Application Note AN2147

Interfacing to a Graphics LCD from PSoC

Author: Pham Minh Tri
Associated Projects: Yes
Associated Part Family: CY8C27xxx
PSoC Designer Version: 4.0
Associated Application Notes:

Summary

This Application Note describes how to control a graphic LCD in a PSoC application.

Introduction

A graphic LCD can be an inexpensive, easy-to-control, powerful solution for the display of information to a user. It can provide both text and information from an application. This Application Note shows how to control a graphic LCD using a PSoC device.

Specifications include:

- 128x64 KS0108-controlled graphic LCD
- 10 kΩ center-tapped resistor
- 1 μF capacitors

Graphic LCD

This application uses a DMC12864 128x64 graphic LCD with two built-in KS0108 LCD drivers (one for the left half of the display and one for the right). Thus, the information presented here will cover LCDs with KS0108 and compatible drivers (e.g., Hitachi HD61202).

The KS0108 drivers are easy to implement and most commonly used for dot-matrix graphic LCDs. They behave almost the same as a DDRAM. It is assumed that the reader knows how the interface works since it is beyond the scope of this Application Note. This means that basic things like reading from/writing to DDRAM, checking the busy flag, etc., will not be covered here.

One can think of a 128x64 LCD as having 1024 bytes of memory, every bit of which is visible. The display is split logically in half. It contains two drivers; one controls the left half of the display, the other control the right half. The former is selected by chip-select signal CS1, the latter by CS2. Each driver must be addressed independently. Each half consists of 8 horizontal pages which are 8 bits (1 byte) high. The page addresses, 0-7, specify one of the 8 pages. This is illustrated in Figure 1.

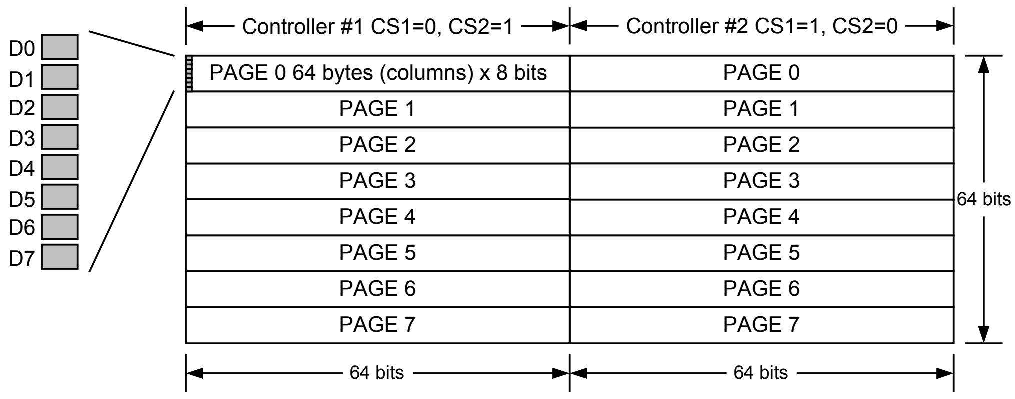
Controller #1 CS1=0, CS2=1
Controller #2 CS1=1, CS2=0
D0
D1
D2
D3
D4
D5
D6
D7
PAGE 0 64 bytes (columns) x 8 bits
PAGE 0
PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
64 bits
64 bits
64 bits

Figure 1. Page Address of a Graphic LCD

CS1 and CS2 are active-low in most LCDs. However, in some rare cases they are active-high. The designer should check the LCD's data sheet in cases of uncertainty. This Application Note assumes chip-selected signals are active-low.

KS0108 Commands

Table 1 shows the KS0108 commands taken from the KS0108's data sheet. These drivers do not have text capability and the commands are few and simple.

Y Address (0-63)
The Y address counter designates the address of the internal DDRAM. An address is set by the instruction and automatically increased by 1 by read or write operations of display data. Y address 0 is the left-most byte, and Y address 63 is the right-most byte of a page.

X Address (0-7)
This is the page address and has no count function.

Display Start Line (0-63)
The display start line register specifies the line in RAM that corresponds to the top line of the LCD panel when displaying contents in display data RAM on the LCD panel. It is used for *scrolling* the screen.

