CE97630 – Interfacing PSoC® 1 with PCD8544 Graphics LCD Module

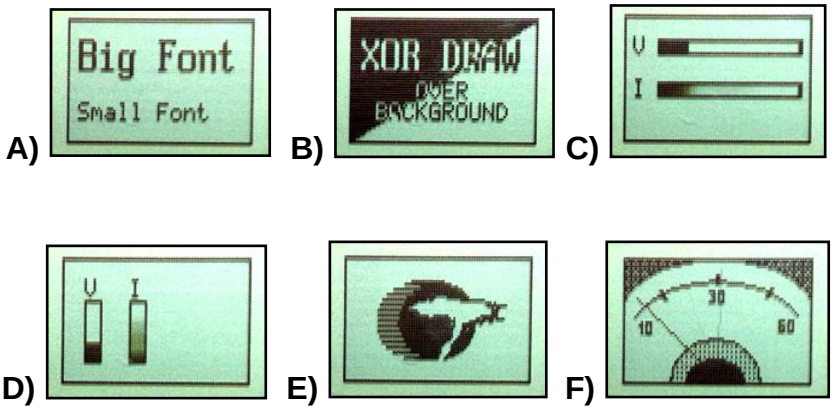
Objective

This example demonstrates how to interface PSoC® 1 with the PCD8544 graphic LCD module.

Overview

The code example provides a software library to interface with the Graphics LCD. The PSoC Designer™ project uses high-level functions in the library to demonstrate an application, which displays the images shown in Figure 1. In addition to that, the code example includes hardware setup details, explaining how to interface a PCD8544 Graphics LCD module with the PSoC 1 device. The Philips PCD8544 controller/driver is a 48×84 graphics LCD, used by several manufacturers to produce low-cost LCD modules. Nokia 5110 and 3310 are two popular LCD modules that use the PCD8544 driver.

Figure 1. LCD Screenshots from Demo Application



Requirements

- **Tool:** PSoC Designer 5.4 SP1
- **Programming Language:** C
- **Associated Parts:** CY8C24x23, CY8C27x43
- **Related Hardware:** CY3210 PSoC Eval1

PSoC Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right PSoC device for your design, and quickly and effectively integrate the device into your design. The following is an abbreviated list for PSoC® 1:

- **Overview:** PSoC Portfolio, PSoC Roadmap
- **Product Selectors:** PSoC® 1, PSoC® 3, PSoC® 4, or PSoC® 5LP. In addition, PSoC Designer offers a device selection tool within PSoC® 1, at the time of creating a new project.
- **Datasheets**: Describe and provide electrical specifications for all the PSoC® 1family of devices. Visit the PSoC® 1 Datasheets webpage for a complete list.
- **Application Notes and Code Examples:**
  - Visit the PSoC®1 Code Examples webpage for a comprehensive list of code examples.
  - Cypress offers a large number of PSoC application notes covering a broad range of topics, from basic to advanced level. Recommended application notes for getting started with PSoC 1 are:
    - AN75320: Getting Started with PSoC® 1
    - AN2094 - PSoC® 1 - Getting Started with GPIO
    - AN2015 - PSoC® 1 - Getting Started with Flash & E2PROM
    - AN2014 - Basics of PSoC® 1 Programming
    - AN32200 - PSoC® 1 - Clocks and Global Resources
    - AN2010 - PSoC® 1 Best Practices and Recommendations
- **Technical Reference Manuals (TRM):**

  Visit the PSoC® 1 TRM page for the complete list of TRMs. Following documents provide detailed descriptions of the Architecture, Programming specification and Register map details of CY8C2XXXX PSoC® 1 device family.
  - PSoC®1 CY8C2XXXX TRM
  - PSoC®1 ISSP Programming Specifications
- **Development Kits:**
  - CY3210 - CY8C24x23 PSoC(R) Evaluation Pods (EvalPod) are 28-pin PDIP adapters that seamlessly connect any PSoC device to the 28-pin PDIP connector on any Cypress PSoC development kit. CY3210-24x23 provides evaluation of the CY8C24x23A PSoC device family on any PSoC developer kit.
  - PSoC developer kits sold separately.

