User Interface - Graphics LCD and PSoC Interface

AN2152

Author: Svyatoslav Paliy
Associated Project: Yes
Associated Part Family: CY8C27443
GET FREE SAMPLES HERE
Software Version: PSoC Designer™ 4.0
Associated Application Notes: None

Application Note Abstract

This Application Note describes how to control a PCD8544-based graphics LCD in a PSoC® project.

Introduction

In most applications there is a need to display information to the user. A graphics LCD is a powerful, easy-to-control solution. It can provide both text and graphical illustration to an application. This Application Note shows how to control a graphics LCD using a PSoC device. The project has a software library for write text and a graphics drawing on LCD and PC software to build a font generator and bitmap-to-C-array converter.

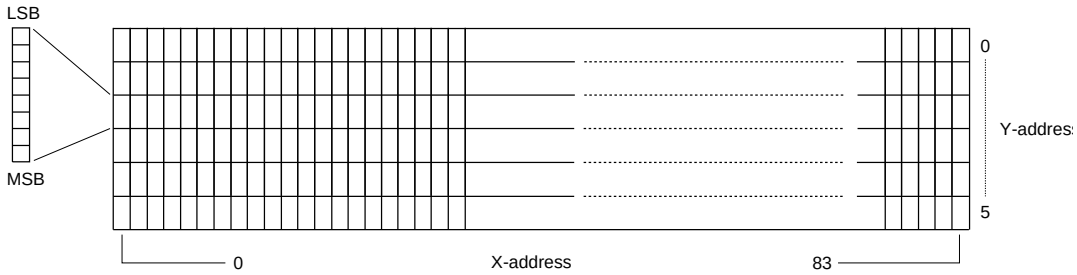
Graphics LCD

In this application, I used a 48x84 graphics LCD with Philips PCD8544 controller/driver. You may find the data sheet for this controller under Reference [1]. Devices similar to the PCD8544 are listed at Reference [2].

Many manufacturers make displays based on the PCD8544 or compatible controllers. This very low-cost LCD controller has memory bits, each of which represent one pixel on the LCD. This memory allows only writes. It is not possible to read from this memory, which can create some difficulties with building routines for the smaller memory versions of PSoC.

Data are downloaded in bytes into the 48x84-bit RAM data display matrix of PCD8544, as indicated in Figure 1. The columns are indicated by the address pointer. The address ranges are: X 0 to 83 (1010011), Y 0 to 5 (101). Addresses outside these ranges are not allowed. The X addresses increment after each byte. After the last X address (X = 83), X wraps around to 0 and Y increments to the address in the next row. After the very last address (X = 83 and Y = 5), the address pointers wrap around to address (X =0 and Y =0).

Figure 1. LCD RAM Format, Addressing



Circuit Schematic

The LCD connects to the PSoC by four wires. Two wires are for one-direction SPI, one wire is for data/control switching, and one wire is for the reset signal. The PCD8544 needs one external capacitor for an internal bias voltage generator. Figure 2 shows the schematic when PSoC is powered by 3.3V. Figure 3 shows the schematic when PSoC is powered by 5V. Some applications only allow use of 3.3V supply. Some require 5V and therefore need an additional level translator to be added.

Figure 2. Circuit Schematic for 3.3V Powered PSoC

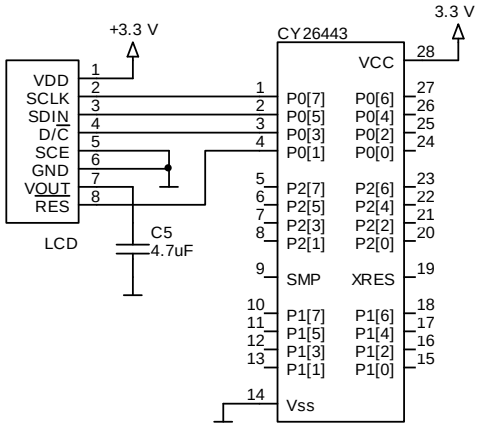
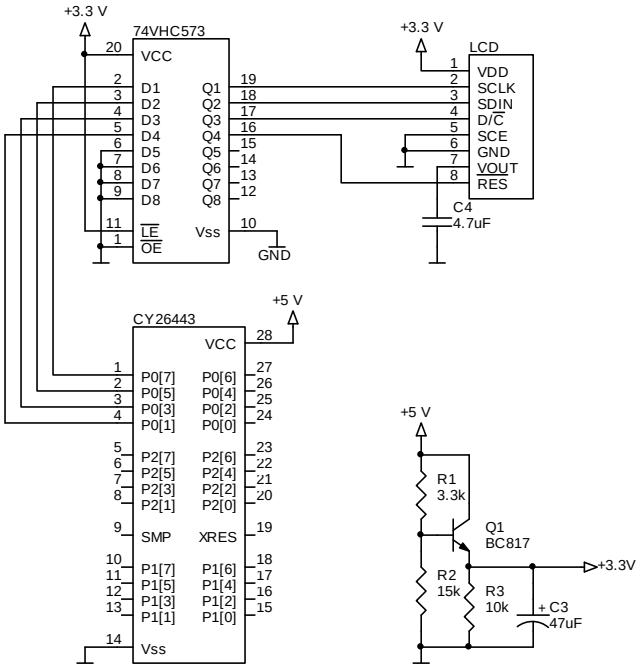


