Cika Electrónica

Nota de Aplicación: CAN-042
Título: **Conexión de módulos LCD con interfaz tipo Motorola a Rabbit 3000A**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	12/10/05	

El objeto de esta nota es demostrar una aplicación de una novedosa característica del Rabbit 3000A: IOSTROBE con CS activo en alto. La misma permite, entre otras cosas, controlar displays alfanuméricos inteligentes y displays gráficos inteligentes basados en chips controladores compatibles con el HD61202, de Hitachi; como por ejemplo Powertip PG12864, de 128x64 pixels.

Hardware

Vamos a aprovechar varias características del R3000A:

- ➔ Conexión al bus: el display mapea dentro del espacio de I/O del procesador. Mediante las líneas de address seleccionamos la operación a realizar
- ➔ Bus Auxiliar de I/O: el Port A oficia de bus de datos, mientras que el Port B provee 6 líneas de address (PB.2 a PB.7 = A0 a A5), más que suficiente para esta aplicación
- ➔ IOSTROBE: los pines del Port E pueden funcionar como chip select
- ➔ CS activo en alto: el estado de reposo es lógico bajo, y al activarse lo hace en estado lógico alto, por el tiempo que dura la operación en el bus, de igual modo a la línea E (Enable) de los Motorola 68xx (razón por la cual a este tipo de interfaz se la suele llamar "tipo Motorola")

Las conexiones del display quedan como puede apreciarse en la tabla a la derecha:

Rabbit	LCD
PA.0	D0
PA.1	D1
PA.2	D2
PA.3	D3
PA.4	D4
PA.5	D5
PA.6	D6
PA.7	D7
PB.2	I/D
PB.5	R/W
PE.1	E
PB.3	CS1
PB.4	CS2

El Rabbit 3000 es un procesador de 3,3V con entradas 5V-tolerant, por lo que el display deberá poder funcionar a 3,3V o se deberá emplear algún tipo de conversor de nivel para las señales empleadas, particularmente en la dirección *microprocesador* → *display* .

Descripción del funcionamiento

Por cuestiones de timing, vamos a mapear lectura y escritura en diferentes espacios de I/O, en vez de emplear la línea $\overline{IOWR}$ como se haría convencionalmente. Esto se debe a que la línea $\overline{IOWR}$ se habilita después del CS (modo de operación de chip select para el IOSTROBE), mientras que las líneas de address son estables con anterioridad. De este modo, dispondremos de direcciones independientes para acceder a cada uno de los controladores y a su vez para instrucciones y datos, tanto en lectura como en escritura.

Dada la diferencia de velocidad entre el poderoso Rabbit 3000 y el pobre turtle display, nos vimos obligados a reducir la velocidad de operación del procesador. En la mayoría de los módulos alcanza con llamar a la función *clockDoublerOff();* en los módulos más rápidos deberá actuarse sobre los divisores que seleccionan el clock del procesador.

Software

En esta nota de aplicación analizaremos unos simples drivers de bajo nivel en assembler; el software de control de alto nivel viene incluido en el archivo adjunto, y su descripción puede obtenerse estudiando notas de aplicación anteriores. Emplearemos variables globales para las cuatro direcciones posibles para cada controlador (recordemos que estos displays poseen dos controladores, uno para cada mitad del display), en las cuales colocaremos las direcciones correspondientes al controlador que estemos accediendo en ese momento.

Una simple función llamada *LCD_SelSide()* realiza esta tarea. Finalmente, la función *LCD_init()* setea lo necesario para habilitar IOSTROBEs con CS activo en alto.

```
#define PORTA_AUX_IO

int dataaddrd,cmdaddrd,dataaddwr,cmdaddwr;

/* Low level functions */

#asm
;
LCD_Status::
l1:     ld hl,(cmdaddrd)                    ; I, R (lectura de instrucción)
        ioe ld a,(HL)
        rla                                 ; Check busy
        jr c,l1                             ; loop while busy
        ret

; resultado en (HL)
;
LCD_ReadData::
        call LCD_Status                     ; check busy
        ld hl,(dataaddrd)                   ; D, R (lectura de dato)
        ioe ld l,(HL)
        ld h,0
        ret

;@sp+2= dato a escribir
;
LCD_WriteData::
        call LCD_Status                     ; check busy
        ld hl,(sp+2)                        ; get value (LSB)
        ld a,l
        ld hl,(dataaddwr)                   ; D, WR (escritura de dato)
        ioe ld (HL),a
        ret

;@sp+2= comando a escribir
;
LCD_WriteCmd::
        call LCD_Status                     ; Check busy
        ld hl,(sp+2)                        ; get value (LSB)
        ld a,l
        ld hl,(cmdaddwr)                    ; I, WR (escritura de instrucción)
        ioe ld (HL),a
        ret

#endasm


/* HD61202 support routines */

void LCD_SelSide(int side)
{
        if(side)       {
                dataaddwr=0x2005;
                dataaddrd=0x200D;
                cmdaddrd=0x200C;
                cmdaddwr=0x2004;
        }
        else    {
                dataaddwr=0x2003;
                dataaddrd=0x200B;
```

```
                cmdaddwr=0x2002;
                cmdaddrd=0x200A;
        }
}


void LCD_init ()
{
// Use Port E bit 1 for active high I/O strobe with 15 wait-states,
#define LCD_STROBE            0x02
#define LCD_CSREGISTER        IB1CR
#define LCD_CSSHADOW          IB1CRShadow
#define LCD_CSCONFIG          0x3C

        // Initialize Port E bit to be a normal I/O pin
        WrPortI(PEFR, &PEFRShadow, (PEFRShadow|LCD_STROBE));

        // Initialize Port E bit to be an output pin
        WrPortI(PEDDR, &PEDDRShadow, (PEDDRShadow|LCD_STROBE));

        // Initialize Port E bit to be a chip select.
        WrPortI(LCD_CSREGISTER, &LCD_CSSHADOW, LCD_CSCONFIG);

        // Set Port E bit to be clocked by PCLK/2
        WrPortI(PECR, &PECRShadow, (PECRShadow & ~0xFF));


        LCD_SelSide(1);
        LCD_WriteCmd ( '\B00111111' );                      // Display on
        LCD_SelSide(0);
        LCD_WriteCmd ( '\B00111111' );                      // Display on
}
```