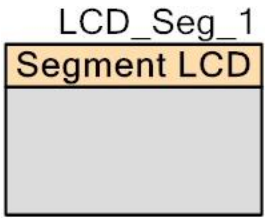
Segment LCD (LCD_Seg)
3.40



Features

- 2 to 768 pixels or symbols
- 1/3, 1/4 and 1/5 bias supported
- 10- to 150-Hz refresh rate
- Integrated bias generation between 2.0 V and 5.2 V with up to 128 digitally controlled bias levels for dynamic contrast control
- Supports both type A (standard) and type B (low power) waveforms
- Pixel state of the display may be inverted for negative image
- 256 bytes of display memory (frame buffer)
- User-defined pixel or symbol map with optional 7-, 14-, or 16-segment character; 5x7 or 5x8 dot matrix; and bar graph calculation routines.

General Description

The Segment LCD (LCD_Seg) component can directly drive a variety of LCD glass at different voltage levels with multiplex ratios up to 16x. This component provides an easy method of configuring the PSoC device to drive your custom or standard glass.

Internal bias generation eliminates the need for any external hardware and allows for software-based contrast adjustment. Using the Boost Converter, the glass bias may be at a higher voltage than the PSoC supply voltage. This allows increased display flexibility in portable applications.

Each LCD pixel/symbol may be either on or off. The Segment LCD component also provides advanced support to simplify the following types of display structures within the glass:

- 7-segment numerals
- 14-segment alphanumeric
- 16-segment alphanumeric
- 5x7 and 5x8 dot matrix alphanumeric (Use the same look-up table on the 5x7 and 5x8. All symbols in the look-up table are the size of 5x7 pixels.)

- 1- to 255-element bar graphs

For more information about using the Segment LCD component, refer to the application note *AN52927: PSoC® 3: Segment LCD Direct Drive Basics*.

When to Use a Segment LCD

Use the Direct Segment Drive LCD component when you need to directly drive a variety of LCD glass at different voltage levels with multiplex ratios up to 16x. The Direct Segment Drive LCD component requires that the target PSoC device support LCD direct drive.

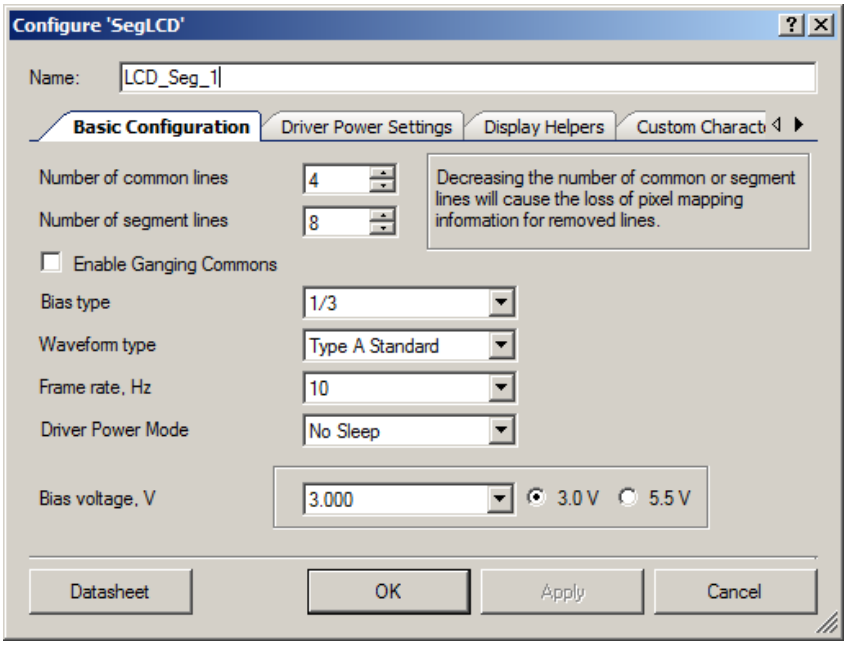
Input/Output Connections

There are no visible connections for the component on the schematic canvas; however, the various signals can be connected to pins using the Design-Wide Resources Pin Editor.

Parameters and Setup

Drag a Segment LCD component onto your design and double click it to open the **Configure** dialog. The **Configure** dialog contains several tabs with different types of parameters to set up the Segment LCD component.

Basic Configuration Tab



Number of common lines

Defines the number of common signals required by the display (default is 4).

Number of segment lines

Defines the number of segment signals required by the display. The range of possible values is from 2 to 62. The default is 8.

Enable Ganging Commons

Select this checkbox to gang PSoC pins to drive common signals. Two PSoC pins are allocated for each common signal. This is used to drive larger displays.

Bias type

This value determines the proper bias mode for the set of common and segment lines.

Waveform type

This determines the waveform type: Type A - 0 VDC average over a single frame (default) or Type B - 0 VDC average over two frames.

Frame rate, Hz

This determines the refresh rate of the display. In No Sleep mode, the range of possible values is selectable from 10 Hz to 150 Hz, in increments of 10. The default is 60 Hz.

In low-power modes, the frame rate selection is limited and is unique for every configuration. For a detailed description, refer to Driver Power Modes in the Functional Description section later in this datasheet.