Figure 3. Circuit Schematic for 5V Powered PSoC



Software Library

For the simplified drawing on this LCD, I wrote a software library that can work in two modes: drawing over background (when C-compiler directive, DRAW_OVER_BACKGROUND, is defined) and drawing without background (in other cases).

Because the LCD internal memory is write only and some PSoC devices have too little memory to build a cache, all drawing routines output immediately on the LCD.

The software library has two low-level functions that are hardware dependent (see Table1). If you want to port this library onto a similar LCD controller with another physical connection (for example, use of BF9864AFPH with I^2C interface), you must rewrite only these two functions.

Table 1. LCD Controller Low-Level, Hardware-Dependent Functions

LcdSendData(char data)	Send byte of data to LCD. For more information, see the LCD driver data sheet.
LcdSendCommand(char data)	Send command byte to LCD. For more information, see the LCD driver data sheet.

High-level functions that may be used with DRAW_OVER_BACKGROUND are listed in Table 2. They differ from the functions that may be used without DRAW_OVER_BACKGROUND (see Table 3) by **dt** parameter, which can take the following values:

- DRAW_OR – Text or graphics draw over background with using Logical OR operator under drawn and background pixels.
- DRAW_XOR – Similar to DRAW_OR but uses XOR instead OR operator.
- DRAW_CLEAR – Does not draw pixels, only restores background. Erases drawn pixels.

Table 2. High-Level Functions used when DRAW_OVER_BACKGROUND is Defined

LcdInit(const char * dataPtr)	Performs LCD initialization, draws background. Parameters: dataPtr – pointer to array in Flash memory that contains background.
LcdSetBackground (const char * dataPtr)	Allows pointer to change to current background. Does not perform repaint. Only the pointer changes. Parameters: dataPtr – pointer to array that contains background.
LcdClear()	Clears display and only shows background.
LcdContrast(char contrast)	Allows contrast change. No visible result at ambient temperature. High temperature allows decrease contrast. Low temperature allows increase contrast. Parameters: contrast – byte describes contrast (higher value means higher contrast).
LcdGoTo(char x, char y)	Changes current text position. Parameters: x – X- coordinant of text position. y – Y- coordinant of text position. Y- coordinant means not quite a pixel, but an 8-pixel bank (e.g., display has 6 bank by height).
LcdImage (char x, char y, char xsize, char ysize, const char * dataPtr)	Draws image. Parameters: x,y – coordinants of image top-left corner. xsize, ysize – image width and height. dataPrt – pointer to the array that contains image.
LcdChr (char ch, draw_type dt)	Draws single character (by the small font) starting from current text position (see `LcdGoto` function above). Parameters: ch – character. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).

LcdStr (char *dataPtr, draw_type dt)	Writes string (by the small font) starting from current text position from data memory. Parameters: dataPtr – pointer to the string in the data memory. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
LcdCStr (const char *dataPtr, draw_type dt)	Writes string (by the small font) starting from current text position from program memory. Parameters: dataPtr – pointer to string in the program memory. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
LcdBigChr (char x, char y, char ch, draw_type dt)	Draws single character by the big font. Parameters: x,y – coordinants of character. ch – character. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
LcdBigStr (char x, char y, char *dataPtr, draw_type dt)	Draws string from data memory by the big font. Parameters: x,y – coordinants of string begin. dataPtr – pointer to the string in data memory. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
LcdBigCStr (char x, char y, const char *dataPtr, draw_type dt)	Draws string from program memory by the big font. Parameters: x,y – coordinants of string begin. dataPtr – pointer to the string in program memory. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
LcdVBargraph (char x, char ystart, char yend, char yposition, draw_type dt)	Draws vertical bar graph. Parameters: x – coordinate of left bar graph. ystart – coordinant of top bar graph (8-pixel bank). yend – coordinant of bottom bar graph (8-pixel bank). yposition – current bar graph position, by pixels. (yposition <=(yend-begin)*8). dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
LcdHBargraph (char y, char xstart, char xend, char xposition, draw_type dt)	Draws horizontal bar graph. Parameters: y – coordinant of the top bar graph (8-pixel bank). xstart – coordinant of the left bar graph. xend – coordinant of the right bar graph. xposition – current bar graph position, by pixels. (xposition <=xyend-xbegin). dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
void LcdLine (char xb, char yb, char xe, char ye, draw_type dt);	Draws line. Parameters: xb,yb – coordinants of where the line begins. xe,ye – coordinants of where the line ends. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).

