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.

AN56384

PSoC® 1 Segment LCD Direct Drive

Author: Rajiv Vasanth Badiger
Associated Project: Yes
Software Version: PSoC® Designer™ 5.4 CP1
Related Application Notes: None

AN56384 explains implementation of software based multiplexed segment LCD driver in PSoC® 1 device. PSoC 1 with its MCU and mixed signal resources offers segment LCD drive as one of the value added feature apart from implementing other major functions.

Contents

Introduction

Segment LCDs are available in two forms – segment LCD glass and the segment LCD module, which comes with an inbuilt driver. Many times, it is difficult to get all the required display features on a LCD module. One possibility is to use a custom LCD glass with an external driver. However, this increases the cost of the system. Cypress PSoC chip can do segment LCD glass drive besides executing some other major tasks with its configurable digital/analog hardware and with its 8-bit MCU. It integrates multiple functions of the system within a single chip offering significant BOM savings.

This application note explains technique to drive segment LCDs and explains how to create Segment LCD based PSoC project using PSoC Designer IDE tool. If you are new to PSoC 1 and PSoC Designer, it is recommended that you see Online Training.

Segment LCD Drive in PSoC 1

PSoC Designer provides an “SLCD” user module (UM) that can directly drive a multiplexed segment LCD. The SLCD UM has the following features:

- Drives LCD with ½ bias
- Supports 2, 3, and 4 common LCD
- 30–150 Hz refresh rate
- Supports Type A waveform
- Contrast control Feature
- Support for Numeric (7-segment), alphanumeric (14- and 16-segment), and special symbols

SLCD is a firmware based module where the CPU generates the ½ bias waveforms by configuring the pins and associated registers. To time the refresh events,

periodic interrupts are generated to the CPU using a timer. This timer is embedded within the module.

SLCD module provides two unique techniques to drive the LCD:

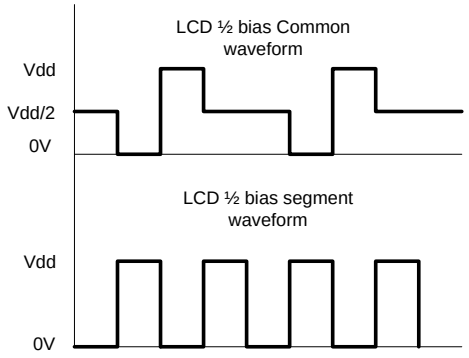
1. AMUX Drive
2. GPIO Direct Drive

Let's take a look at both these drive techniques.

AMUX Drive Method

This method implements ½ bias drive for LCD. In ½ bias, there are three voltage levels in the common/backplane drive signal - Vdd, ½ Vdd, and GND. There are two voltage levels in the segment drive signal - Vdd and GND. Figure 1 shows typical ½ bias LCD drive waveforms.

Figure 1. Voltage Levels in ½ Bias LCD Waveform



CPU bit bangs the pin to drive Vdd and GND level at common and segment lines. To output ½ Vdd level to the pin (at common line), SLCD module uses Analog MUX bus. Analog MUX bus is an electrical line in the chip which has the capability to connect to all the I/O pins. SLCD biases this bus to ½ Vdd and uses it to distribute to common pins. Bias is provided by an internal resource (using internal reference) or can be fed externally. This option can be selected in SLCD Configuration Wizard, which is discussed later.

Contrast Control

Contrast of LCD can be changed by changing the RMS voltage applied to the pixels. To change the RMS voltage, SLCD module introduces dead time between the LCD drive frames during which both the segment and the common pins are held at Vdd level leading to 0 V difference across all the pixels. Controlling the dead time, controls the RMS voltage across the pixel, thus controlling the contrast. In simple terms, SLCD turns pixels ON and OFF; controlling the ON and OFF time sets the contrast. Refresh rate is maintained same irrespective of dead time value.

Figure 2 - Left waveform shows the waveform of common and segment lines in AMUX method for a 2 common LCD at 100% and <100% contrast. The drive waveform with 100% contrast is typical ½ bias, 2 common LCD drive waveform.

Figure 2 - Right waveform shows the LCD drive with dead time. In the figure, α is the pulse width of active common time. To change the contrast, pulse width α is changed keeping the refresh rate constant. That means, α is decreased every time the dead time is increased and vice versa. However, UM limits the values of α to 500 us to avoid immediate interrupt to the CPU for the LCD waveform update.

$$\alpha_{min} = 500\ us$$

Maximum value, that α can have (with dead time inserted), is also limited and it is given by:

$$\alpha_{max} = \left(\frac{refresh\ period - 500\ us}{2 \times n}\right)$$

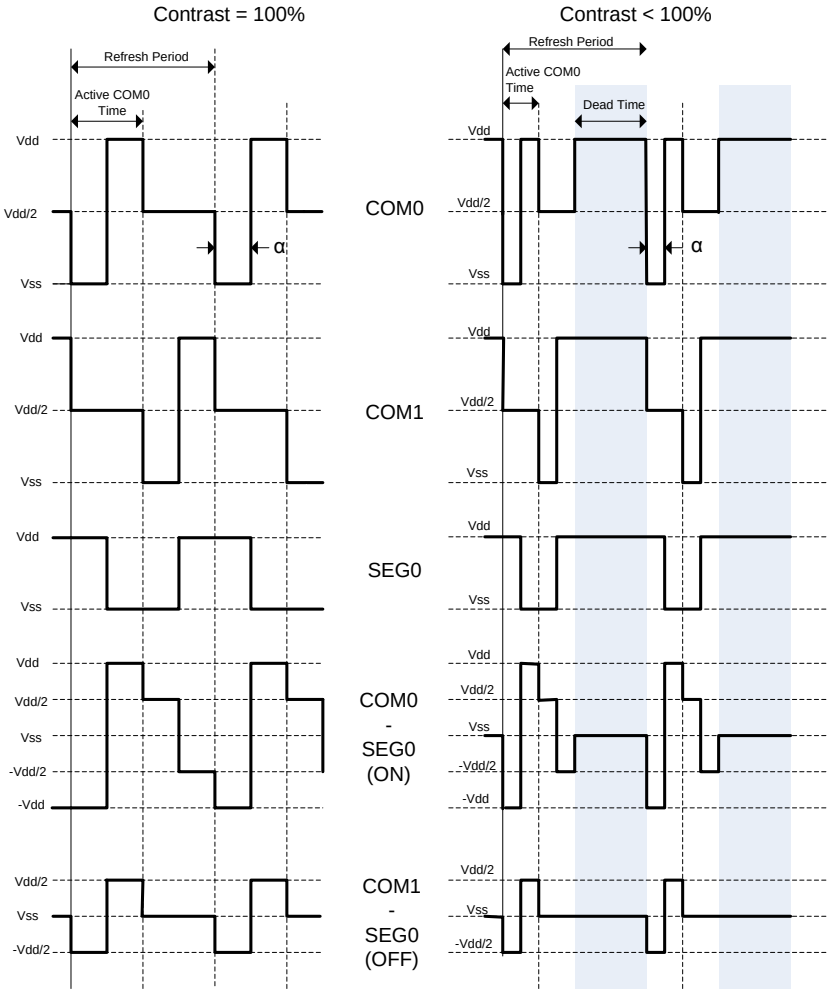
Where, n is the number of commons of LCD. This equation is calculated considering dead time of 500 us. Note that at 100% contrast, dead time is completely eliminated yielding maximum possible pulse width α.

The number of contrast levels depends on how precisely α can be varied between its limits. This depends on the timer input frequency and it is given by:

$$Contrast\ Levels = (\alpha_{max} - \alpha_{min}) \times Timer\ Input\ Freq$$

To get some numbers, let us take an example of 3 common LCD refreshed at 50 Hz. Let the timer input frequency be 400 kHz. This gives 1300 contrast levels.

Figure 2. LCD Waveforms in AMUX Method (2 commons)



GPIO Direct Drive Method

GPIO direct drive mode does not require AMUX bus for its operation. This method involves bit banging at the common and segment lines. There are only two voltage levels (Vdd and GND) in both common and segment waveform. Figure 3 shows the common drive waveforms. For segment to be ON, segment drive voltage is kept out of phase to the respective common drive voltage during active time (shaded region). The segment, which needs to be turned OFF, is given voltage in phase with common drive voltage during active common time (shaded region).