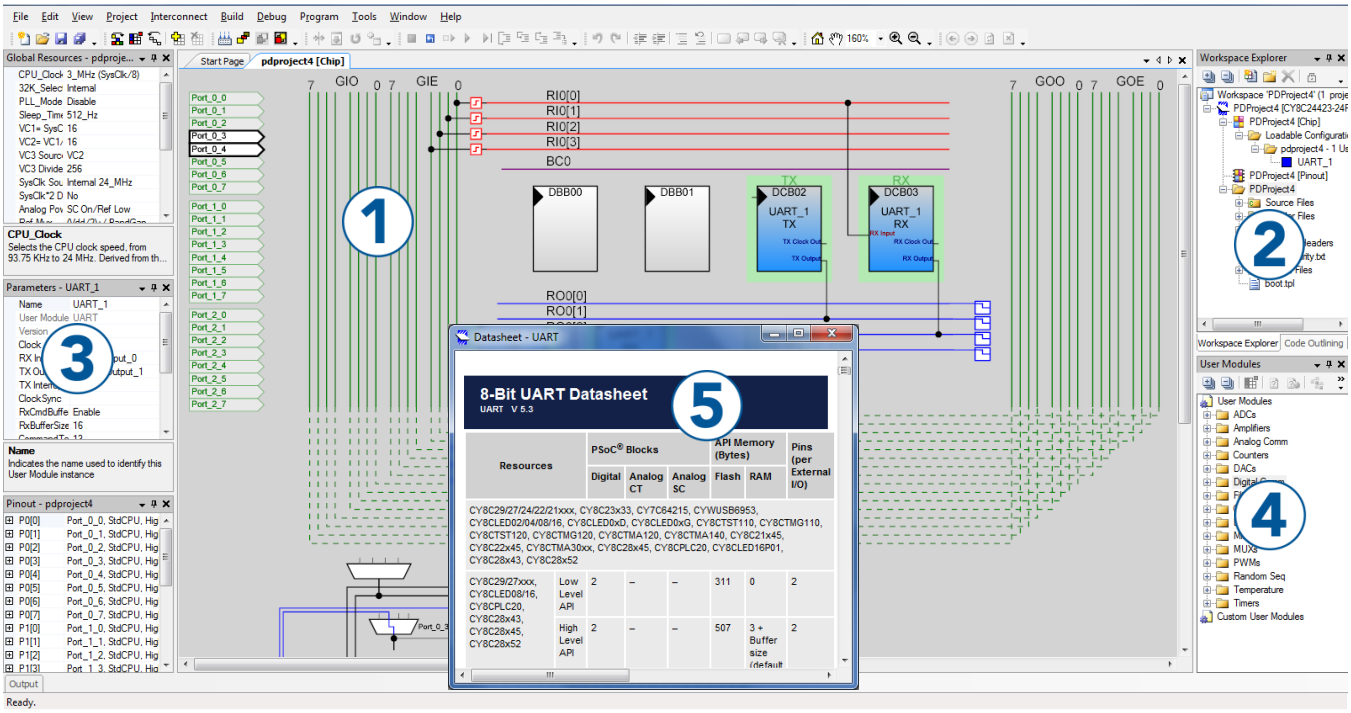
  Visit the PSoC® 1 Kits page and refer the Kit Selector Guide document to find out the suitable development kits and debuggers for all PSoC 1 families.
- The CY3217-MiniProg1 and CY8CKIT-002 PSoC® MiniProg3 device provides an interface for flash programming.
  - Knowledge Base Articles (KBA): Provide design and application tips from experts on the devices/kits. For instance, Flash read/write access from firmware, explains how we can read and write to flash in PSoC 1 devices.

PSoC Designer

PSoC Designer is a free Windows-based Integrated Design Environment (IDE). It enables concurrent hardware and firmware design of systems based on PSoC®1. See Figure 2 – with PSoC Designer, you can:

1. Drag and drop user modules to build your hardware system design in the main design workspace
2. Codesign your application firmware with the PSoC hardware, using the PSoC Designer IDE C compiler
3. Configure user module
4. Explore the library of User modules
5. Review user module datasheets

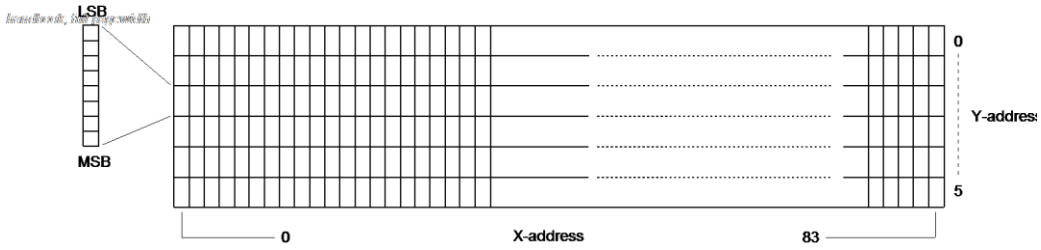
Figure 2. PSoC Designer Features



Graphics LCD

PCD8544 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 difficulties with building routines for the smaller memory versions of PSoC.