Driver Power Mode

The Driver Power Mode parameter defines the power mode of the component. The following power mode settings are available:

- **No Sleep**: LCD DAC is always turned on and the chip will not enter a sleep mode
- **Low Power using ILO**: LCD DAC is turned on but the chip will enter sleep between voltage transactions. As a wakeup source, the component will use 1-kHz internal ILO.
- **Low Power using Ext 32kHz crystal**: LCD DAC is turned on but the chip will enter sleep between voltage transactions. As a wakeup source the component will use 8-K tap from OPPS timer.

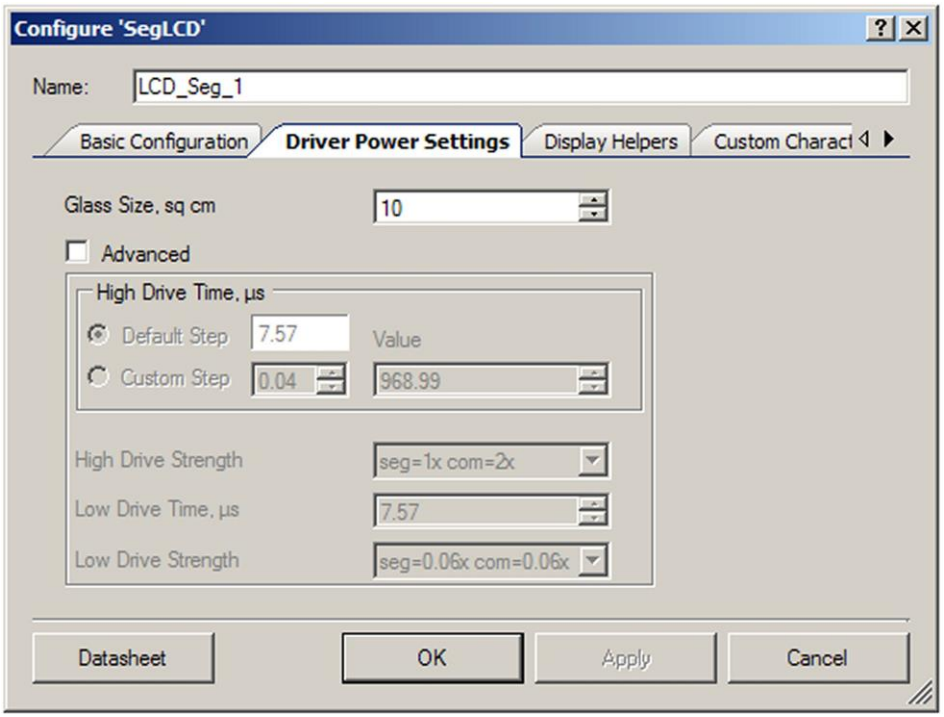
Note Depending on the low-power mode you are using, use either an ILO set to 1 kHz or an external 32-kHz crystal connected and enabled. You can enable the 32-kHz crystal or set the frequency of the ILO in the Design-Wide Resources Clock Editor.

Refer also to Driver Power Modes in the Functional Description section later in this datasheet.

Bias voltage, V

This determines the bias voltage level for the LCD DAC. The range of possible values is from 2.35 V to 5.5 V or from 2.017 V to 3 V, depending on the supply source.

Driver Power Settings Tab



Glass Size, sq cm

Use this field to enter the approximate size of the active area of the glass in square centimeters. This value will be used in conjunction with the **Number of common lines** parameter to calculate the approximate capacitive load.

Advanced

If this checkbox is checked, you are allowed to manually change the **Driver Power Settings** of the LCD. If it is unchecked, the driver power settings will be set automatically based on the selection from the **Basic Configuration** tab and the **Glass Size** parameter.

High Drive Time, µs

This parameter defines the time during which High Drive mode will be active within one voltage transaction.

Default Step

Defines the automatically calculated High Drive step for the selected configuration. If it is selected, this is the step with which the High Drive time value will be incremented/decremented.

Default High Drive step is the only possible selection for No Sleep mode.

Custom Step

Available only in low-power modes. Defines the user-selectable High Drive step. The minimum possible value of the custom High Drive step depends on the frequency of the master clock. The maximum possible value is limited by the value of **Default Step**.

Note If you change the **Frame rate**, **Number of common lines**, or **Waveform type** parameters, the **High Drive Time** parameter will be set automatically to half of the frame as that is its default value (only if the **Advanced** checkbox is checked):

*HighDriveTimeValdef = DefaultStep * 128* *(for both Type A and Type B waveforms)*

The calculation of **High Drive Time** minimum value is defined later in this datasheet. The maximum value of **High Drive Time** in the current configuration is:

*HighDriveTimeValmax = DefaultStep * 253* *(for both Type A and Type B waveforms)*

The value of the Default Step is calculated through the following equations:

*DefaultStep = 1 / (numCommons * FrameRate * 512 * 1.075)* *(Type A)*

*DefaultStep = 1 / (numCommons * FrameRate * 256 * 1.075)* *(Type B)*

The above equations are also applied to **High Drive Time** value with a **Custom Step**.