Figure 3. Common Drive Waveforms (GPIO Direct Drive Mode)

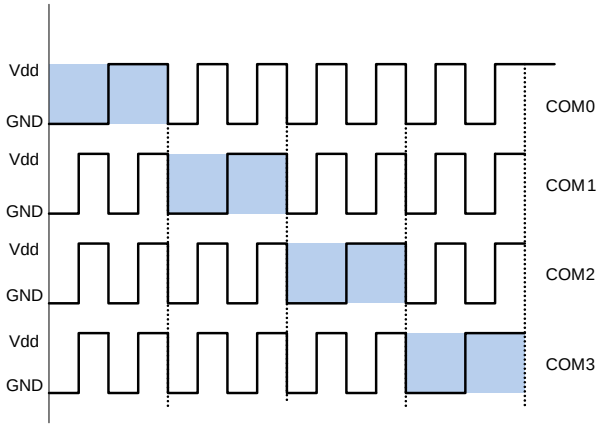


Figure 4 shows the detailed drive waveforms. As shown in waveform, for a pixel to be ON, the common-segment difference voltage is set as |Vdd| for the entire duration of active COM time by applying out-of-phase signals at the common and segment line. For a segment to be off, in-phase signals are applied at the segment and common line that provides zero difference voltage. The other pixel on the same segment line but a different common line gets Vdd voltage for ½ the active COM time. The RMS voltage of the OFF segment can be adjusted so that the pixel is completely blacked out.

Contrast Control

Contrast is varied by introducing dead time in the LCD refresh cycle similar to the AMUX method of LCD drive. In this method also, during dead time, both the segment line and common line are pulled to Vdd, creating zero difference potential across all the pixels. In this method, pulse width α is limited to 250 µs.

$$\alpha_{min} = 250\ us$$

Also, α_{max} is calculated, considering dead time to be 500 us.

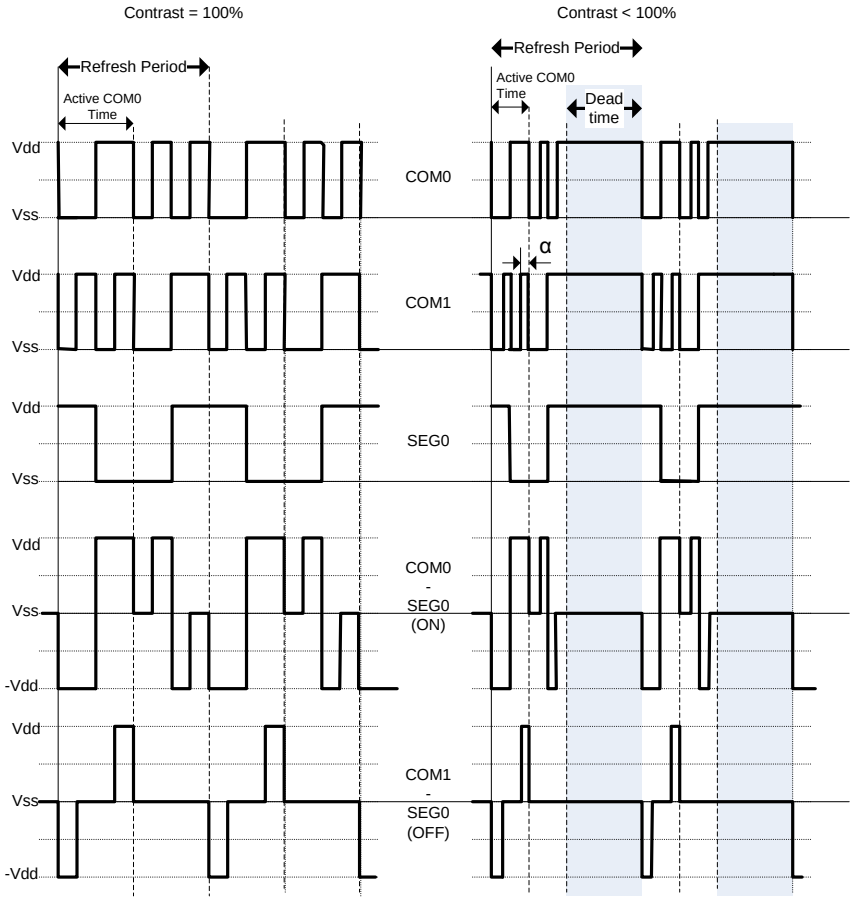
$$\alpha_{max} = \left(\frac{refresh\ period - 500\ us}{4 \times n}\right)$$

Number of contrast levels is given by:

$$Contrast\ Level = (\alpha_{max} - \alpha_{min}) \times F_T$$

Where n is the refresh period and F_T is the Timer input frequency.

Figure 4. LCD Waveforms in GPIO Direct Drive Mode (2 commons)



Comparison and Usage Cases of Methods

Following table gives the comparison of the AMUX drive and GPIO direct drive technique:

Parameters	AMUX Drive	GPIO Direct Drive
AMUX Bus Requirement	Required	Not required
Reference Requirement	Requires Vdd/2 reference voltage	Does not require any reference voltage
Contrast Ratio	(n+3)/(n-1)	(n+1)/(n-1)
	n is number of commons of LCD. AMUX technique provides more contrast ratio than GPIO technique.	
PSoC Device Applicability	Can be implemented in only devices equipped with analog MUX bus	Can be implemented in any PSoC device
	See Appendix B: PSoC Devices Capable of Implementing Segment LCD Drive for details	
Interrupts/sec	2 x n x refresh rate	4 x n x refresh rate
	GPIO technique provides twice the number of interrupts to the CPU as compared to AMUX technique in the same time period	
Resource Consumption	Digital blocks: 2/1 (depends on whether 16 bit or 8 bit timer is used) Analog blocks: 1 (if internal reference generator is used)	Digital blocks: 2/1 (depends on whether 16 bit or 8 bit timer is used) Analog blocks: 0

Usage Cases:

Use AMUX technique:

- If the device contains Analog MUX bus and if it is not used by any application
- If there are other interrupt sources in the system which are critical in operation (AMUX technique generates lesser number of interrupts)

Use GPIO technique:

- If the device does not have AMUX bus or it is used by other application such as capacitive touch sensing
- If low power is desired (it eliminates the use of reference generator which is always powered ON in AMUX method). This low power advantage, however, is somewhat faded by the more number of interrupts to the CPU. Note that CPU can be put to sleep between refresh events, interrupts from SLCD UM can be used to wake the device from sleep. See Low-Power Operation of SLCD.

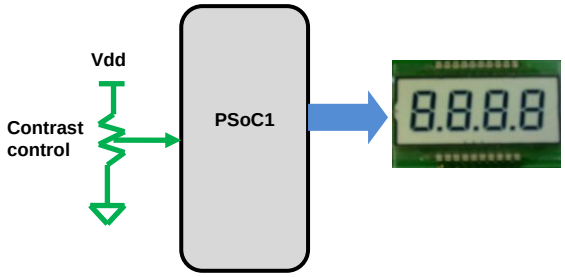
See Appendix B: PSoC Devices Capable of Implementing Segment LCD Drive to check what all PSoC devices can implement AMUX drive and GPIO drive technique.

PSoC Designer Project: AMUX Drive Mode

This section guides you to create PSoC Designer project based on CY8C28645 PSoC device with SLCD module configured in AMUX mode. You can use this procedure for other PSoC devices as well. See Appendix B: PSoC Devices Capable of Implementing Segment LCD Drive for the supported PSoC devices for AMUX drive method.

In this project, SLCD module is configured to drive a 3 common LCD – VIM 404. The details of this LCD are given in Appendix A: VIM 404 Segment LCD.

Figure 5. Block Diagram of the Function

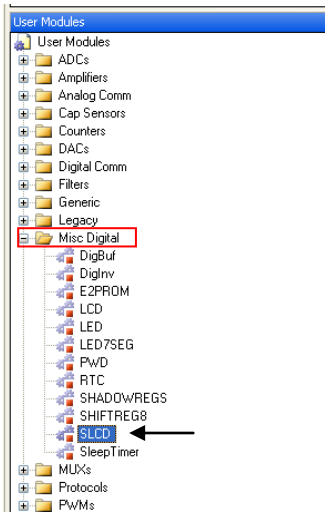


You will also learn to use contrast control feature. For contrast control, we can interface one potentiometer and read it using SAR ADC present in PSoC 1.