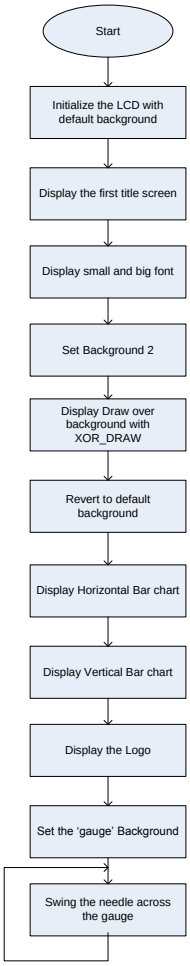
Figure 3. LCD RAM Format, Addressing



Data is downloaded in bytes, into the (6*8) 48×84-bit RAM data display matrix of PCD8544, as indicated in Figure 3. Rows are byte-addressable and columns are bit-addressable. 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 last address (X = 83 and Y = 5), the address pointers wrap around to the addresses X =0 and Y =0.

Design

Figure 4. Flowchart



The flow of the LCD demonstration project is described in Figure 4. The LCD initialization routine adjusts the bias, contrast settings of the LCD, and sets up the default background screen, which is “Back1”. After the LCD is initialized, the screens are displayed in sequence with a couple of seconds delay in between. The final screen, in which a needle swings across the gauge, is displayed constantly.

The hex file of the code is available in the folder **Display** as “*Display.hex*”. To view the flow of the code mentioned above, program the hex file to the specified part number mentioned in the Requirements section.

Software Library and API

The PSoC Designer project, available as part of this code example, includes a software library. The software library has two low-level functions that are hardware-dependent (see Table 1), that is based on the SPI interface. If you want to port this library to a similar LCD controller with a different physical interface (for example, use of the BF9864AFPH with I^2C interface), you must rewrite only these two functions.

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

Function Name	Description
`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.

The library 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). DRAW_OVER_BACKGROUND will enable us to display a background image or text and then superimpose additional text.

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 data parameter, which can take the following values:

- DRAW_OR – Text or graphics are drawn over background using Logical OR operator.
- DRAW_XOR – Similar to DRAW_OR, but uses XOR instead of 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

Function Name	Description
`LcdInit(const char * dataPtr)`	Performs LCD initialization with default background.
	Parameters: dataPtr – pointer to array in flash memory that contains background.
`LcdSetBackground( const char * dataPtr)`	Changes the background on the screen.
	Parameters: dataPtr – pointer to array that contains background.
`LcdClear()`	Clears display retaining the background.
`LcdContrast(char contrast)`	Changes the contrast. Result is not visible at ambient temperature. Higher temperature decreases the contrast. Low temperature increases the contrast.
	Parameters: contrast – byte describes contrast (higher value means higher contrast).
`LcdGoTo(char x, char y)`	Changes the current text position.
	Parameters: x – X- co-ordinate of text position. y – Y- co-ordinate of text position. Y- coordinate indicates the line number of the LCD along the vertical direction
`LcdImage ( char x,`	Draws an image.

Function Name	Description
char y, char xsize, char ysize, const char * dataPtr)	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)	Writes a 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 with a 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, with a 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).
LcdBigCStr (char x, char y, const char *dataPtr, draw_type dt)	Draws string from program memory, with 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, in 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, in pixels. (xposition =(xend-xbegin)). dt – (DRAW_OR, DRAW_XOR or DRAW_CLEAR).
void LcdLine (	Draws line.

Function Name	Description
char xb, char yb, char xe, char ye, draw_type dt);	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

Function Name	Description
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- co-ordinate means not quite a pixel, but an 8-pixel bank (for example, display has 6 banks 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)	Writes a single character (by the small font) starting from current text position (see LcdGoto function above).
	Parameters: ch – character.
LcdStr (char *dataPtr)	Writes string (with 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.

