.6	----------	D6
P0.7	----------	D7
P2.0	----------	RD
P2.1	----------	WR
P2.3	----------	A0
P2.4	----------	CS

Software

Breve descripción del display gráfico

Estos displays son sumamente versátiles, la memoria puede ser dividida en diferentes pantallas, las cuales, a su vez pueden definirse como gráficas o de texto, habilitarse y/o superponerse independientemente. Para el caso del modo texto, el SED1335 dispone de un generador de caracteres y una ROM de caracteres de 5x7, aunque es posible utilizar una ROM externa o la misma RAM del display. Al momento de definir cada pantalla, definimos también en qué posición de memoria comienza y cómo se asigna la memoria.

La estructura de memoria de cada pantalla es lineal, tanto en modo gráfico como en modo textol. En este último modo, el primer byte corresponde al caracter ubicado arriba a la izquierda, y el último byte corresponde al ubicado abajo a la derecha. En el modo gráfico, los pixels se agrupan horizontalmente en bytes, correspondiendo el primer byte de memoria a los primeros ocho pixels de la primera línea de arriba a la izquierda, y el último byte a los últimos ocho pixels de la última línea de abajo a la derecha. El bit más significativo del primer byte de memoria corresponde al punto situado en la pantalla arriba a la izquierda, y el bit menos significativo del último byte de memoria corresponde al punto situado en pantalla abajo a la derecha.

El direccionamiento del byte a leer o escribir en memoria se hace mediante comandos, especificando el offset desde la primera dirección correspondiente a la pantalla que nos ocupa. Tiene además un contador autoincrementado, el cual apunta a la dirección siguiente luego de una lectura o escritura. Esto resulta óptimo para enviar los datos byte por byte hasta completar una pantalla.

Una característica interesante del display, es que puede funcionar a una alta velocidad de acceso, simplemente da prioridad a la interfaz con el procesador e interrumpe el display. Si nuestra aplicación requiere que no haya parpadeo (flicker) al momento de escritura, podemos primero chequear el flag de ocupado (busy) y realizar nuestra escritura en el momento que el controlador no accede a la RAM para refrescar el LCD. Si, por el contrario, toleramos el flicker o, como en nuestro caso, hacemos una escritura muy rápida que el mismo no se nota, podemos obviar un paso y escribir directamente sobre el controlador sin chequear el busy flag.

Algoritmos

Si bien el display tiene muchas formas de utilización, en esta nota de aplicación desarrollaremos algoritmos en base a las más simples.

Para direccionar un punto debemos traducir sus coordenadas a una dirección lineal, para ello, deberemos multiplicar la coordenada vertical *y* por la cantidad de bytes en sentido horizontal de la pantalla (40 si utilizamos la totalidad de 320 pixels=40 bytes) y sumarle la coordenada horizontal *x* dividida por 8 (pixels por byte). El resto de dividir *x*/8 es el número de pixel dentro del byte. Dado que el MSB se halla a la izquierda, el pixel 0 corresponde al bit 7 y el pixel 7 al bit 0, es decir: *address=40*y+x/8 ; bit=7-resto(x/8).*

Para graficar funciones, debemos tener en cuenta que la coordenada (0;0) se halla en el extremo superior izquierdo de la pantalla.

Para mostrar pantallas, deberemos agrupar los datos de modo tal de poder enviarlos de forma que aproveche de manera eficiente los contadores autoincrementados y la estructura de memoria; dada la estructura lineal, esto se reduce simplemente a enviar todos los bytes corridos. Si comparamos la estructura de memoria del display con la forma de guardar imágenes blanco y negro 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 el '0' para el negro y el '1' para el blanco, si el display es STN+, '1' corresponde a un punto negro. Además, BMP incluye un encabezado de 62 bytes.