Following steps will guide you to create the project:

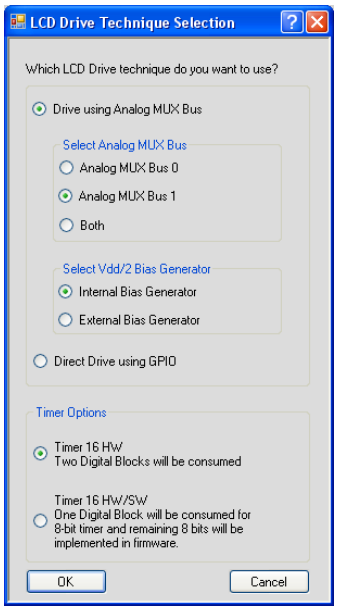
Step 1: Select the SLCD UM from user module Catalog in PSoC designer

Figure 6. UM Catalog



Double click on the UM to place it on the design. Following Drive Technique selection window opens:

Figure 7. LCD Drive Technique Selection Window



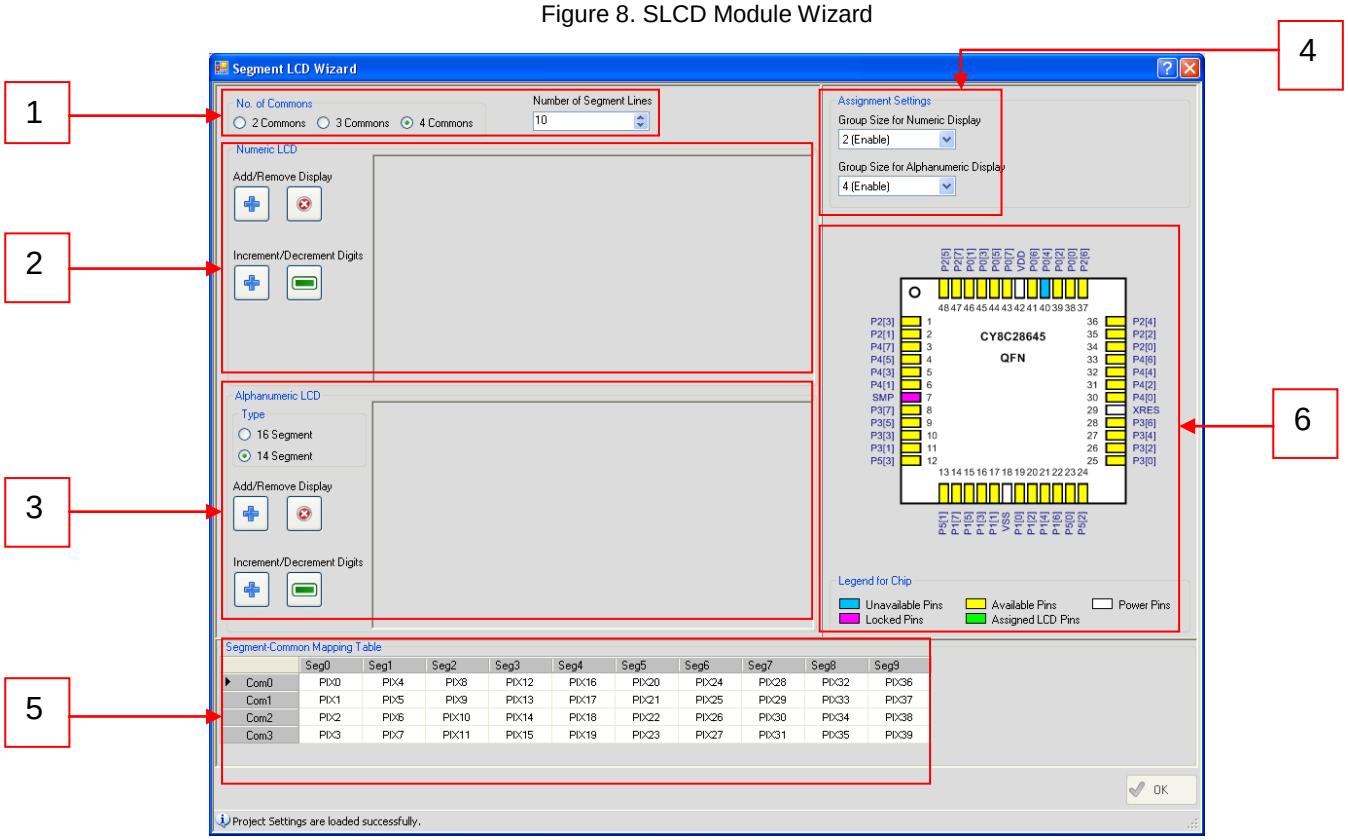
This is place where you select the particular drive technique. Besides this, you can also select the hardware timer resolution. Let us choose 16-bit HW timer and AMUX drive method with internal ½ bias generator and Analog MUX bus 1. This will consume one Analog block in the device for bias generation. If an external ½ bias generator is selected, Vdd/2 bias should be fed from an external source. Note that selecting Analog MUX 0 will allow only odd pins to be selected for LCD common lines and selecting Analog MUX bus 1 will only allow even pins. Two digital blocks are consumed for 16-bit timer. If 8 bit HW/SW option is selected, only one digital block is consumed and it makes use of software to implement the required resolution timer.

Step 2: Configuring the SLCD Module

SLCD module needs to be configured based on the LCD used. GUI based Wizard is provided with the module to configure the module. Right click on the module and select Wizard.

Figure 8 shows the wizard.

Figure 8. SLCD Module Wizard



1. **Number of commons and segments:** Set the number of common and segment lines here. VIM-404 LCD has 3 commons and 12 segment lines.
2. **Numeric LCD:** Here, you specify the number of 7-segment display sections and number of digits each section has. VIM-404 has only one 7-segment display section with 4 digits. Click on Add/Remove display buttons to specify number of 7-segment display sections and use increment/decrement digits button to specify number of digits in each display section. For this project, set 4 digits.
3. **Alphanumeric LCD:** This section is similar to 7-segment display section. Here you specify about 14-Segment or 16-segment alphanumeric display section. VIM-404 LCD does not have alphanumeric section. We will leave this blank for the project. Note that LCDs with both numeric and alphanumeric display sections are allowed by the user module.
4. **Group size:** Grouping refers to bringing together all the segment pins associated with display digit and assigning sequential pins. This results in less CPU time in updating the segments. Group size is the number of segment lines per digit of numeric or alphanumeric display. As far as possible, enable grouping. Let us enable grouping in this project.
5. **Segment-Common mapping table:** This table carries the segment-common mapping information. You should see the LCD datasheet for mapping information. LCD VIM-404 segment common mapping information is shown in the following table:

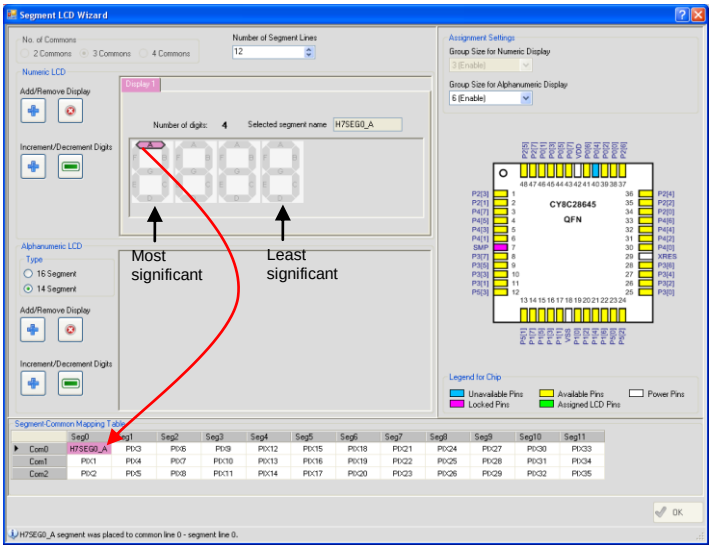
Table 1. VIM-404 Segment Common Mapping Table

LCD Pin		18	4	19	16	5	17	14	6	15	12	7	13
		SEG0	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	SEG10	SEG11
11	COM0	1A	1B	1F	2A	2B	2F	3A	3B	3F	4A	4B	4F
20	COM1	1G	1C	1E	2G	2C	2E	3G	3C	3E	4G	4C	4E
8	COM2	1D	1P	NC	2D	2P	NC	3D	3P	NC	4D	NC	NC

1A, 1B, 1F, and so on, are the segments of most significant digit whereas 4A, 4B, 4F, and so on, are segments of least significant digits. Assign names to the segment lines (SEG0 - SEG11 as shown in Table 1). This helps in configuring the mapping table in the wizard. Note that there is no fixed order to assign the names to the segment lines. However care should be taken to place all the segments of the group under same segment name.

To fill the table, click on the segment in display section and drag it on to the cell of mapping table. See Figure 9. Note that when grouping is enabled, depending on the mapping of segments of first digit, segment mapping of other digits will be restricted to specific cells.

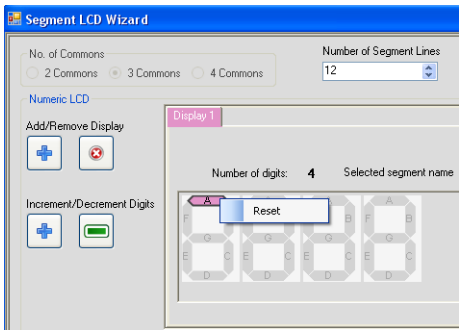
Figure 9. Segment-Common Mapping Procedure



Note that the most significant digits are prefixed by 0 (H7SEG**0**_A in Figure 9) and should not be confused with the segment names given in LCD datasheet (1A in Table 1).