Table 1. KS0108 Commands Taken from the KS0181 Data Sheet

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display ON/OFF	L	L	L	L	H	H	H	H	H	L/H	Controls the display on or off. Internal status and display RAM data is L: Off, H: ON.
Set Address	L	L	L	H	Y Address (0-63)						Sets the Y address in the Y address counter.
Set Page (X Address)	L	L	H	L	H	H	H	Page (0-7)			Sets the Y address at the X address register.
Display Start Line	L	L	H	H	Display Start Line (0-7)						Indicates the display data RAM displayed at the top of the screen.
Status Read	L	H	BUSY	L	ON/OFF	RESET	L	L	L	L	Read Status. Busy: L: Ready H: In Operation ON/OFF: L: Display ON H: Display OFF RESET: L: Normal H: Reset
Write Display Data	H	L	Write Data								Writes data (DB0-7) into display data RAM. After a writing instruction, Y address is automatically increased by 1.
Read Display Data	H	H	Read Data								Reads data (DB0-7) from display data RAM to the data bus.

Pin Assignment

The LCD consists of 20 pins, the functions of which are described in Table 2. The power supply for the graphic LCD comes from V_{SS} and V_{DD}. V_{ee} is the LCD's negative output voltage (-10V). It is used in combination with V_{DD} to produce contrast-adjust voltage. A block diagram for the LCD's connection for contrast adjustment is shown in Figure 2.

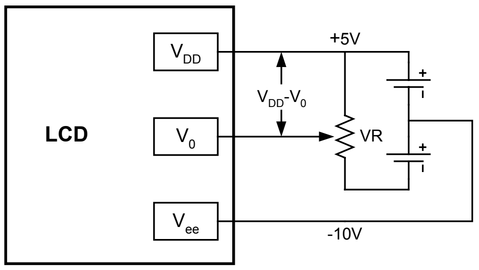

Figure 2. LCD Connection for Contrast Adjustment

Table 2. Pin Assignment

Pin	Symbol	Function
1	V_{SS}	Power Supply (GND)
2	V_{DD}	Power Supply (+5V)
3	V_0	Contrast Adjust
4	RS	Instruction/Data Register Select
5	R/W	Data Read/Write
7-14	DB0-DB7	Data Bus Line
15	CS1	Selection Signal for Chip1
16	CS2	Selection Signal for Chip2
17	RST	Reset
18	V_{ee}	Negative Output (-10V)
19-20	A, K	Power Supply for LED Backlight

The LCD can be reset by holding RST low for at least 100 ns. When it is reset, the display is off and the display start line register becomes 0. The content is not affected. While RST is low, no further command is executed.

The LCD's backlight is controlled by pins 19(A) and 20(K).

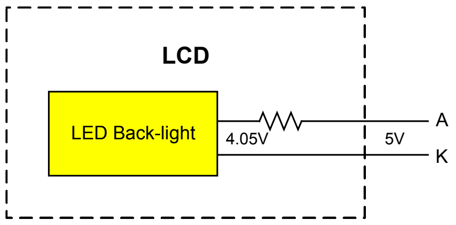

Figure 3. Connection for Backlight

The remaining signals of the LCD (i.e., RS, E, R/W, CS1, CS2 and data bus DB0-DB7) are used to control the KS0108. Note that DB0-DB7 are input/output pins. Their respective PSoC pins should be in High-Z mode before a read operation. Similarly, before a write operation, their respective PSoC pins should be in Strong mode.

PSoC Implementation

The implementation is written in 'C'. The CPU speed is configured at 3 MHz. The software-controlling interface for the graphic LCD is implemented in 2 levels. Level 1 provides basic functions for reading from and writing to the LCD drivers. Level 2 provides routines for execution of KS0108 commands listed in Table 1.

The descriptions of the routines implemented in Level 1 are listed in Table 3.

Table 3. Level 1 Routines

Routine	Description
GLCD_reset	Reset the LCD
GLCD_read_status	Read current status of a driver
GLCD_write_ins	Write an instruction to a driver
GLCD_read_data	Read a data byte from a driver
GLCD_write_data	Write a data byte to a driver

The descriptions of the routines implemented in Level 2 are listed in Table 4.

Table 4. Level 2 Routines

Routine	Description
GLCD_IsBusy	Check if LCD is busy
GLCD_IsDisplay	Check if LCD display is on
GLCD_Display	Turn ON/OFF LCD display
GLCD_SetYAddress	Set Y address of a driver
GLCD_SetXPage	Set X page of a driver
GLCD_StartLine	Set LCD display's start line