Table 3. High-Level Functions used when DRAW_OVER_BACKGROUND is Undefined

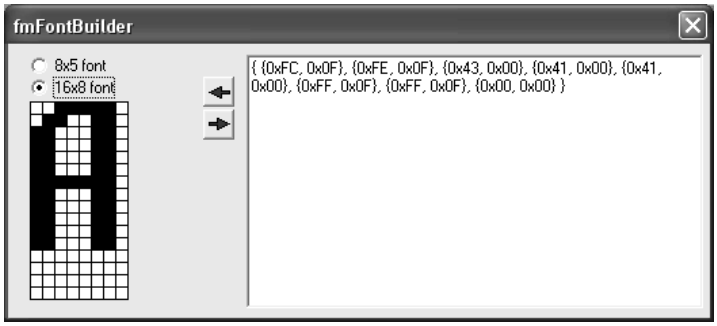
LcdInit()	Performs LCD initialization.
LcdClear()	Clears display and shows blank.
LcdContrast(char contrast)	Allows contrast change. No visible result is observed at ambient temperature. Parameters: contrast – byte describes contrast (higher value means higher contrast).
LcdGoTo(char x, char y)	Change current text position. Parameters: x – X-coordinant of text position. y – Y-coordinant of text position. Y- coordinant means not quite a pixel, but an 8-pixel bank (e.g., display has 6 bank by height).
LcdImage (char x, char y, char xsize, char ysize, const char * dataPtr)	Draws image. Parameters: x,y – coordinants of image top-left corner. xsize, ysize – image width and height. dataPrt – pointer to the array that contains image.
LcdChr (char ch)	Draws single character (by the small font) starting from current text position (see `LcdGoto` function above). Parameters: ch – character.
LcdStr (char *dataPtr)	Writes string (by the small font) starting from current text position. Parameters: dataPtr – pointer to the string in the data memory.
LcdCStr (const char *dataPtr)	Writes string (by the small font) starting from current text position. Parameters: dataPtr – pointer to the string in the program memory.
LcdBigChr (char x, char y, char ch)	Draws single character by the big font. Parameters: x,y – coordinants of character. ch – character.
LcdBigStr (char x, char y, char *dataPtr)	Draws string from data memory by the big font. Parameters: x,y – coordinants where string begins. dataPtr – pointer to the string in data memory.
LcdBigCStr (char x, char y, const char *dataPtr)	Draw string from program memory by the big font. Parameters: x,y – coordinants where string begins. dataPtr – pointer to the string in program memory.
LcdVBargraph (char x, char ystart, char yend, char yposition)	Draws vertical bar graph. Parameters: x – coordinant of left bar graph. ystart – coordinant of top bar graph (8-pixel bank). yend – coordinant of bottom bar graph (8-pixel bank). yposition – coordinate of current bar graph position, by pixel. (yposition <=(yend-ybegin)*8).
LcdHBargraph (char y, char xstart, char xend, char xposition)	Draws horizontal bar graph. Parameters: y – coordinant of the top bar graph (8-pixel bank). xstart – coordinant of the left bar graph. xend – coordinant of the right bar graph. xposition – current bar graph position, by pixels. (xposition <=xyend-xstart).
void LcdLine (char xb, char yb, char xe, char ye);	Draws line. Parameters: xb,yb – coordinants of where the line begins. xe,ye – coordinants of where the line ends.

PC Utilities

The software library contains two fonts. Both big and small fonts have been written as separate header files (*big_font.h* and *small_font.h*). To simplify font building, I wrote a utility for the PC that facilitates the font building process (see Figure 4).

In the left side of the form, you may draw a character and give its hexadecimal representation in the text editor. You may also write hexadecimal code and get a character picture.