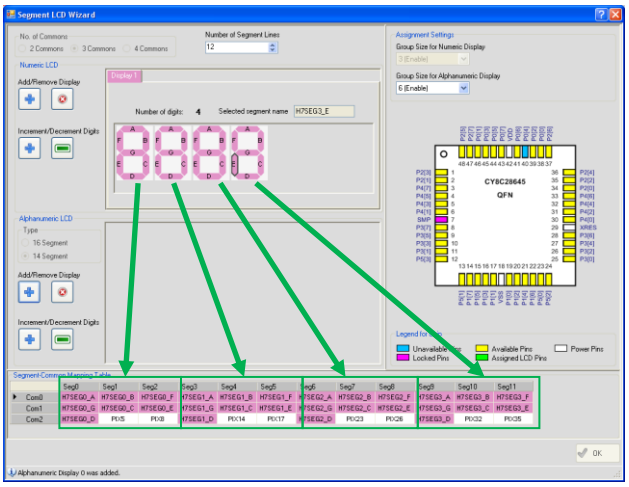
To reset the already made mapping, right click on the segment and use reset option.

Figure 10. Resetting the Mapping



The completed mapping table is shown below:

Figure 11. Complete Mapping Table



Mapping Symbols

In case of special symbols on the display (like decimal point in this LCD), you just need to rename the cell of the pixel mapping table by entering directly in the cell. PSoC Designer generates pointer to these symbols with the specified name. Renaming the symbols helps in easy identification of those pointers which is used for controlling the pixels.

All the symbols are renamed as shown in the following figure.

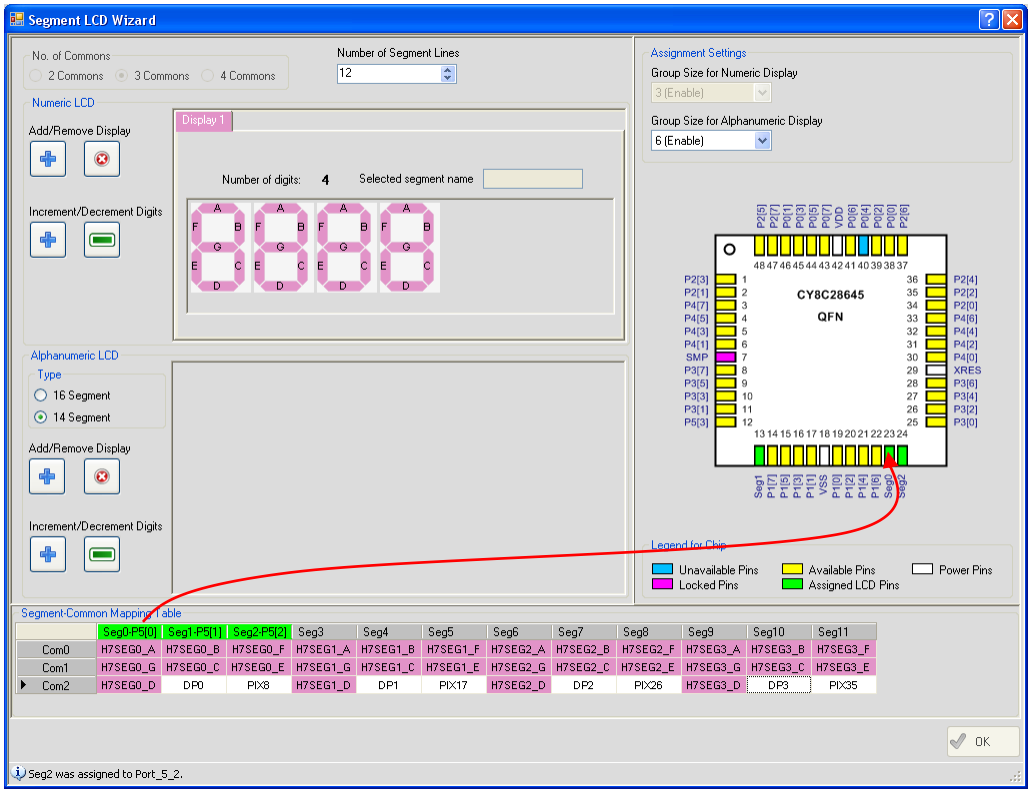
Figure 12. Mapping Symbols

Segment-Common Mapping Table

	Seg0	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Seg9	Seg10	Seg11
Com0	H7SEG0_A	H7SEG0_B	H7SEG0_F	H7SEG1_A	H7SEG1_B	H7SEG1_F	H7SEG2_A	H7SEG2_B	H7SEG2_F	H7SEG3_A	H7SEG3_B	H7SEG3_F
Com1	H7SEG0_G	H7SEG0_C	H7SEG0_E	H7SEG1_G	H7SEG1_C	H7SEG1_E	H7SEG2_G	H7SEG2_C	H7SEG2_E	H7SEG3_G	H7SEG3_C	H7SEG3_E
Com2	H7SEG0_D	DP0	PIX8	H7SEG1_D	DP1	PIX17	H7SEG2_D	DP2	PIX26	H7SEG3_D	PIX32	PIX35

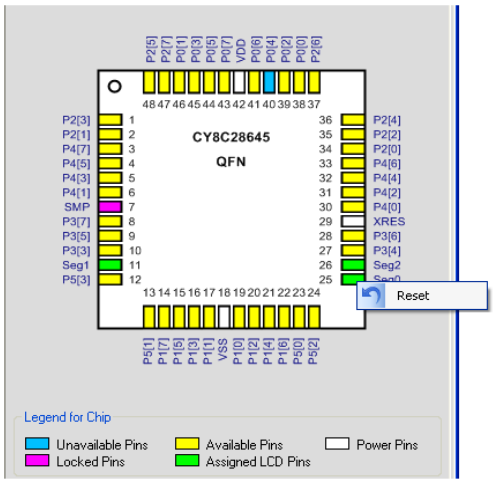
6. **Pin assignment:** To assign the pins for segment and common lines, click on the segment and common label. Drag and drop on the appropriate pin of chip legend diagram. See Figure 13. As grouping is enabled here, if you choose one of the segments, all the other segments of the group will be automatically assigned adjacent pins of the port. Also note that only even numbered pins are possible for common lines as Analog MUX bus 1 is selected for SLCD. The common pins are allowed to be selected from a single port. This is done to reduce the CPU time in updating the common voltage. Select the pins for the segment and common lines based on the board design.

Figure 13. Assigning Pins to Segment and Common Lines



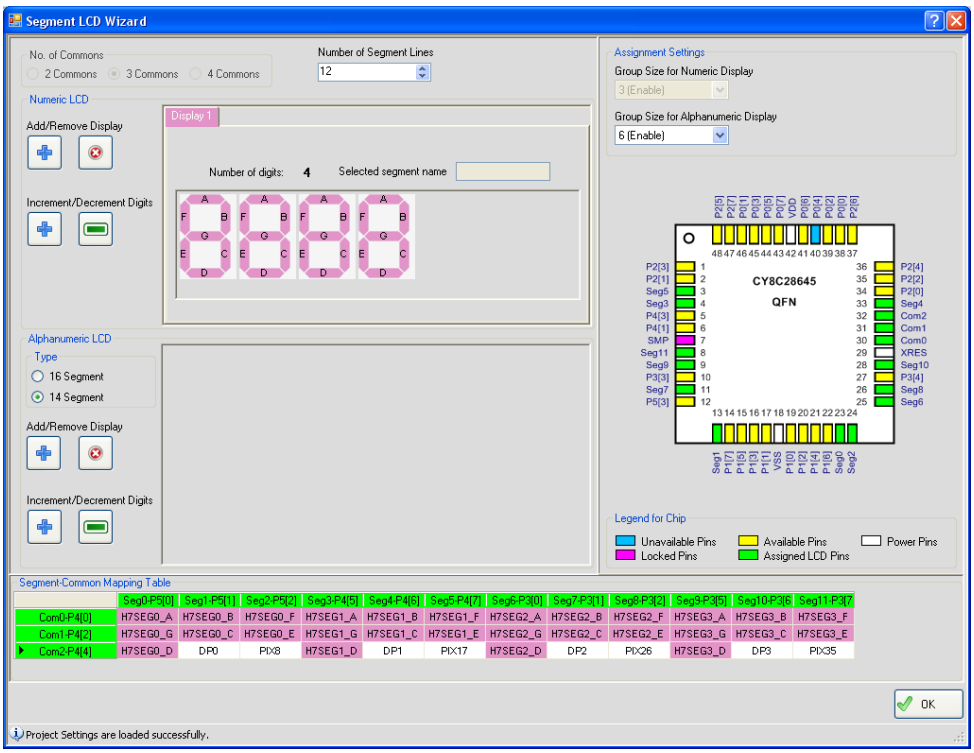
To undo the pin selection, right click on the pin and use reset option. See Figure 14.