If you have selected one of the low-power modes, it is not recommended to set the **High Drive Time** to its maximum values if you are using the default High Drive step. The use of low-power modes will be ineffective because the device will enter sleep mode for only a short time, and in the worst case it may not enter sleep at all. With a Custom Step it is possible to adjust the **High Drive Time** value more accurately, which is a requirement of many low-power applications.

High Drive Strength

This parameter selects the drive strength for High Drive.

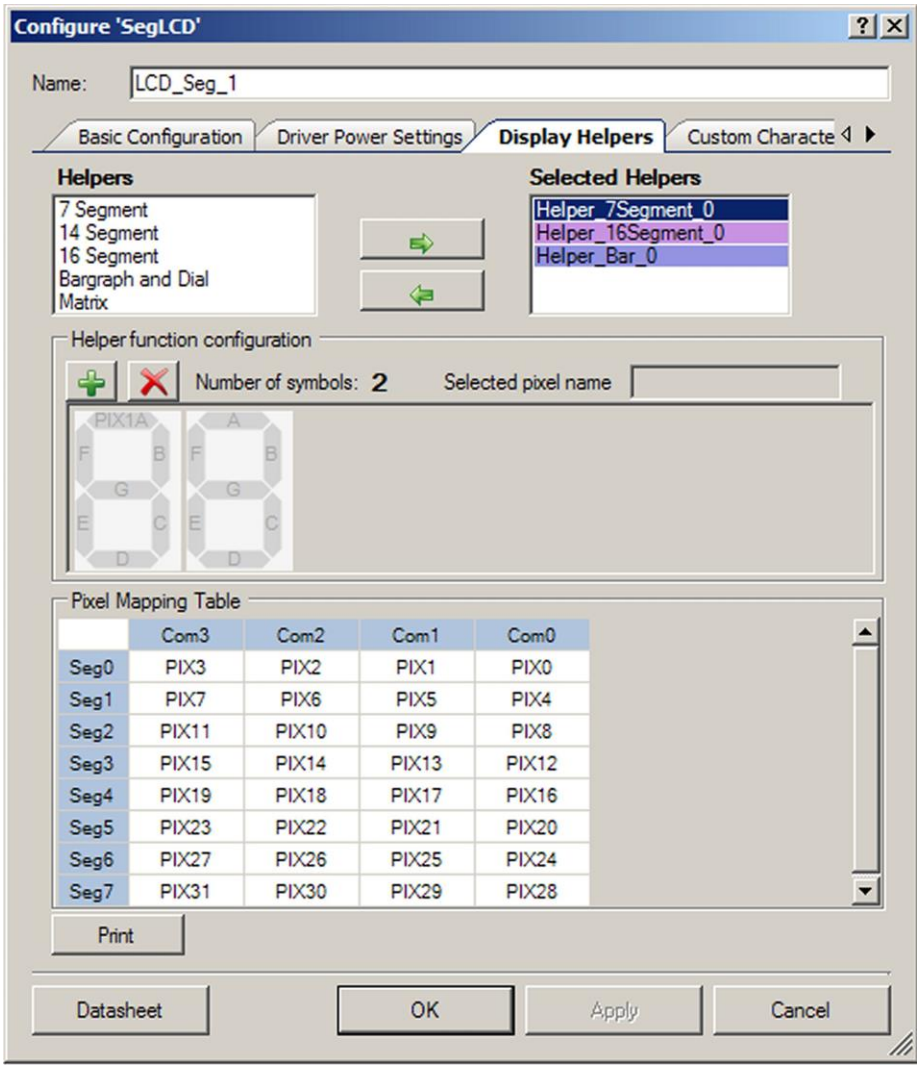
Low Drive Time, µs

The **Low Drive Time** parameter defines the time during which Low Drive Mode will be active within the voltage transaction.

Low Drive Strength

This parameter selects the drive strength for Low Drive. **Low Drive Strength** is relative to **High Drive Strength** and it sets automatically depending on the High Drive.

Display Helpers Tab



Display Helpers allow you to configure a group of display segments to be used together as one of several predefined display element types:

- 7-, 14-, or 16-segment displays
- Dot matrix display (5x7 or 5x8)
- Linear or circular bar graph display

The character-based display helpers can be used to combine multiple display symbols to create multi-character display elements.

Helpers/Selected Helpers

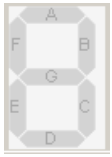
You may add one or more helpers to the **Selected Helpers** list by selecting the desired helper type in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to support the new helper, it will not be added. To delete a helper, select it in the **Selected Helpers** list and click the left-arrow button.

Note: It is important to set the number of common and segment lines for the component before defining any display helpers. Any defined display helpers must be removed before you change the number of common or segment lines, because you can lose helper configuration information. If you attempt to change the number of common or segment lines, a warning will display indicating that helper pixel mapping configuration can be lost.

The order in which the **Selected Helpers** appear in the list is significant. By default, the first helper of a given type added to the **Selected Helper** list is named with a 0 suffix, the next one of the same type will have a suffix of 1, and so on. If a **Selected Helper** is removed from the list, the remaining helpers will not be renamed. When a helper is added, the name will use the lowest available suffix.

APIs are provided for each helper. Refer to the Application Programming Interface section for more information.