Por consiguiente, para adaptar una imagen, debemos llevarla a la resolución deseada, espejarla verticalmente, invertir sus colores, salvarla en formato BMP y por último descartar los 62 bytes del comienzo con algún editor hexa. En el caso de íconos, si la cantidad de bytes por línea no es par, BMP agregará bytes de padding, y según el programa que utilicemos para salvar, puede que la inversión de colores no funcione, puesto que sólo invierte la paleta, que no utilizamos. Dado que luego generaremos un header file, es bueno tener una opción en el programa de conversión para invertir el contenido de los bytes.

Para imprimir textos, calculamos simplemente la posición de memoria a partir de fila y columna de modo similar: *address=40*fila+columna*, para 40 caracteres por fila (matriz de caracteres de 8x8).

Desarrollo

Desarrollamos a continuación el software de base para manejo del display. Definiremos dos pantallas: una gráfica de 320x240 y una de texto de 40x30, con caracteres de 8x8, usando el generador interno, por lo que se verán caracteres de 5x7 en una trama de 8x8. Para una mejor comprensión, dada la complejidad del SED1335, se recomienda consultar su manual técnico.

```
#define LCD_RD  0
#define LCD_A0  3
#define LCD_WR  1
#define LCD_CS  4


void LCD_Write(unsigned char dat)
{
/* S1D13305: 0ns address setup, 120ns data setup, 120ns strobe width */

        P0=dat;                             // write data
        P0PINCFG=0;                         // P0 = Outputs, data en bus
        P2&=~((1<<LCD_WR)|(1<<LCD_CS));     // Baja WR+CS
        _asm
          DA A                              ; 100ns a 40Mhz
        _endasm;
        P2|=((1<<LCD_WR)|(1<<LCD_CS));      // Sube WR+CS
        _asm
          DA A
          DA A                              ; 400ns a 40Mhz
          DA A                              ; RET agrega más delay
          DA A
        _endasm;
```

```
}


void LCD_WriteCmd(unsigned char cmd)
{
       P2|=(1<<LCD_A0);              // Sube A0 (Cmd)
       LCD_Write(cmd);
       P2&=~(1<<LCD_A0);             // Baja A0 (Data)
}

void LCD_WriteStrCmd(unsigned char *cmd,unsigned char len)
{
       LCD_WriteCmd(*cmd++);
       while(--len) {
              LCD_Write(*cmd++);
       }
}


unsigned char LCD_Read()
{
/* S1D13305: 0ns address setup, 50ns data delay, 120ns strobe width */

unsigned char dat;

       P0PINCFG=0xFF;                         // P0 = Inputs
       P2&=~((1<<LCD_RD)|(1<<LCD_CS));       // Baja RD+CS
       _asm
         DA A                                 ; 100ns a 40Mhz
        _endasm;
       dat=P0;                                // lee del bus
       P2|=((1<<LCD_RD)|(1<<LCD_CS));        // Sube RD+CS
       _asm
         DA A
         DA A                                 ; 400ns a 40Mhz
         DA A                                 ; RET agrega más delay
         DA A
        _endasm;
       return(dat);
}

unsigned char LCD_ReadData()
{

unsigned char dat;

       P2|=(1<<LCD_A0);              // Sube A0 (Read Data)
       dat=LCD_Read();
       P2&=~(1<<LCD_A0);             // Baja A0
       return(dat);
}
```