Circuit Schematic

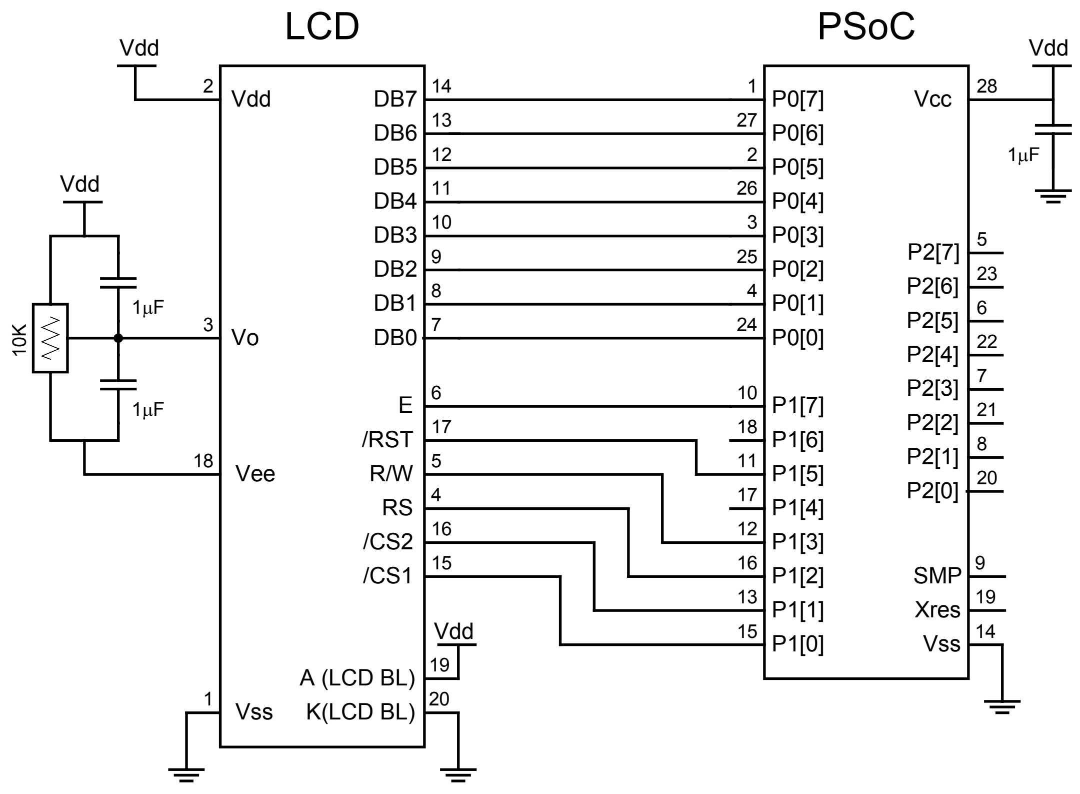

Figure 4. Schematic

main.c Source

```
#include <m8c.h>
#include "logo.h"

//-----------------------------------------------------------------------------
// DMC12864 level-1 routines
//-----------------------------------------------------------------------------

//-------------------------------------------
// Reset the LCD
//-------------------------------------------
void GLCD_reset()
{
                                                BYTE i;
                                 PRT1DR = 0x00; // pull down reset
                               for( i = 0; i < 100; i++ ); // delay
                                  PRT1DR = 0x23; // pull up reset
}


//-------------------------------------------
// Read the current status of a driver
// drv=
//    0: left driver
//    1: right driver
//-------------------------------------------
BYTE GLCD_read_status( BYTE drv )
{
                                                BYTE x;
                                      PRT1DR = drv? 0x29: 0x2A;
```

```
                              PRT0DM0 = 0x00; // turn off port 0's output
                                            PRT0DM1 = 0xFF;
                                      PRT1DR |= 0x80; // turn E on
                                       x = PRT0DR; // read status
                                     PRT1DR &= 0x7F; // turn E off
                                               return x;
}


//-------------------------------------------
// Write an instruction to a driver
// drv=
//    0: left driver
//    1: right driver
//-------------------------------------------
void GLCD_write_ins( BYTE drv, BYTE ins )
{
                                       PRT1DR = drv? 0x21: 0x22;
                                  PRT0DR = ins; // write instruction
                       PRT0DM0 = 0xFF; // turn on port 0's output - strong mode
                                            PRT0DM1 = 0x00;
                                      PRT1DR |= 0x80; // turn E on
                                     PRT1DR &= 0x7F; // turn E off
}


//-------------------------------------------
// Write data to a driver
// drv=
//    0: left driver
//    1: right driver
//-------------------------------------------
void GLCD_write_data( BYTE drv, BYTE data )
{
                                       PRT1DR = drv? 0x25: 0x26;
                                      PRT0DR = data; // write data
                       PRT0DM0 = 0xFF; // turn on port 0's output - strong mode
                                            PRT0DM1 = 0x00;
                                      PRT1DR |= 0x80; // turn E on
                                     PRT1DR &= 0x7F; // turn E off
}


//-------------------------------------------
// Read data from a driver
// drv=
//    0: left driver
//    1: right driver
//-------------------------------------------
BYTE GLCD_read_data( BYTE drv )
{
                                                BYTE x;
                                       PRT1DR = drv? 0x2D: 0x2E;
                              PRT0DM0 = 0x00; // turn off port 0's output
                                            PRT0DM1 = 0xFF;
                                      PRT1DR |= 0x80; // turn E on
                                        x = PRT0DR; // read data
                                     PRT1DR &= 0x7F; // turn E off
                                               return x;