Figure 14. Resetting Selected Pin



The completed configuration looks similar to that shown in Figure 15. Click OK to apply the settings.

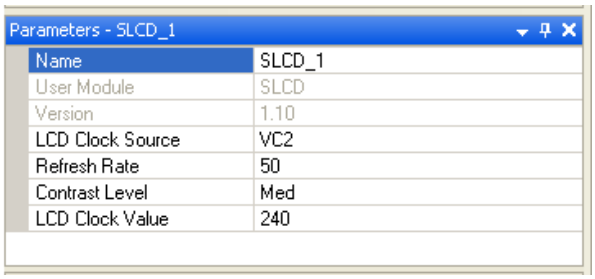
Figure 15. Completed SLCD Wizard



Step 3: Now we need to set the properties of the SLCD module. Here, you set the refresh rate, contrast level, and clock information.

Following figure shows the properties window:

Figure 16. SLCD Properties

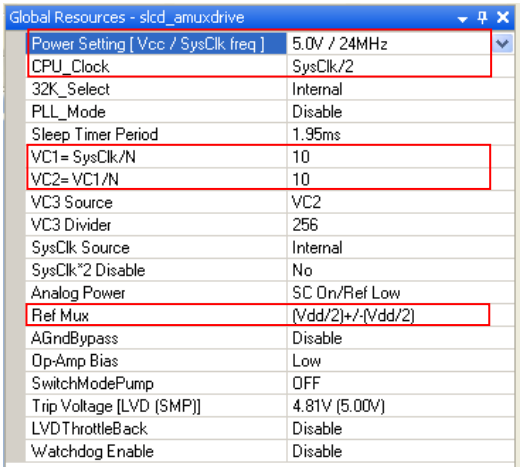


- In this project, refresh rate is set to 50 Hz and contrast level is set to medium. Note that you can change the contrast in firmware.
- Let us use VC2 clock of 240 kHz for the module. You need to set 240 kHz for VC2 in global resources in next step. As you set higher frequency, you get high number of contrast levels as explained earlier (see Page 2). The valid frequency range is 100 kHz to 2 MHz.

Step 4: Global Resource settings

Here you set the clock dividers and power related parameters.

Figure 17: Global Resource Settings

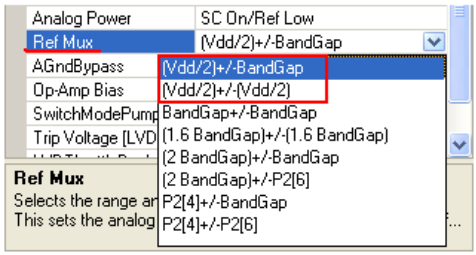


- System clock is set to 24 MHz. CPU clock is set to 12 MHz. It is important that you set CPU clock as high as possible to avoid latency issues in servicing other interrupts in the system.
- In the properties of the SLCD module, the LCD clock source is set to VC2 and the clock value is set to 240 kHz. VC2 of 240 kHz is generated by setting the VC1 and VC2 divider to 10 each.
- As we are using the internal reference generator, AGND value needs to be set using the Ref MUX parameter in global resources. Ref MUX parameter sets Reference High, Reference Low, and Analog ground for the application. It is in the format given below:

AGND ± Reference

Figure 18 shows the two options with Vdd/2 as AGND in Global resource settings. It should be set to one of these two options.

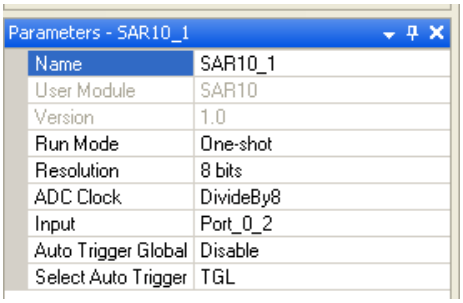
Figure 18. Ref MUX Options



Using internal bias generator introduces some power consumption; significant portion is from analog output buffer. To reduce the power consumption, use an external bias generator with a resistor-based potential divider. The pin to which Vdd/2 bias needs to be provided is selected in SLCD Wizard.

Step 5: For changing the contrast, let's interface a potentiometer and read it using ADC. The selected PSoC device CY8C28645 provides SAR ADC. Select and configure the SAR ADC. Input pin is configured to P0[2]. You need to connect potentiometer or any other voltage source at pin P0[2].

Figure 19. SAR10 UM Properties



This completes the hardware configuration of the device. The status of analog and digital blocks can be seen in the following figure:

Figure 20. Analog Blocks Consumption

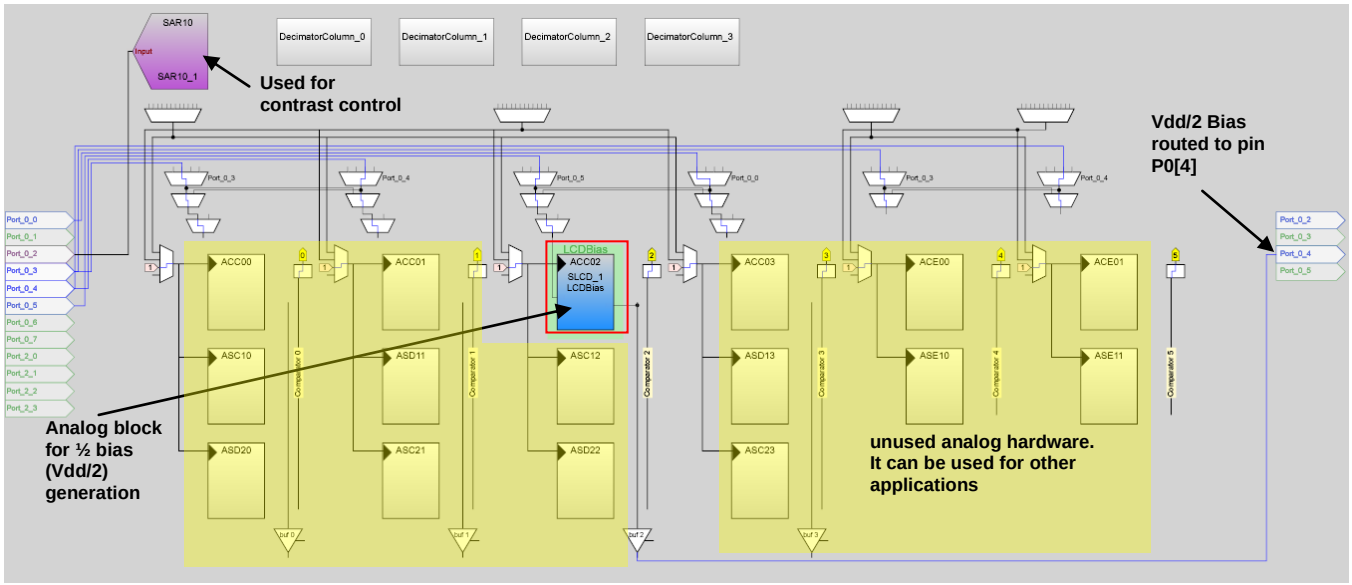
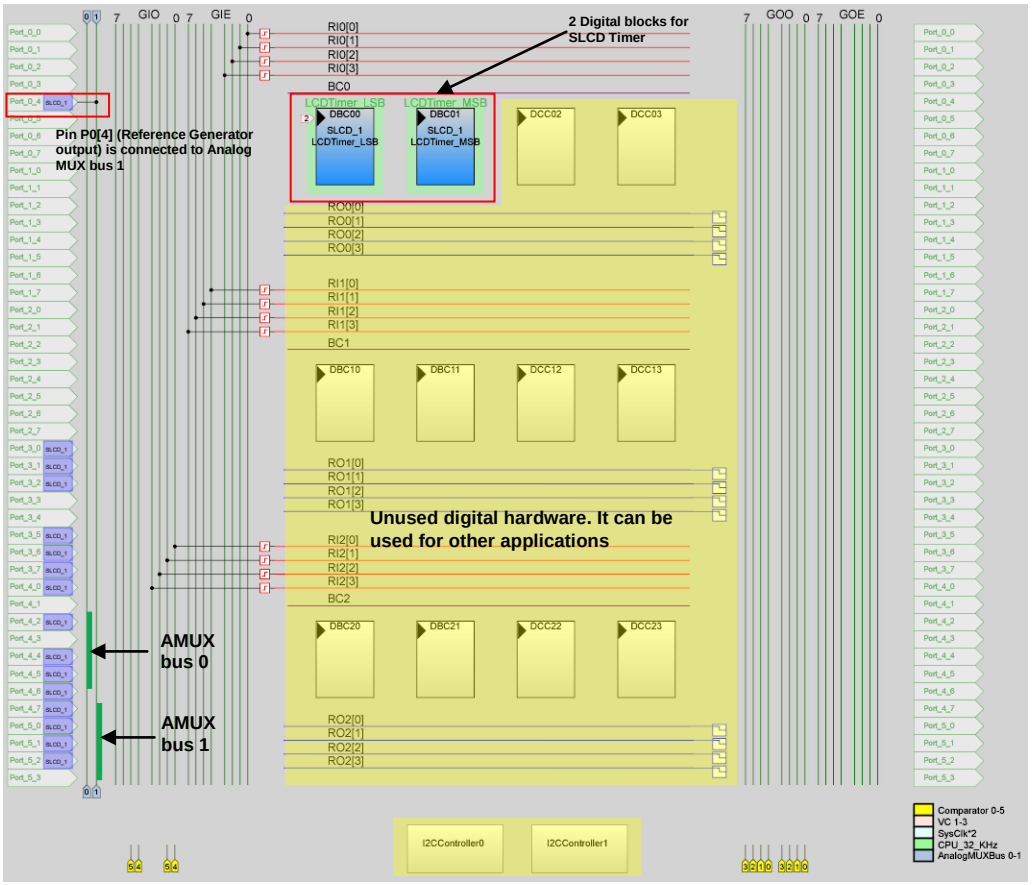