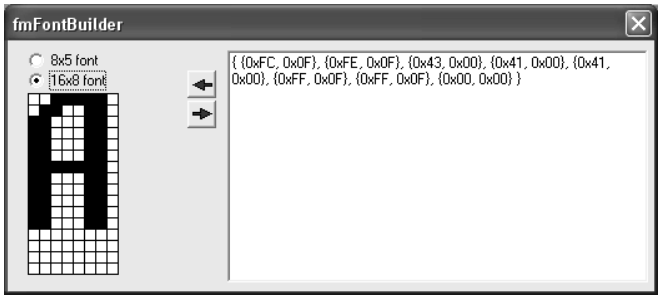
Function Name	Description
`LcdVBargraph ( char x, char ystart, char yend, char yposition)`	Draws vertical bar graph.
	Parameters: x – coordinate of left bar graph. ystart – coordinate of top bar graph (8-pixel bank). yend – coordinate of bottom bar graph (8-pixel bank). yposition – coordinate of current 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 are written as separate header files (*big_font.h* and *small_font.h*). To simplify font building, a PC utility is included, which facilitates the font building process (see Figure 5).

On 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.

Figure 5. Font Building Utility



Another utility (Figure 6) converts the bitmap 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).

In addition, 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 6. Utility for Bitmap-to-C-Array Conversion



User Modules

Table 4 lists the PSoC Designer user modules used in this example, as well as the placement of each.

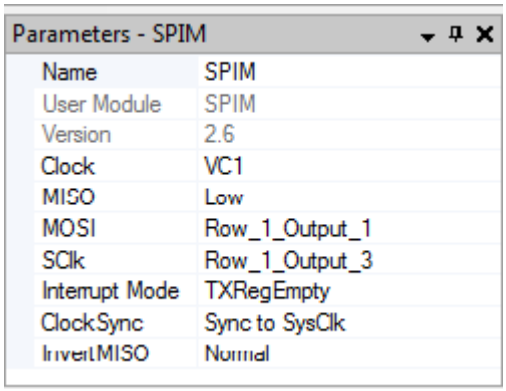
Table 4. List of PSoC Designer User Modules

Component or User Module	Placement
SPIM	DCB 12

User Module Parameters

Figure 7 shows the user module parameter for the SPIM component.

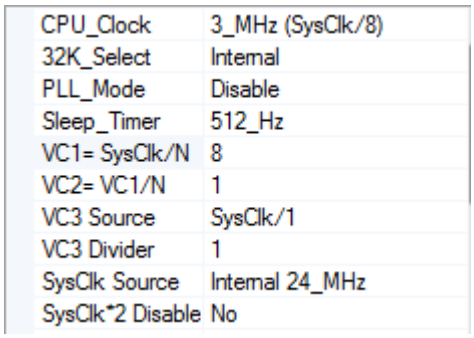
Figure 7. User Module Parameters



Clock Domain settings

Figure 8 shows the clock settings for this specific project.