- **7 Segment Helper –** This helper may be one to five digits in length and can display either hexadecimal digits 0 to F or decimal 16-bit unsigned integer (uint16) values. A decimal point is not supported by the helper functions.



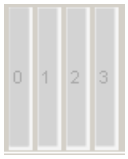
- **14 Segment Helper –** This helper may be up to 20 characters in length. It may display a single ASCII character or a null terminated string. Possible values are standard ASCII printable characters (with codes from 0 to 127).



- **16 Segment Helper –** This helper may be up to 20 characters in length. It may display a single ASCII character or a complete null terminated string. Possible values are standard ASCII characters and table of extended codes (with codes from 0 to 255). A table of extended codes is not supplied.



- **Bar Graph and Dial Helper –** These helpers are used for bar graphs and dial indicators with 1 to 255 segments. The bar graph may be a single selected pixel or the selected pixel and all of the pixels to the left or right of the specified pixel



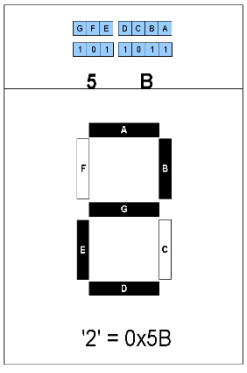
- **Matrix Helper –** This helper supports up to eight character elements. The component supports x5x7 or x5x8 row/column characters. Longer strings of characters can be created by configuring two or more dot-matrix helpers to define adjacent dot-matrix sections of the display. The helper displays a single ASCII character or a null terminated string.

The dot matrix helper has pin-out constraints. The dot matrix helper must use 7 or eight sequential common drivers for the matrix rows and 5 to 40 sequential segment drivers for the matrix columns. The component supports the standard Hitachi HD44780 character set.

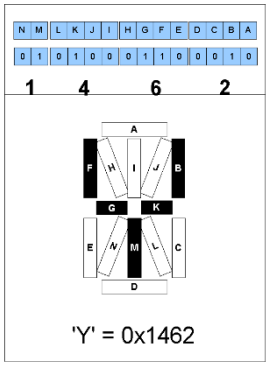
Character Encoding

All high-level Helper APIs have their own look-up tables. The tables include a set of encoded pixel states, which construct a specific character reflection. The following examples show how the specific character can be encoded (segment names may be different than shown in the **Configure** dialog).

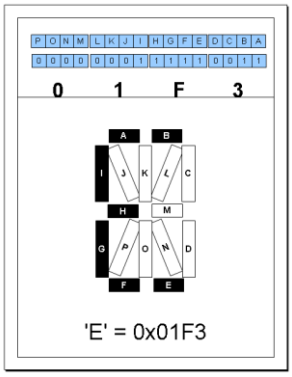
7-Segment Encoding



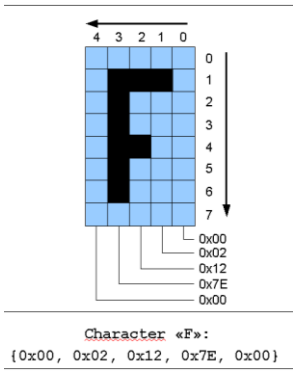
14-Segment Encoding



16-Segment Encoding



Dot-Matrix Encoding



Helper function configuration

This section of the dialog allows you to configure a helper; this includes adding or removing symbols to or from a helper as well as naming the pixels.

1. Select a helper from the **Selected Helpers** list.
2. Click the **[+]** or **[x]** button to add or remove a symbol for the selected helper.

   The maximum number of symbols you may add depends on the helper type and the total number of pixels supported by the component. If the number of available pins is not sufficient to support a new symbol, it will not be added.
3. To rename a pixel which is a part of a helper function, select the pixel on the symbol image in the **Helper function configuration** display. The current name will display in the selected pixel name field and can be modified as desired.

Pixel Naming

The default pixel names have the form “PIX#,” where “#” is the number of the pixel in incremental order starting from right upper corner of the **Pixel Mapping Table**.

The default naming for pixels associated with a helper symbol has a different format. The default name consists of a prefix portion, common to all of the pixels in a symbol, and a unique segment identifier. The default prefix indicates the helper type and the symbol instance. For example, the default name of a pixel in one of the symbols in a 7-segment display helper might be “H7SEG4_A” where:

H7	indicates the pixel is part of a 7-segment helper
SEG4	indicates the pixel is part of the symbol designated as the fourth 7-segment symbol in the project
A	identifies the unique segment within the 7 segment symbol

For default pixel names, only the unique portion of the pixel name is shown on the symbol image. If you modify a pixel name, then the entire name will be shown on the symbol image even if they have a common prefix.

Note All pixel names must be unique.

When a helper function symbol element is assigned to a pixel in the **Pixel Mapping Table** (described below), the pixel assumes the name of the helper symbol element. The helper symbol element name supersedes the default pixel name, but does not replace it. You cannot reuse the default pixel name of pixels that are associated with a helper function.

