Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 001-34580

Spec Title: GRAPHICS LCD AND PSOC(R) INTERFACE - AN2152

Sunset Owner: Ramnath R.K (RKRM)

Replaced by: 001-97630

AN2152

Graphics LCD and PSoC® Interface

Author: Svyatoslav Paliy
Associated Project: Yes
Associated Part Family: CY8C27443
Software Version: PSoC Designer™ 5.1
Related Application Notes: None

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN2152.

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

Contents

Introduction 1
Graphics LCD 1
Circuit Schematic 2
Software Library 3
PC Utilities 6
Demonstration Applications 7
Summary 9
References 9
Worldwide Sales and Design Support 11

Introduction

With 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 illustrations for an application. This application note shows how to control a graphics LCD using a PSoC device. The project included with this application note consists of the following:

- A software library for text and graphics on an LCD
- PC software to build a font generator and bitmap-to-C array converter

Graphics LCD

This application uses a 48×84 graphics LCD with a Philips PCD8544 controller/driver. To obtain the data sheet for this controller, see:

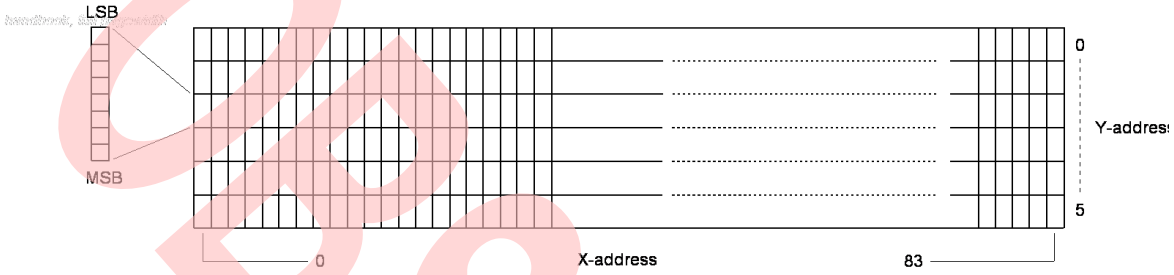
http://www.nxp.com/acrobat_download2/datasheets/PCD8544_1.pdf.

Many manufacturers make displays based on the PCD8544 or compatible controllers. This very low-cost LCD controller has memory bits, each of which represents 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.

OBSOLETE

Data is downloaded in bytes into the 48×84–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).