Veamos ahora la función de inicialización, la cual es algo extensa dadas las prestaciones del SED1335 y su inherente complejidad.

```
void LCD_init ()
{
__code const static unsigned char init_string1[]={
0x40,                          // INIT
0x30,                          // 32 char CGRAM, 8 pixel chars, LCD, normal
0x87,0x7,                      // 8x8 chars
39,59,                         // 320 pixels, cracteres de 8 pixels, 40 por línea
239,                           // 240 líneas (30 de texto)
40,0                           // virtual screen = display screen
};

__code const static unsigned char init_string2[]={
0x44,                          // SCROLL
0,                             // SAD1L: dirección de inicio screen 1 (0x0000)
0,                             // SAD1H: (texto)
239,                           // SL1  : líneas de pantalla screen 1 (239)
0xB0,                          // SAD2L: inicio screen 2 (1200=0x4b0)
0x04,                          // SAD2H: (gráficos)
```

```
239,                        // SL2  : líneas de pantalla screen 2
0,                          // SAD3L:
0,                          // SAD3H:
0,                          // SAD4L:
0                           // SAD4H:
};

__code const static unsigned char init_string3[]={
0x5D,                       // CSRFORM
0x04,                       // CRX: tamaño horizontal del cursor (4 pixels)
0x86                        // CRY: tamaño vertical del cursor (6 pixels) CM: block
};

__code const static unsigned char init_string4[]={
0x5A,                       // HDOT_SCR (0x5a)
0                           // 1 pixel scroll
};

__code const static unsigned char init_string5[]={
0x5B,                       // OVLAY, layered screens, text/graphics.
0x01                        //MX0, MX1: text/graphics overlay XOR
};

__code const static unsigned char init_string6[]={
0x59,                       // DISP_ON
0x16                        // cursor flash rate = 2 Hz, SAD1/2 no flash, SAD3 off
};

       P0=0;
        P2|=0x1B;           // Salidas en alto
        P2PINCFG&=0xE4;     // xxx00x00 P2.0,1,3,4 = output

       LCD_WriteStrCmd ( init_string1,sizeof(init_string1) );
       LCD_WriteStrCmd ( init_string2,sizeof(init_string2) );
       LCD_WriteStrCmd ( init_string3,sizeof(init_string3) );
       LCD_WriteCmd ( 0x4C );       // CSRDIR_RIGHT
       LCD_WriteStrCmd ( init_string4,sizeof(init_string4) );
       LCD_WriteStrCmd ( init_string5,sizeof(init_string5) );
       LCD_WriteStrCmd ( init_string6,sizeof(init_string6) );
}
```

Rutinas de soporte de alto nivel, sumamente simples

```
void LCD_cursor ( unsigned int address )
{
       LCD_WriteCmd(0x46);                          // CSRW, dirección del cursor
       LCD_Write(address&0xFF);                     // LSB
       LCD_Write((address>>8)&0xFF);                // MSB
}

void LCD_fill(unsigned int address,unsigned char width,unsigned char height,unsigned char
pattern)
{
unsigned int w;

   while(height--){
     LCD_cursor(address);
     LCD_WriteCmd(0x42);                            // MWRITE
     w=width;
     while(w--)
       LCD_Write(pattern);                          // escribe patrón
     address+=40;
   }
}

#define LCD_cleargfx() LCD_fill(1200,40,240,0)
#define LCD_cleartxt() LCD_fill(0,40,30,' ')


void LCD_printat (unsigned char row, unsigned int col, char *ptr)
{
       LCD_cursor ((row<<5)+(row<<3)+col);          // 40*y+col (setea dirección)
       LCD_WriteCmd(0x42);                          // MWRITE
       while (*ptr)
```

```
                LCD_Write (*ptr++);
}


void LCD_plot (unsigned int x, unsigned char y)
{
unsigned char strip,bt,dat;
unsigned int addr;

        strip=(unsigned char)(x>>3);
                        // strip = x/8
        bt=(unsigned char)x & 0x07;
        // bit = remainder
        addr=(y<<5)+(y<<3);                          // 40*y
        addr+=1200+strip;
        LCD_cursor(addr);                            // direcciona byte
        LCD_WriteCmd(0x43);                          // MREAD
        dat=LCD_ReadData();                          // lee byte
        dat|=(0x80>>bt);                             // setea punto (MSB a la izquierda)
        LCD_cursor(addr);                            // direcciona otra vez
        LCD_WriteCmd(0x42);                          // MWRITE
        LCD_Write(dat);                              // escribe
}


void LCD_icon(unsigned int address,unsigned char width,unsigned char height,
__code unsigned char * imgdata)
{
unsigned int w;

   while(height--){
     LCD_cursor(address);
     LCD_WriteCmd(0x42);                             // MWRITE
     w=width;
     while(w--)
          LCD_Write(*(imgdata++));
     address+=40;
   }
}

#define LCD_dump(x) LCD_icon(1200,40,240,x)
```

También hemos portado la rutina de impresión de líneas basada en el algoritmo de Bresenham, la cual puede observarse en los archivos que acompañan a esta nota de aplicación.