Pixel Mapping Table

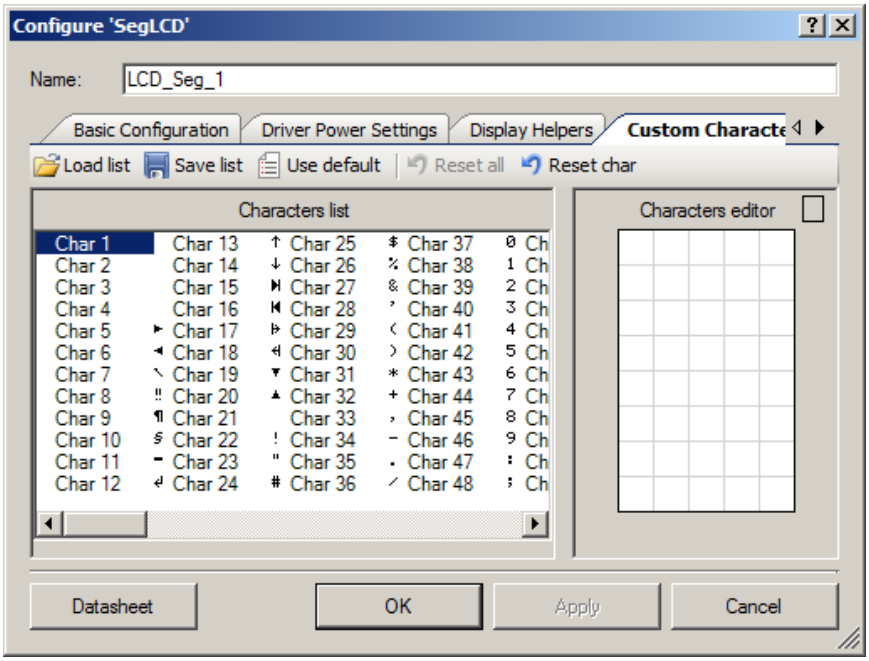
The **Pixel Mapping Table** is a representation of the frame buffer. For the API functions to work properly, each pixel from the **Helper function configuration** must be assigned to a pixel location in the **Pixel Mapping Table**. Refer to the datasheet for your LCD glass for the information you will need to make the correct assignments.

To assign pixels, select the desired pixel in the **Helper function configuration** panel and drag it to the correct location in the **Pixel Mapping Table**.

You can rename a pixel in the **Pixel Mapping Table** by double clicking on the pixel in the table display and entering the desired name. This method can be used to name a pixel that is not associated with one of the available helper types.

The **Print** button prints the **Pixel Mapping Table**.

Custom Characters Tab



This tab allows you to create custom characters for 5x8 dot-matrix displays. You may also use it to store a custom character look-up table as an XML string.

By default, the **Characters list** field contains 255 ASCII characters that have reflection as the standard Hitachi HD44780 character set. You can access and modify any of those characters using the **Character editor**.

The **Reset char** option allows you to reset unsaved characters to a default reflection. The **Reset all** option will bring all unsaved characters to the standard reflection.

After you have saved your own character set, you can save it as an XML string using the **Save list** command. The **Load list** command allows you to load your character list from an XML string. You can go back to a standard character set using the **Use default** option.

Clock Selection

The LCD_Seg component uses an internal clock and does not require an external clock. Once the component is placed, the clock is automatically dedicated to the LCD component. Its frequency is calculated automatically depending on the number of common lines, refresh rate, and waveform type.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections covers each function in more detail.

By default, PSoC Creator assigns the instance name “LCD_Seg_1” to the first instance of a component in a given design. You can rename the instance to any unique value that follows the PSoC Creator syntax rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is “LCD_Seg.”

General APIs

Function	Description
LCD_Seg_Start()	Sets the initVar variable, calls the LCD_Seg_Init() function, and then calls the LCD_Seg_Enable() function.
LCD_Seg_Stop()	Disables the LCD component and associated interrupts and DMA channels.
LCD_Seg_EnableInt()	Enables the LCD interrupts. Not required if LCD_Seg_Start() is called
LCD_Seg_DisableInt()	Disables the LCD interrupt. Not required if LCD_Seg_Stop() is called
LCD_Seg_SetBias()	Sets the bias level for the LCD glass to one of up to 64 values.
LCD_Seg_WriteInvertState()	Inverts the display based on an input parameter.
LCD_Seg_ReadInvertState()	Returns the current value of the display invert state: normal or inverted
LCD_Seg_ClearDisplay()	Clears the display and associated frame buffer RAM.
LCD_Seg_WritePixel()	Sets or clears a pixel based on PixelState
LCD_Seg_ReadPixel()	Reads the state of a pixel in the frame buffer.
LCD_Seg_Sleep()	Stops the LCD and saves the user configuration.
LCD_Seg_Wakeup()	Restores and enables the user configuration.
LCD_Seg_SaveConfig()	Saves the LCD configuration.
LCD_Seg_RestoreConfig()	Restores the LCD configuration.
LCD_Seg_Init()	Initializes or restores the LCD per the Configure dialog settings.
LCD_Seg_Enable()	Enables the LCD.

uint8 LCD_Seg_Start(void)

Description: Starts the LCD component and enables required interrupts, DMA channels, frame buffer, and hardware. Does not clear the frame buffer RAM.

Parameters: None

Return Value: uint8 cystatus: Standard API return values.