}


//----------------------------------------------------------------------------
// DMC12864 level-2 routines
//----------------------------------------------------------------------------

//-------------------------------------------
// Check if the LCD display is on
// assumming driver are both on or both off
//-------------------------------------------
BOOL GLCD_IsDisplay()
{
                                return (GLCD_read_status(0) >> 5) & 1;
}


//-------------------------------------------
// Check if a driver is busy
//-------------------------------------------
BOOL GLCD_IsBusy( BYTE drv )
{
```

```
                               return ((char)GLCD_read_status(drv)) < 0;
}


//-------------------------------------------
// Turn ON/OFF the LCD display (both drivers)
//-------------------------------------------
void GLCD_Display( BOOL on )
{
                                       BYTE x = 0x3E | (on & 1);
                                        GLCD_write_ins( 0, x );
                                        GLCD_write_ins( 1, x );
}


//-------------------------------------------
// Set Y address of a driver
//-------------------------------------------
void GLCD_SetYAddress( BYTE drv, BYTE addr )
{
                              GLCD_write_ins( drv, 0x40 | (addr & 63) );
}


//-------------------------------------------
// Set X page of a driver
//-------------------------------------------
void GLCD_SetXPage( BYTE drv, BYTE page )
{
                               GLCD_write_ins( drv, 0xB8 | (page & 7) );
}


//-------------------------------------------
// Set start line of both drivers
// Use this for scrolling the LCD
//-------------------------------------------
void GLCD_StartLine( BYTE line )
{
                                     BYTE x = 0xC0 | (line & 63);
                                        GLCD_write_ins( 0, x );
                                        GLCD_write_ins( 1, x );
}



//----------------------------------------------------------------------------
// Main module
//----------------------------------------------------------------------------
void main()
{
    // Insert your main routine code here.
    BYTE i, j, line;
    BYTE x;

    GLCD_reset();

    GLCD_Display(1);

    // write down Cypress logo to fill up the screen
                                        GLCD_SetYAddress(0,0);
                                        GLCD_SetYAddress(1,0);
    for( i = 0; i < 8; i++ )
      for( j = 0; j < 64; j++ )
      {
                                          GLCD_SetXPage(0,i);
                               GLCD_write_data(0,i<4?logo[j+i*64]:0x00);

                                          GLCD_SetXPage(1,i);
                             GLCD_write_data(1,i>3?logo[j+(i-4)*64]:0x00);
      }

    // scroll it
    while(1)
    {
                                                line++;
                        for( i = 0; i < 100; i++ ) for( j = 0; j < 100; j++ );
                                         GLCD_StartLine(line);
    }
}
```

Graphic LCD Screenshot and Set Up

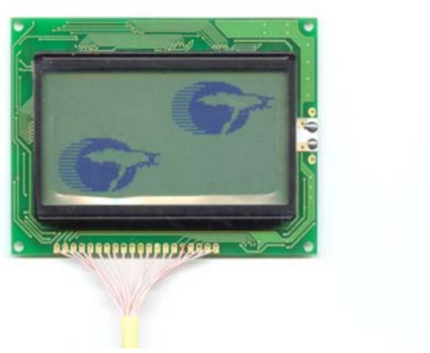

Figure 5. LCD Screenshot

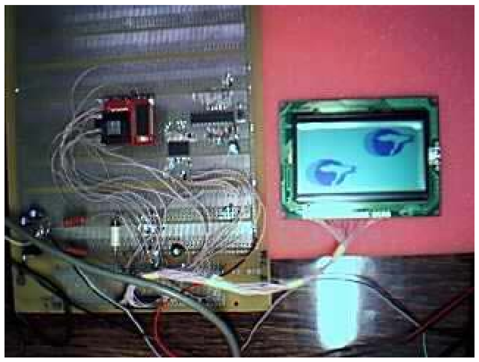

Figure 6. LCD Set Up

About the Author

Name:	Pham Minh Tri
Background:	Earned B. Eng (Computer Engineering) with Honors from Nanyang Technological University, Singapore in 2003. A member of CMS worldwide consultant team.
Contact:	pmtri80@yahoo.com