Figure 21. Digital Blocks Consumption



Step 6: Firmware

SLCD UM provides wide range of functions for all the display sections - Numeric, 14-segment, 16-segment, and symbols. See SLCD UM datasheet for all the APIs.

As an example, incrementing values are printed on LCD and ADC value is used to adjust the contrast. See Appendix C: *Main.c* Code for *main.c* code.

APIs used in the project:

Printing decimal number on 7 segment display section:

```
void SLCD_1_PrintDecNumber(BYTE DisplayID,
int Number);
```

This function prints integer value (Number) on LCD display section (pointed by DisplayID) in decimal format. There can be multiple numeric display section and/or alphanumeric section. DisplayID is used to point to particular display section on LCD. You can find the display ID in user module *.h* file. User module *.h* and *.c* files are created after generated the project.

Controlling Symbols:

```
void SLCD_1_EnableSymbol(BYTE SegmentID,
BYTE ON_OFF);
```

This API is used to control the pixel specified by SegmentID. See user module *.h* file to find the pixel name. If you have named the pixel as suggested in step 2 (segment-common mapping), it will be easy to identify the pixel in the mapping table.

Changing LCD Contrast:

```
BYTE    SLCD_1_ChangeContrast(BYTE    OPTION,
BYTE Delta);
```

This API is used to change the contrast of LCD. “OPTION” can either be presets - MAX, MIN, or Med or increase/decrease contrast option. When increase or decrease contrast option is given, it is necessary that Delta parameter is provided. Delta parameter is desired change in contrast levels. As you provide high delta parameter, contrast changes with high percentage. For fine control in contrast, provide low values of delta (<10).

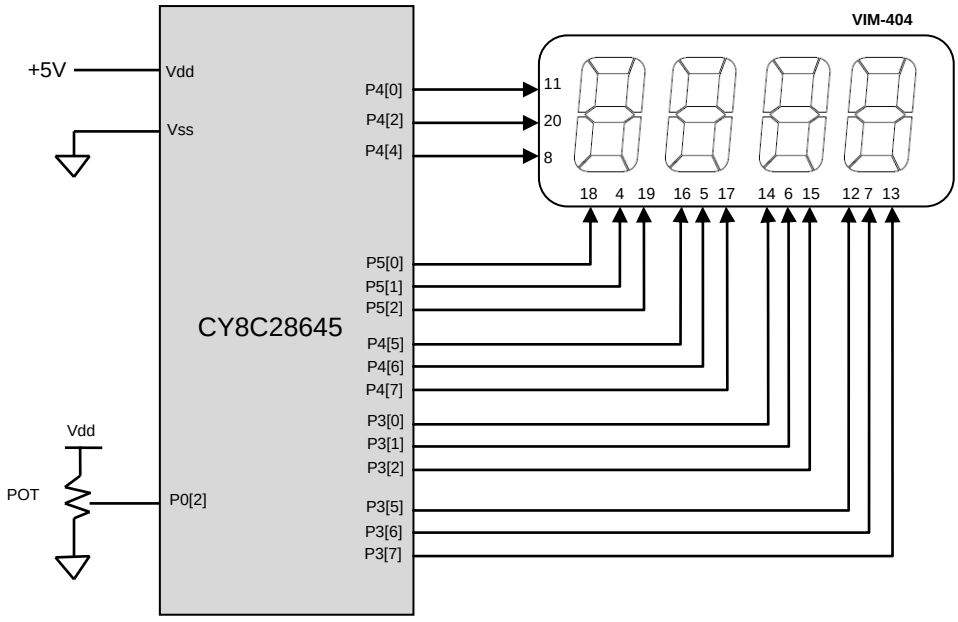
In the project, new function “SetContrast” is provided which takes percent contrast. It uses SLCD_ChangeContrast API of the module.

See Appendix C: Main.c Code for firmware.

Step 7: Build and program the chip to test the design

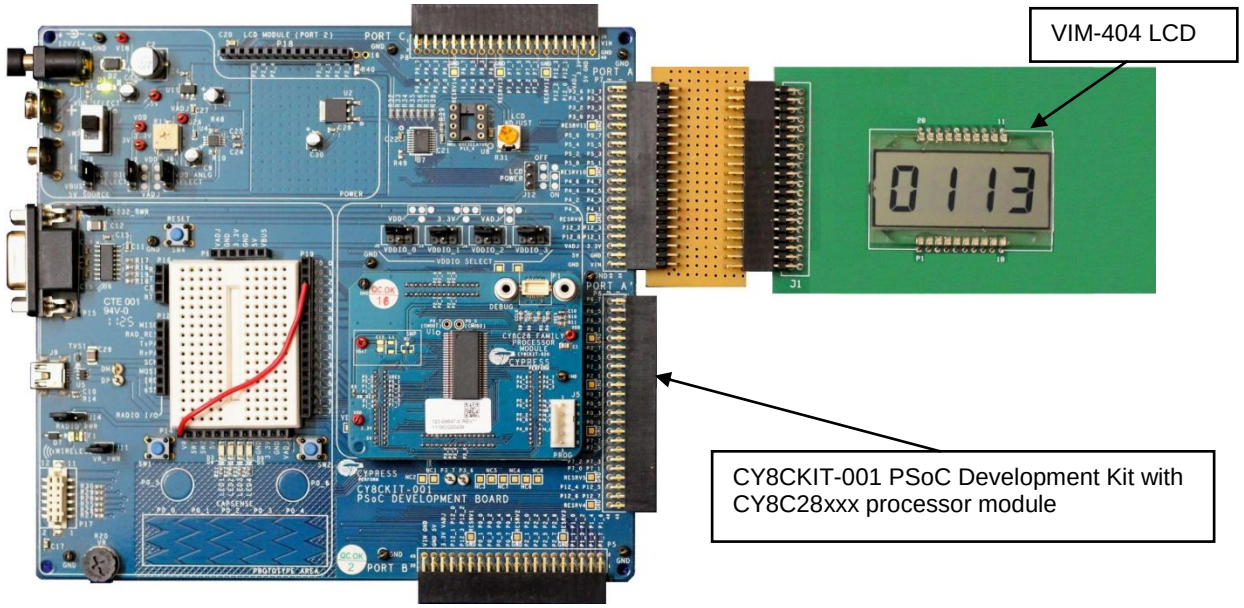
External connections of PSoC are shown in Figure 22. External connection has to be made based on the pin assignment in Step 2.

Figure 22. External Connections



PSoC Designer project “SLCD_AMUXDrive” is provided for your reference. You can test this project on CY8CKIT-001 PSoC Development Kit. Test setup is shown in Figure 23.

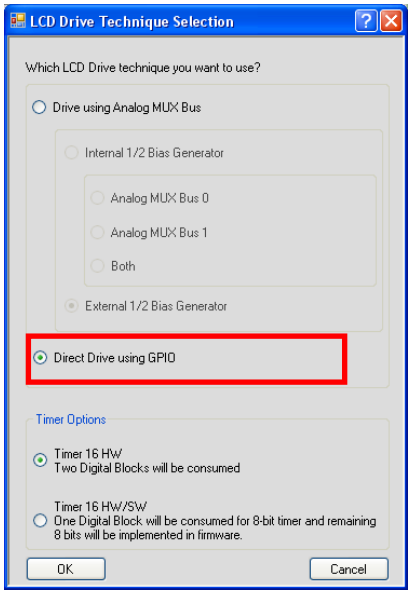
Figure 23. Test Setup



PSoC Designer Project: GPIO Direct Drive Mode

Creating a project with SLCD in GPIO mode is similar to the way you create the project with SLCD in AMUX mode. When you select the SLCD module, you need to set GPIO direct drive in the LCD Technique section window.