Return Value	Description
CYRET_LOCKED	Some of DMA TDs or a channel already in use.
CYRET_SUCCESS	Function completed successfully

Side Effects: This API enables one pulse-per-second bit from the Timewheel Configuration Register 2. This only valid for Low Power 32kHz External Xtal component mode.

void LCD_Seg_Stop(void)

Description: Disables the LCD component and associated interrupts and DMA channels. Automatically blanks the display to avoid damage from DC offsets. Does not clear the frame buffer.

Parameters: None

Return Value: None

Side Effects: This API doesn't clear one pulse-per-second bit from the Timewheel Configuration Register 2 previously enabled by LCD_Seg_Init() API. This only valid for Low Power 32kHz External Xtal component mode.

void LCD_Seg_EnableInt(void)

Description: Enables the LCD interrupts. An interrupt occurs after every LCD update (TD completion).

If the PSoC 5LP device is used, this API also enables an LCD wakeup Interrupt. This function should always be called when the component's operation in sleep is desired.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_DisableInt(void)

Description: Disables "every subframe" and LCD wakeup interrupts.

Parameters: None

Return Value: None

Side Effects: None

uint8 LCD_Seg_SetBias(uint8 biasLevel)

Description: This function sets the bias level for the LCD glass to one of up to 64 values. The actual number of values is limited by the Analog supply voltage, Vdda. The bias voltage cannot exceed Vdda. Changing the bias level affects the LCD contrast.

Parameters: uint8 biasLevel: bias level for the display

Return Value: uint8 cystatus: Standard API return values.

Return Value	Description
CYRET_BAD_PARAM	Evaluation of biasLevel parameter is failed.
CYRET_SUCCESS	Function completed successfully.

Side Effects: None

uint8 LCD_Seg_WriteInvertState(uint8 invertState)

Description: This function inverts the display based on an input parameter. The inversion occurs in hardware and no change is required to the display RAM in the frame buffer.

Parameters: uint8 invertState: Sets the invert state of the display.

Return Value: uint8 cystatus: Standard API return values.

Side Effects: None

uint8 LCD_Seg_ReadInvertState(void)

Description: This function returns the current value of the display invert state: normal or inverted.

Parameters: None

Return Value: (uint8) invertState: The invert state of the display.

Side Effects: None

void LCD_Seg_ClearDisplay(void)

Description: This function clears the display and the associated frame buffer RAM.

Parameters: None

Return Value: None

Side Effects: None

uint8 LCD_Seg_WritePixel(uint16 pixelNumber, uint8 pixelState)

Description: This function sets or clears a pixel based on the input parameter PixelState. The pixel is addressed by a packed number.

Parameters: uint16 pixelNumber: The packed number that points to the pixel's location in the frame buffer. The lowest three bits in the LSB low nibble are the bit position in the byte, the LSB upper nibble (four bits) is the byte address in the multiplex row and the MSB low nibble (four bits) is the multiplex row number. The generated component .h file includes the #defines of this format for each pixel.

uint8 pixelState: The pixelNumber specified is set to this pixel state.

Return Value: uint8 status: Pass or fail based on a range check of the byte address and multiplex row number. No check is performed on bit position.

Side Effects: None

uint8 LCD_Seg_ReadPixel(uint16 pixelNumber)

Description: This function reads the state of a pixel in the frame buffer. The pixel is addressed by a packed number.

Parameters: uint16: pixelNumber: is the packed number that points to the pixel's location in the frame buffer. The lowest three bits in the LSB low nibble are the bit position in the byte, the LSB upper nibble (four bits) is the byte address in the multiplex row and the MSB low nibble (four bits) is the multiplex row number. The generated component .h file includes the #defines of this format for each pixel.

Return Value: uint8 pixelState: Returns the current status of the PixelNumber specified.

Value	Description
0x00	The pixel is off.
0x01	The pixel is on.
0xFF	The pixel is not assigned.

Side Effects: None

void LCD_Seg_Sleep(void)

Description: This is the preferred routine to prepare the component for sleep. The LCD_Seg_Sleep() routine saves the current component state. Then it calls the LCD_Seg_Stop() function and calls LCD_Seg_SaveConfig() to save the hardware configuration.

Call the LCD_Seg_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the PSoC Creator *System Reference Guide* for more information about power management functions.

Parameters: None

Return Value: None

Side Effects: Doesn't change component pin's drive modes.

void LCD_Seg_Wakeup(void)

Description: This is the preferred routine to restore the component to the state when LCD_Seg_Sleep() was called. The LCD_Seg_Wakeup() function calls the LCD_Seg_RestoreConfig() function to restore the configuration. If the component was enabled before the LCD_Seg_Sleep() function was called, the LCD_Seg_Wakeup() function will also re-enable the component.

Parameters: None

Return Value: uint8 cystatus: Standard API return values

Return Value	Description
CYRET_LOCKED	Some of DMA TDs or a channel already in use
CYRET_SUCCESS	Function completed successfully

Side Effects: Calling the LCD_Seg_Wakeup() function without first calling the LCD_Seg_Sleep() or LCD_Seg_SaveConfig() function may produce unexpected behavior.

void LCD_Seg_SaveConfig(void)

Description: This function saves the component configuration. This will save non-retention registers. This function will also save the current component parameter values, as defined in the Configure dialog or as modified by appropriate APIs. This function is called by the LCD_Seg_Sleep() function.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_RestoreConfig(void)

Description: This function restores the component configuration. This will restore non-retention registers. This function will also restore the component parameter values to what they were prior to calling the LCD_Seg_Sleep() function.