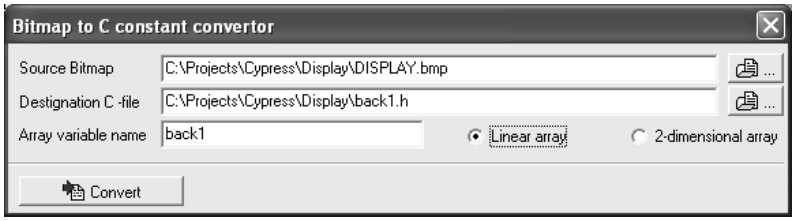
Figure 4. Font Building Utility



Another utility (Figure 5) converts bitmaps to C-language header files. Users must choose the path to the bitmap. Only black-and-white bitmaps with a height divisible by eight are supported (which is a consequence of using LCD controller page organization).

Also, users must choose a target file. If a target file exists, the utility rewrites it. The name for the hexadecimal array will be built from the file name but can be changed. By pressing the Convert button, the bitmap converts to a constant array of hexadecimal values. A file with conversion results is also generated.

Figure 5. Utility for Bitmap-to-C-Array Conversion

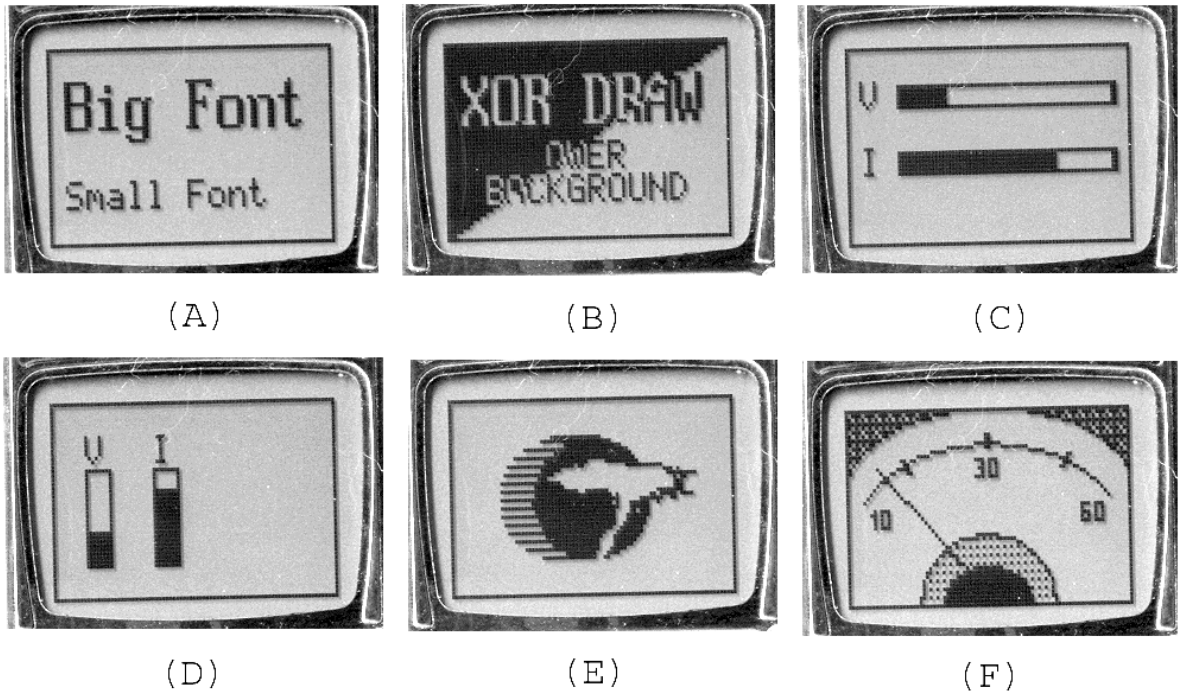


Demonstration Applications

The demonstration application consists of a few screens that show display output possibilities. The first demonstration screen shows big and small text writing on the LCD. The second screen shows a text drawing using the DRAW_XOR parameter.

The third and fourth screens show horizontal and vertical drawings of bar graphs, respectively. The fifth screen shows a bitmap drawing. And the sixth screen is an example of an analog gauge showing line drawings with DRAW_OR and DRAW_CLEAR parameters.

Figure 6. Screenshots from Demo Application



References

1. Datasheet for PCD8544 can be downloaded from: www.semiconductors.philips.com/acrobat/datasheets/PCD8544_1.pdf
2. List similar to PCD8544 devices can be found at: www.semiconductors.philips.com/similar/PCD8544U_2_F1.html

About the Author

Name: Svyatoslav Paliy
Title: Application Engineer
Background: Svyatoslav earned his Master of Science diploma in 2000 from National University "Lviv Polytechnic" (Ukraine). His interests include programming for embedded systems and Windows and Linux.

Contact: You may reach him at svt@isto.lviv.ua.

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/