Utilización

Inicializamos el display llamando a la función:

```
                LCD_init();
```

Limpiamos las pantallas de texto y gráfica con las siguientes macros:

```
                LCD_cleartxt();
                LCD_cleargfx();
```

que en realidad utilizan la función *LCD_fill()*, que puede utilizarse para borrar un ícono o un área de texto:

```
                    LCD_fill(addr,40,54,0);
```

o resaltar (invertir) un texto:

```
                LCD_printat(1,2,"superpuestos");
                LCD_fill(1200+8*40+2,12,8,0xFF);                              // invertir
```

los parámetros de esta función son: la dirección en memoria, el ancho en bytes (pixels/8), la altura en líneas, y el patrón a escribir.
Mostramos un ícono en pantalla con la siguiente función:

```
                    LCD_icon(addr,40,54,cika);
```

los parámetros de esta función son: la dirección en memoria, el ancho en bytes (pixels/8), la altura en líneas, y la dirección en flash de la imagen correspondiente.
Para mostrar una pantalla completa, podemos tratarla como un ícono o utilizar la siguiente macro:

```
LCD_dump(goku);
```

Para trazar líneas utilizamos:

```
LCD_line(0,120,319,120);
```

los parámetros de esta función son: las coordenadas x e y del primer punto, y las coordenadas x e y del segundo punto.
Para generar un rectángulo:

```
LCD_rectangle(0,0,319,239);
```

los parámetros de esta función son: las coordenadas x e y del vértice superior izquierdo, y las coordenadas x e y del vértice inferior derecho.

El siguiente es un ejemplo de un simple programa que muestra algunos íconos y texto en pantalla:

```
#include "goku.h"
#include "powertip.h"
#include "ramtron.h"
#include "ramtronb.h"
#include "pdm.h"
#include "lowlevel.h"
#include "highlevel.h"

main()
{

	LCD_init();
	while(1){
		LCD_cleartxt();
		LCD_cleargfx();
		MsDelay ( 1000 );                                   // 1 seg
		LCD_fill(1200,40,240,0xFF);                         // negro
		MsDelay ( 3000 );                                   // 3 segs
		LCD_cleargfx();
		LCD_icon(1200+40*99,40,42,ramtronb);
		MsDelay (3000);                                     // 3 segs
		LCD_cleargfx();
		LCD_line(0,120,319,120);                            // horizontal
		LCD_line(160,0,160,239);                            // vertical
		LCD_icon(1200+40*50,152/8,20,ramtron);              // 4 íconos en cuadrantes
		LCD_icon(1200+21+40*42,152/8,35,powertip);
		LCD_icon(1200+40*150,152/8,61,pdm);
		LCD_icon(1200+21+40*170,152/8,20,ramtron);
		LCD_rectangle(0,0,319,239);                         // borde
		MsDelay ( 6000 );                                   // 6 segs
		LCD_cleargfx();
		LCD_dump(goku);                                     // pantalla
		LCD_printat(0,0,"Texto y graficos");
		LCD_printat(1,2,"superpuestos");
		LCD_fill(1200+8*40+2,12,8,0xFF);                    // invierte texto
		MsDelay ( 5000 );                                   // 5 segs
		LCD_cleargfx();
		LCD_printat(12,9,"Cika Electronica S.R.L.");
		LCD_printat(15,9,"Master distribuidor de");
		LCD_printat(18,8,"componentes electronicos");
	}
}
```