Parameters: None

Return Value: None

Side Effects: Calling this function without first calling the LCD_Seg_Sleep() or LCD_Seg_SaveConfig() function may produce unexpected behavior.

void LCD_Seg_Init(void)

Description: Initializes or restores the component parameters per the Configure dialog settings. It is not necessary to call LCD_Seg_Init() because the LCD_Seg_Start() routine calls this function and is the preferred method to begin component operation. Configures and enables all required hardware blocks, and clears the frame buffer.

Parameters: None

Return Value: None

Side Effects: All registers will be set to values per the Configure dialog. This API enables one pulse-per-second bit from the Timewheel Configuration Register 2. This only valid for Low Power 32 -kHz External Xtal component mode.

void LCD_Seg_Enable(void)

Description: Enables power to the LCD fixed hardware and enables generation of UDB signals.

Parameters: None

Return Value: uint8 cystatus: Standard API return values

Return Value	Description
CYRET_LOCKED	Some of DMA TDs or a channel already in use.
CYRET_SUCCESS	Function completed successfully

Side Effects: None

Note Because of specific component implementation, you cannot use the component with the CyPmSaveClocks() and CyPmRestoreClocks() APIs when it is in low-power mode. In other cases, you can use the CyPmSaveClocks() and CyPmRestoreClocks() APIs.

Optional Helper APIs

The following APIs are present only when the respective helper has been selected in the Configure dialog.

Function	Description
LCD_Seg_Write7SegDigit_n	Displays a hexadecimal digit on an array of 7-segment display elements.
LCD_Seg_Write7SegNumber_n	Displays an integer value on a 1- to 5-digit array of 7-segment display elements.
LCD_Seg_WriteBargraph_n	Displays an integer location on a linear or circular bar graph.
LCD_Seg_PutChar14Seg_n	Displays a character on an array of 14-segment alphanumeric character display elements.
LCD_Seg_WriteString14Seg_n	Displays a null terminated character string on an array of 14-segment alphanumeric character display elements.
LCD_Seg_PutChar16Seg_n	Displays a character on an array of 16-segment alphanumeric character display elements.
LCD_Seg_WriteString16Seg_n	Displays a null terminated character string on an array of 16-segment alphanumeric character display elements.
LCD_Seg_PutCharDotMatrix_n	Displays a character on an array of dot-matrix alphanumeric character display elements.
LCD_Seg_WriteStringDotMatrix_n	Displays a null terminated character string on an array of dot-matrix alphanumeric character display elements.

Note Function names that contain a suffix “n” indicate that multiple display helpers of the same symbol type were created in the component customizer. Specific display helper elements are controlled by the API functions with the respective “n” suffix in the function name.

void LCD_Seg_Write7SegDigit_n(uint8 digit, uint8 position)

Description: This function displays a hexadecimal digit on an array of 7-segment display elements. Digits can be hexadecimal values in the range of 0 to 9 and A to F. The customizer Display Helpers facility must be used to define the pixel set associated with the 7-segment display elements Multiple 7-segment display elements can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a 7-segment display element is defined in the component customizer.

Parameters: uint8 digit: Unsigned integer value in the range of 0 to 15 to be displayed as a hexadecimal digit.

uint8 position: Position of the digit as counted right to left starting at 0 on the right. If the position is outside the defined display area, the character will not be displayed.

Return Value: None

Side Effects: None

void LCD_Seg Write7SegNumber_n(uint16 value, uint8 position, uint8 mode)

Description: This function displays a 16-bit integer value on a 1- to 5-digit array of 7-segment display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the 7-segment display element(s). Multiple 7-segment display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. Sign conversion, sign display, decimal points, and other custom features must be handled by application-specific user code. This function is only included if a 7-segment display element is defined in the component customizer.

Parameters: uint16 value: The unsigned integer value to be displayed.

uint8 position: The position of the least significant digit as counted right to left starting at 0 on the right. If the defined display area contains fewer digits then the value requires, the most significant digit or digits will not be displayed.

uint8 mode: Sets the display mode. Can be zero or one.

Value	Description
0	No leading 0s are displayed.
1	Leading 0s are displayed

Return Value: None

Side Effects: None

void LCD_Seg_WriteBargraph_n(uint8 location, uint8 mode)

Description: This function displays an 8-bit integer Location on a 1- to 255-segment bar graph (numbered left to right).The bar graph may be any user-defined size between 1 and 255 segments. A bar graph may also be created in a circle to display rotary position. The customizer Display Helpers facility must be used to define the pixel set associated with the bar graph display elements Multiple bar graph displays can be created in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a bar graph display element is defined in the component customizer

Parameters: uint8 location: The unsigned integer Location to be displayed. Valid values are from zero to the number of segments in the bar graph. A zero value turns all bar graph elements off. Values greater than the number of segments in the bar graph result in all elements on.

uint8 mode: Sets the bar graph display mode.