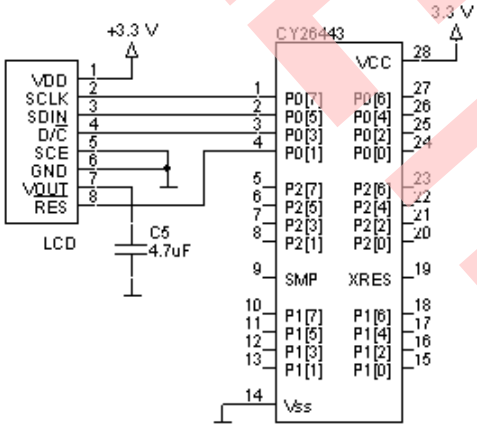
Figure 1. LCD RAM Format, Addressing



Circuit Schematic

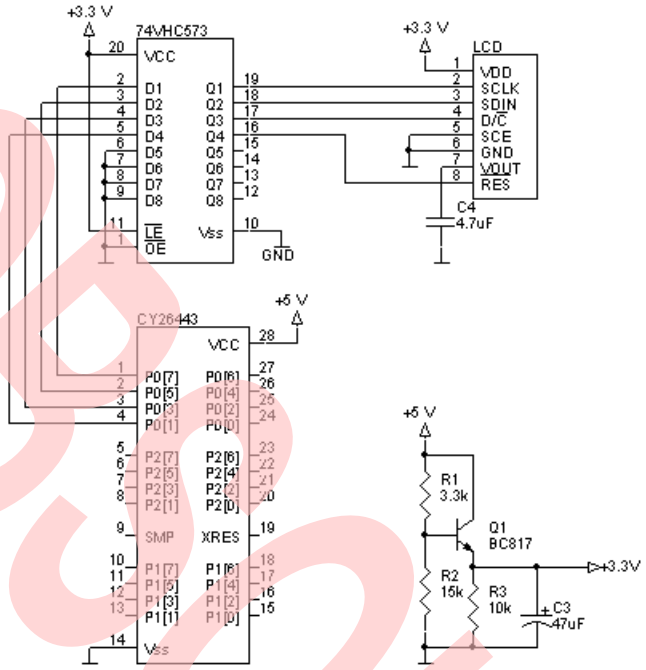
The LCD connects to the PSoC by four wires. Two wires are for one-direction SPI, one wire is for data/control switching, and one wire is for the reset signal. The PCD8544 needs one external capacitor for an internal bias voltage generator. Figure 2 shows the schematic when PSoC is powered by 3.3 V. Figure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

Figure 2. Circuit Schematic for 3.3 V–Powered PSoC



OBSOLETE

Figure 3. Circuit Schematic for 5 V–Powered PSoC



Software Library

The project included with this application note includes a software library that can work in two modes: drawing over background (when the 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 Table 1). 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 the **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.

OBSOLETE

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- coordinate of text position. y – Y- coordinate of text position. Y- coordinate means not quite a pixel, but an 8-pixel bank (for example, display has 6 bank by height).
LcdImage (char x, char y, char xsize, char ysize, const char * dataPtr)	Draws image. Parameters: x,y – coordinates 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 shown previously). 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 – coordinates 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 – coordinates of string begin. dataPtr – pointer to the string in data memory. dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).

OBSOLETE

LcdBigCStr (char x, char y, const char *dataPtr, draw_type dt)	Draws string from program memory by the big font. Parameters: x,y – coordinates 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 – coordinate of top bar graph (8-pixel bank). yend – coordinate 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 – coordinate of the top bar graph (8-pixel bank). xstart – coordinate of the left bar graph. xend – coordinate 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 – coordinates of where the line begins. xe,ye – coordinates 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-coordinate of text position. y – Y-coordinate of text position. Y- coordinate means not quite a pixel, but an 8-pixel bank (for example, display has 6 bank by height).
LcdImage (char x, char y, char xsize, char ysize, const char * dataPtr)	Draws image. Parameters: x,y – coordinates 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.

OBSOLETE

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 – coordinates of character. ch – character.
LcdBigStr (char x, char y, char *dataPtr)	Draws string from data memory by the big font. Parameters: x,y – coordinates 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 – coordinates 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 – coordinate of left bar graph. ystart – coordinate of top bar graph (8-pixel bank). yend – coordinate 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 – coordinate of the top bar graph (8-pixel bank). xstart – coordinate of the left bar graph. xend – coordinate 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 – coordinates of where the line begins. xe,ye – coordinates of where the line ends.

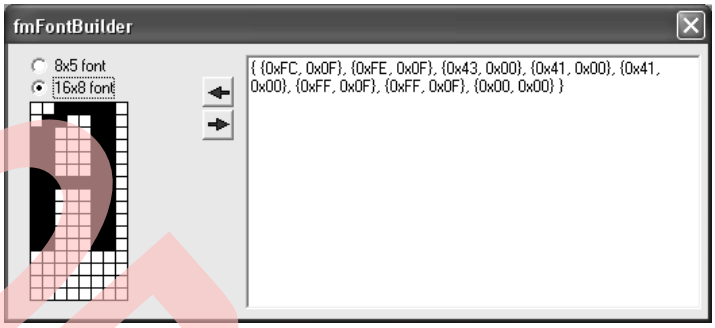
PC Utilities

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, a utility for the PC is included that facilitates the font building process (see Figure 4).