Figure 24. SLCD Drive Technique Selection

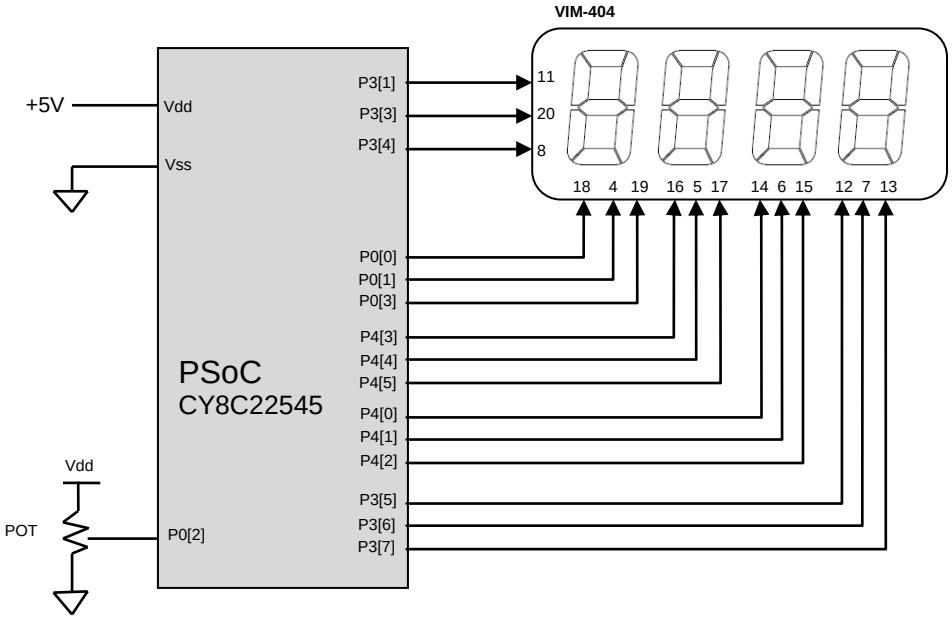


The rest of the procedure is same as the one described earlier. Note that the design consumes only digital blocks in this case; to implement Timer and no analog blocks as GPIO direct drive does not require any bias generator. So you need not worry about Ref MUX in Global settings described in Step 4.

For your reference, project “SLCD_GPIODrive” for target PSoC chip CY8C22545 is provided. This project interfaces to same LCD VIM-404 (details of this LCD given in Appendix A: VIM 404 Segment LCD). In this project, “grouping” is disabled. This enables to use any pins for segment lines. Also as SLCD in GPIO direct drive mode does not use AMUX bus, we can use any pins for commons lines. But it should be from same port.

This project can be tested on CY8C3280-22x Universal Capsense Controller board. Use the external connections as shown in Figure 25.

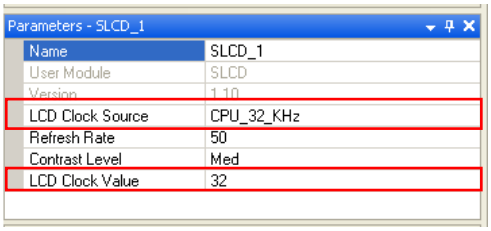
Figure 25. External Connections



Low-Power Operation of SLCD

All portable systems running from a battery requires all its functions to be efficient. PSoC has the feature to keep SLCD module alone active with all other resources put to sleep. This helps to keep LCD active even when chip is sleeping. This requires clock that runs the SLCD timer to be active when the device is sleeping. In PSoC, internal low speed oscillator (ILO) of 32 kHz frequency is one clock which is always active. SLCD module should be configured to run with this clock. This clock can be selected in SLCD module properties as shown in the following figure:

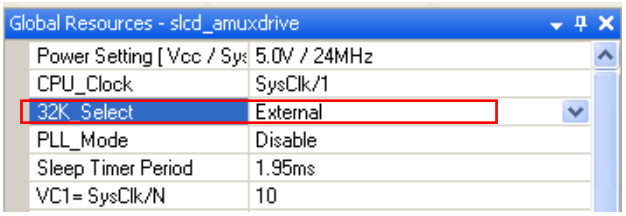
Figure 26. SLCD Properties for Low-Power Operation



ILO is designed for low power. But it is not accurate. It can be vary anywhere between 15 kHz to 64 kHz. To obtain more accuracy for the clock frequency, use the PSoC external crystal oscillator (ECO) logic, which requires a 32.768 kHz external watch crystal. To use the ECO:

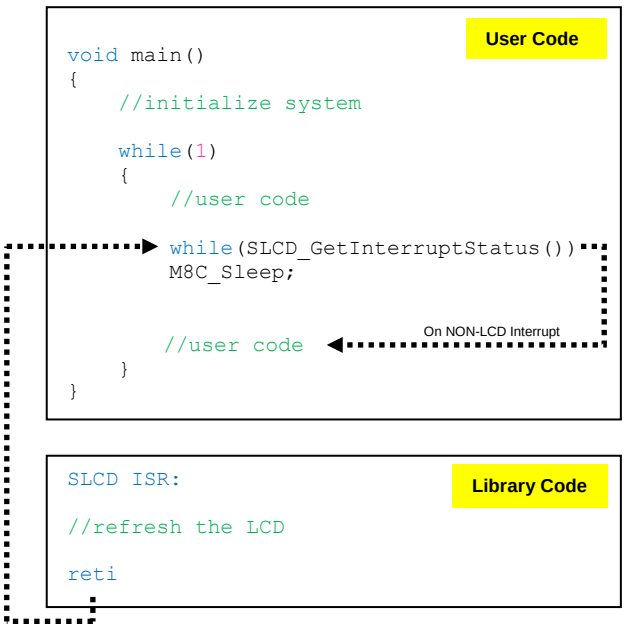
- Connect crystal at P1[0] and P1[1] pins
- Set the **32K_select** parameter in global resources to **External** as shown in Figure 27.

Figure 27. Global Resource Settings



Firmware Design

In firmware, periodic sleep and wake up scheme can be implemented to reduce the power consumption. SLCD timer (sourced from ILO) can be used to wake the device from sleep. SLCD timer issues interrupt periodically to refresh the LCD. However, there can be multiple interrupt sources in a solution. To check if SLCD has issued interrupt, you can use the *GetInterruptStatus()* API provided by SLCD module. The following code snippet explains the basic firmware architecture, which can be adopted for low power solutions. This code puts the device back to sleep mode after LCD refresh each time the device wakes up due to SLCD interrupt. If the interrupt is due to other sources, it comes out of the loop and executes the user code. Use this kind of firmware architecture when low power is desired.



Interrupt Service Routines (ISRs)

As PSoC 1 segment LCD drive is firmware based technique, LCD refresh action takes place in the ISR of the timer. When the application has other ISRs, it should be kept as short as possible to avoid latency in LCD refresh. Large amount of delay in execution of SLCD ISR causes asymmetry in common and segment waveforms, causing offset voltage (see SLCD user module datasheet for details). Here are some tips:

- Keep CPU clock high (>=12 MHz) to decrease latency in SLCD ISR entry. This also creates CPU bandwidth for other applications.
- If other user ISRs has lengthy code it is recommended to move the less time critical tasks to *main.c* and execute the code from *main.c*. This can be done by polling for the flag (set in interrupt routine) in main loop.
- Avoid using high refresh rates when operating CPU at lower than 12 MHz

Summary

This application note introduces two drive techniques encapsulated in SLCD UM offering quick designs that can be made with less effort and cost.