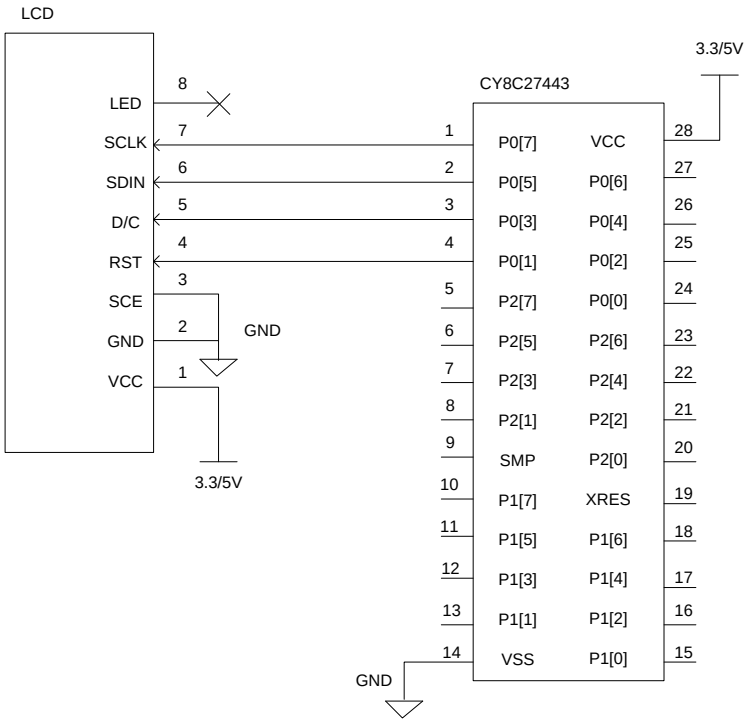
Figure 8. Clock Domain Settings



Hardware Setup

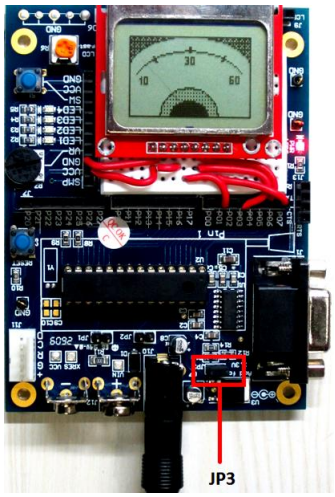
The circuit schematic shown in the Figure 9 explains the hardware connection details of PSoC 1 and the LCD interface. The LCD module is generally capable of supporting both 3.3 V and 5 V. f the LCD is not capable of supporting both the voltages, include an additional level translator.

Figure 9. Circuit Schematic for PSoC 1



RST is an active low reset signal. PSoC 1 drives the reset signal Low. SCE is an active low chip enable input to the LCD; it should be connected to GND. SCLK and SDIN of LCD are connected to the SCLK and MOSI of PSoC 1 respectively. Figure 10 shows the hardware setup. Jumper connection JP3 determines the supply voltage to PSoC 1. If JP3 is left open, 5 V is supplied to PSoC®1. If JP3 is shorted, 3.3 V is supplied to PSoC.

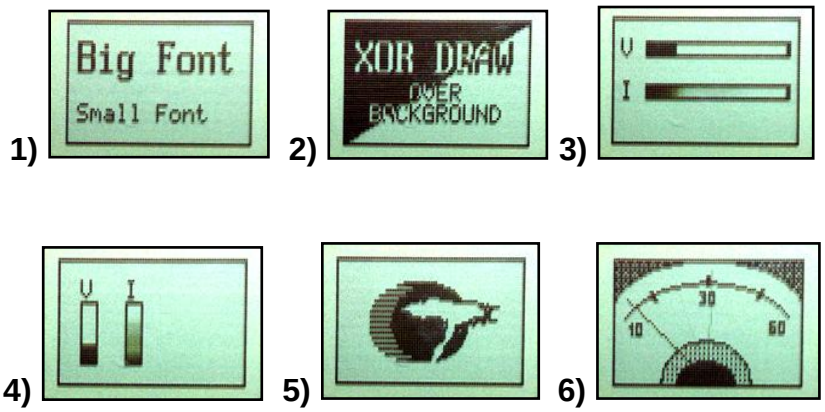
Figure 10. Hardware Setup



Test Methodology

1. Set up the hardware as mentioned in the Hardware Setup section.
2. Ensure that the supply voltage to PSoC is 5 V
3. The first demonstration screen shows big and small text writing on the LCD.
4. The second screen shows a text drawing using the DRAW_XOR parameter.
5. The third and fourth screens show horizontal and vertical drawings of bar graphs, respectively.
6. The fifth screen shows a bitmap drawing.
7. The sixth screen is an example of an analog gauge showing the line drawings with DRAW_OR and DRAW_CLEAR parameters.
8. Change the PSoC supply voltage to 3.3 V by shorting JP3 as mentioned in Hardware Setup. Repeat steps 3 to 7.

Figure 11. LCD Screenshots from Demo Application



Related Documents

Table 5 lists all relevant application notes, device datasheets, and user module datasheet.

Table 5. Related Documents

Application Note		
Document	**Title**	**Comment**
AN75320	Getting Started with PSoC® 1	Provides the basic information required to design with PSoC 1
AN32200	PSoC® 1 Clocks and Global Resources	Provides extensive information on the clocking structure in PSoC 1
AN56384	PSoC® 1 Segment LCD Direct Drive	Describes implementation of a software-based multiplexed segment LCD driver in a PSoC 1 device
Device Datasheet		
CY8C24x23	CY8C24123A,CY8C24223A, CY8C24423A: PSOC® Programmable System On Chip	CY8C24x23 family datasheet
CY8C27x43	CY8C27143, CY8C27243, CY8C27443, CY8C27543, CY8C27643: PSOC® Programmable System On Chip	CY8C27x43 family datasheet
User Module Datasheet		
SPI Master Datasheet		Describes the SPIM Standard and the working in PSoC1

Document History

Document Title: Interfacing PSoC® 1 with PCD8544 Graphics LCD Module - CE97630

Document Number: 001-97630

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4777500	RKRM	05/27/2015	New Code Example

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

PSoC is a registered trademark and 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, 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.