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

OBSOLETE

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



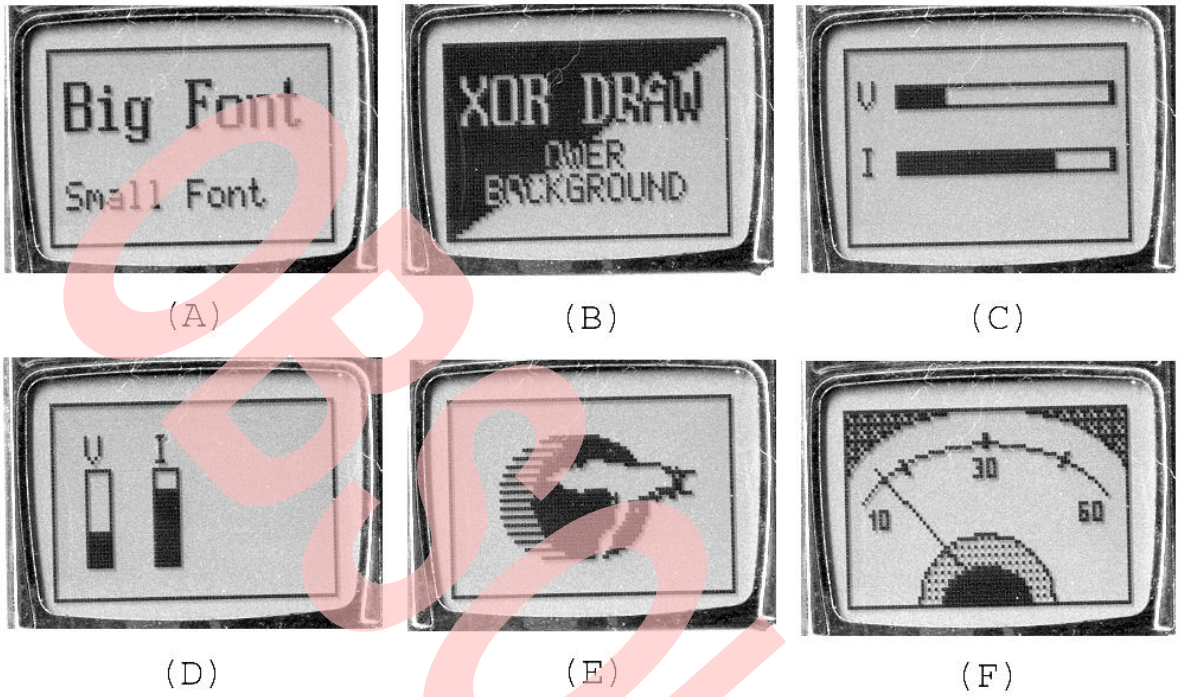
Demonstration Applications

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.

OBSOLETE

Figure 6. Screenshots from Demo Application



OBSOLETE

Summary

This application note presented the technique to interface a Graphic LCD with PSoC. It also explains the functions offered by the software library for text and graphics on an LCD.

References

Datasheet for PCD8544 can be downloaded from: http://www.nxp.com/acrobat_download2/datasheets/PCD8544_1.pdf

About the Author

Name:	Svyatoslav Paliy. Multiple tables if multiple authors
Title:	Sys Desgn/Arch Engrg Mgr Sr.
Background:	Svyatoslav earned his Master of Science diploma in 2000 from National University “Lviv Polytechnic” (Ukraine). His interests include programming for embedded systems, Windows, and Linux.

OBSOLETE

Document History

Document Title: Graphics LCD and PSoC® Interface - AN2152

Document Number: 001-34580

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1505943	SVYP	10/08/2007	New application note.
*A	3221119	SVYP	04/09/2011	Updated version of PSoC Designer.
*B	4357864	RKRM	04/23/2014	Updated in new template. Completing Sunset Review.
*C	4749076	RKRM	05/27/2015	Obsolete document. Completing Sunset Review.

OBSOLETE

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

OBSOLETE

PSoC is a registered trademark of Cypress Semiconductor Corp. PSoC Designer is a trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2007-2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.