About the Author

Name:	Rajiv Vasanth Badiger
Title:	Sr. Applications Engineer
Background:	BE-Electronics and Communication
Contact:	rjvb@cypress.com

Appendix A: VIM 404 Segment LCD

VIM 404 features:

- Four Digit 7 segment LCD
- 3 commons
- TN-Twisted Nematic
- Reflective

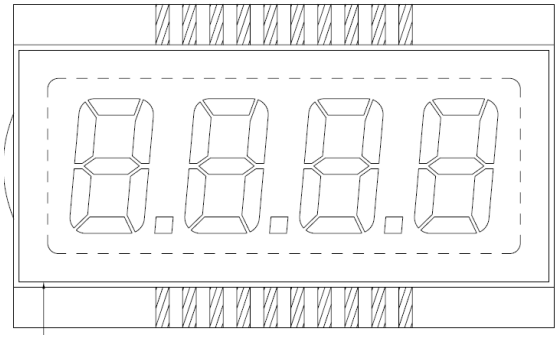


Table 2. Segment-Common Mapping

Pin	COM1	COM2	COM3
1-3	NC	NC	NC
4	1B	1C	1P
5	2B	2C	2P
6	3B	3C	3P
7	4B	4C	-
8	-	-	COM3
9-10	NC	NC	NC
11	COM1	-	-
12	4A	4G	4D
13	4F	4E	-
14	3A	3G	3D
15	3F	3E	-
16	2A	2G	2D
17	2F	2E	-
18	1A	1G	1D
19	1F	1E	-
20	-	COM2	-

Appendix B: PSoC Devices Capable of Implementing Segment LCD Drive

PSoC Device	Number of AMUX Bus Available	SLCD (AMUX Drive Method)	SLCD (GPIO Direct Drive Method)
CY8C20x34	1	✓	✓
CY8C20xx6	1	✓	✓
CY8C21x23	0	X	✓
CY8C21x34	1	✓	✓
CY8C21345	2	✓	✓
CY8C22x45	2	✓	✓
CY8C23x33	0	X	✓
CY8C24x23	0	X	✓
CY8C24x94	2	✓	✓
CY8C27x43	0	X	✓
CY8C28xxx	2	✓	✓
CY8C29x66	0	X	✓

Appendix C: *Main.c* Code

```
********************************************************************************
*  Project Name: SLCD_AMUXDrive
*  Device Tested: CY8C28645/CY8C22545
*  Software Version: PSoC Designer 5.4 CP1
*  Complier tested: ImageCraft
*  Related Hardware: CY8CKIT-001/CY3280-22xxx
*  Related Application Note: AN56384: PSoC1 Segment LCD Direct Drive
********************************************************************************/

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules


/* Edit these parameters based on the application */
#define REFRESH_RATE 50           //Hz
#define TIMER_INPUT_FREQ 240 //KHz
#define COMMONS 3

/* Calculates maximum contrast levels */
#define ALPHA_MAX ((1.0/REFRESH_RATE)-500.0/1000000)/(2*COMMONS)
#define ALPHA_MIN (500.0/(1000000))
#define MAX_CONTRAST_LEVELS   TIMER_INPUT_FREQ*(ALPHA_MAX-ALPHA_MIN)*10
/* NOTE: This will give number of contrast levels / 100  */

/* This function changes the contrast. It takes value in percent */
void SetContrast(BYTE);

BYTE PresentContrastValue=50; /* Initially contrast is set to medium */

BYTE OneSecFlag;/* This flag is set on sleep timer interrupt. Sleep timer is configured to
1sec interval in firmware */


void main(void)
{
    BYTE ADCValue=0;
    unsigned int DisplayCount=0;

    /* Enable global interrupt */
    M8C_EnableGInt ;

    /* Configure sleep timer is generate 1 sec intervals.  */
    /* Note that sleep timer is just used to generate 1sec intervals. Device is not put to
    sleep in the project */
    OSC_CR0|=0x18;

    /* Enable sleep timer interrupt */
    INT_MSK0|=0x40;

    /* Start SLCD module */
    SLCD_1_Start(SLCD_1_NORMAL_STATE);

    /* Initially set contrast to medium */
    SLCD_1_ChangeContrast(SLCD_1_SET_TO_MED,0);
```

```
    /* Start SAR ADC */
    SAR10_1_Start();

    /* Turn ON all segments */
    SLCD_1_PrintDecNumber(0,8888);
    SLCD_1_EnableSymbol(SLCD_1_DP0_ID,1);
    SLCD_1_EnableSymbol(SLCD_1_DP1_ID,1);
    SLCD_1_EnableSymbol(SLCD_1_DP2_ID,1);

    /* 2 sec Delay */
    while(OneSecFlag==0);
    OneSecFlag=0;
    while(OneSecFlag==0);
    OneSecFlag=0;

    /* Turn OFF all the segments */
    SLCD_1_ClearAll();

    /* Print count in decimal format on LCD */
    SLCD_1_PrintDecNumber(SLCD_1_DISPLAY_ID_1,DisplayCount);



    while(1)
    {
        /* Trigger ADC to start conversion */
        SAR10_1_Trigger();

        /* Wait till ADC result is available */
        while(SAR10_1_fIsDataAvailable()==0);

        /* Read ADC */
        ADCValue=SAR10_1_bGetData();

        /* Adjust the contrast. 8 bit ADCValue is divided by 2 to get close to 0-100 range.
*/
        SetContrast(ADCValue>>1);

        /* OneSecFlag is set every 1 sec on sleep timer interrupt */
        if(OneSecFlag)
        {
            OneSecFlag=0;

            /* Increment display count */
            DisplayCount++;
            if(DisplayCount > 9999)
                DisplayCount = 0;

            /* Print count in decimal format on LCD */
            SLCD_1_PrintDecNumber(SLCD_1_DISPLAY_ID_1,DisplayCount);
        }
    }
}
```

```
/* This is the interrupt handler for sleep timer interrupt */

#pragma interrupt_handler OneSecInterrupt
void OneSecInterrupt(void)
{
    /* This flag is polled in main.c */
    OneSecFlag=1;
}


/* This function sets the contrast. It takes the contrast request in percentage (0-100). This
function uses ChangeContrast API provided by the SLCD user module */

void SetContrast(BYTE Contrast)
{
    unsigned int Delta; /* represents change in contrast levels requested */
    BYTE i;

    /* If the value passed to this function is greater than 100, then limit it to 100. */
    if(Contrast > 100)
    {
        Contrast=100;

        /* Set contrast to maximum */
        SLCD_1_ChangeContrast(SLCD_1_SET_TO_MAX,0);
    }
    else
    /* Check whether contrast requested is more than presently set */
    if(Contrast > PresentContrastValue)    /* More contrast requested */
    {
        /* Calculate the change in contrast levels requested */
        Delta=(unsigned int)(MAX_CONTRAST_LEVELS*(Contrast-PresentContrastValue));

        /* This logic is implemented as ChangeContrast API provided by SLCD user module
        accepts only 8 bit value. This logic calls ChangeContrast API multiple times based on
        Delta Value */
        i=Delta/0xff;
        while(i)
        {
            /* Increase contrast by 255 levels */
            SLCD_1_ChangeContrast(SLCD_1_INCREASE_CONTRAST,0xff);
            Delta-=0xff;
            i--;
        }
        /* Increase contrast by "Delta" value */
        SLCD_1_ChangeContrast(SLCD_1_INCREASE_CONTRAST, Delta);
        /*****************************************************************/
    }
    else
    if(Contrast==0)
    {
        /* Set contrast to minimum */
        SLCD_1_ChangeContrast(SLCD_1_SET_TO_MIN,0);
    }
    else
    if(Contrast < PresentContrastValue)        /* Less contrast requested */
    {
        Delta=(unsigned int)(MAX_CONTRAST_LEVELS*(PresentContrastValue-Contrast));

        /* This logic is implemented as ChangeContrast API provided by SLCD user module
```

```
        accepts only 8 bit value. This logic
        calls ChangeContrast API multiple times based on Delta Value */
        i=Delta/0xff;
        while(i)
        {
            /* Decrease contrast by 255 levels */
            SLCD_1_ChangeContrast(SLCD_1_DECREASE_CONTRAST,0xff);
            Delta-=0xff;
            i--;
        }
        /* Decrease contrast by "Delta" value */
        SLCD_1_ChangeContrast(SLCD_1_DECREASE_CONTRAST, Delta);
        /*****************************************************************/
    }

    PresentContrastValue=Contrast;
}
```

Document History

Document Title: PSoC® 1 Segment LCD Direct Drive – AN56384

Document Number: 001-56384

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2768545	RJVB	09/23/2009	New Spec.
*A	3047191	RJVB	01/19/2011	Changed entire document and project due to new UM (SLCD).
*B	3286104	RJVB	06/17/2011	Changed the title to PSoC® 1 Segment LCD Direct Drive Removed CapSense from the projects. AN explains only about segment LCD drive using SLCD UM Added more explanation about AMUX and GPIO drive Added section Low-Power Operation of SLCD. Updated as per template.
*C	3457922	RJVB	12/07/2011	Updated document and project with PSoC Designer 5.2. Updated template.
*D	3587660	RJVB	19/03/2012	Replaced CY3280-28xxx kit with CY8CKIT-001 Corrected Figure 8
*E	3769777	RJVB	10/10/2012	Updated in new template.
*F	4530202	VAIR	10/09/2014	Updated the document and the projects to PSoC Designer 5.4 CP1.

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
Optical Navigation Sensors	cypress.com/go/ons
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 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

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

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