Value	Description
0	The specified Location segment is turned on.
1	The Location segment and all segments to the left are turned on.
–1	The Location segment and all segments to the right are turned on.
2 to 10	Display the Location segment and 2 to 10 segments to the right. This mode can be used to create wide indicators.

Return Value: None

Side Effects: None

void LCD_Seg_PutChar14Seg_n(uint8 character, uint8 position)

Description: This function displays an 8-bit character on an array of 14-segment alphanumeric character display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the 14-segment display element. Multiple 14-segment alphanumeric display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a 14-segment element is defined in the component customizer.

Parameters: uint8 character: The ASCII value of the character to display (printable characters with ASCII values 0 o 127)

uint8 position: The position of the character as counted left to right starting at 0 on the left. If the position is outside the defined display area, the character will not be displayed.

Return Value: None

Side Effects: None

void LCD_Seg_WriteString14Seg_n(uint8 const character[], uint8 position)

Description: This function displays a null terminated character string on an array of 14-segment alphanumeric character display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the 14 segment display elements. Multiple 14-segment alphanumeric display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a 14-segment display element is defined in the component customizer

Parameters: *uint8 character: The pointer to the null terminated character string.

uint8 position: The position of the first character as counted left to right starting at 0 on the left. If the length of the string exceeds the size of the defined display area, the extra characters will not be displayed.

Return Value: None

Side Effects: Doesn't clear display prior data output. All the locations that weren't affected will remain their previous pixel states.

void LCD_Seg_PutChar16Seg_n(uint8 character, uint8 position)

Description: This function displays an 8-bit character on an array of 16-segment alphanumeric character display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the 16-segment display element(s). Multiple 16-segment alphanumeric display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a 16-segment display element is defined in the component customizer

Parameters: uint8 character: The ASCII value of the character to display (printable ASCII and table extended characters with values 0 to 255).

uint8 position: The position of the character as counted left to right starting at 0 on the left. If the position is outside the defined display area, the character will not be displayed.

Return Value: None

Side Effects: None

void LCD_Seg_WriteString16Seg_n(uint8 const character[], uint8 position)

Description: This function displays a null terminated character string on an array of 16-segment alphanumeric character display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the 16 segment display elements. Multiple 16-segment alphanumeric display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a 16-segment display element is defined in the component customizer.

Parameters: *uint8 character: The pointer to the null terminated character string.

uint8 position: The position of the first character as counted left to right starting at 0 on the left. If the length of the string exceeds the size of the defined display area, the extra characters will not be displayed.

Return Value: None

Side Effects: Doesn't clear display prior data output. All the locations that weren't affected will remain their previous pixel states.

void LCD_Seg_PutCharDotMatrix_n(uint8 character, uint8 position)

Description: This function displays an 8-bit character on an array of dot-matrix alphanumeric character display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the dot matrix display elements. Multiple dot-matrix alphanumeric display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a dot-matrix display element is defined in the component customizer

Parameters: uint8 character: The ASCII value of the character to display.

uint8 position: The position of the character as counted left to right starting at 0 on the left. If the position is outside the defined display area, the character will not be displayed.

Return Value: None

Side Effects: None

void LCD_Seg_WriteStringDotMatrix_n(uint8 const character[], uint8 position)

Description: This function displays a null terminated character string on an array of dot-matrix alphanumeric character display elements. The customizer Display Helpers facility must be used to define the pixel set associated with the dot-matrix display elements. Multiple dot-matrix alphanumeric display element groups can be defined in the frame buffer and are addressed through the suffix (n) in the function name. This function is only included if a dot- matrix display element is defined in the component customizer

Parameters: *uint8 character: The pointer to the null terminated character string.

uint8 position: The position of the first character as counted left to right starting at 0 on the left. If the length of the string exceeds the size of the defined display area, the extra characters will not be displayed.

Return Value: None

Side Effects: Doesn't clear display prior data output. All the locations that weren't affected will remain their previous pixel states.

Pins APIs

These API functions are used to change drive mode of pins used by Segment LCD component.

Function	Description
LCD_Seg_ComPort_SetDriveMode	Sets the drive mode for all pins used by common lines of the Segment LCD component
LCD_Seg_SegPort_SetDriveMode	Sets the drive mode for all pins used by segment lines of the Segment LCD component.

void LCD_Seg_ComPort_SetDriveMode(uint8 mode)

Description: Sets the drive mode for all pins used by common lines of the Segment LCD component.

Parameters: uint8 mode: The desired drive mode. See the Pins component datasheet for information on drive modes.

Return Value: None

Side Effects: None

LCD_Seg_SegPort_SetDriveMode(uint8 mode)

Description: Sets the drive mode for all pins used by segment lines of the Segment LCD component.

Parameters: uint8 mode: The desired drive mode. See the Pins component datasheet for information on drive modes.

Return Value: None

Side Effects: None

Global Variables

Variable	Description
LCD_Seg_initVar	LCD_Seg_InitVar indicates whether the Segment LCD has been initialized. The variable is initialized to 0 and set to 1 the first time LCD_Seg_Start() is called. This allows the component to restart without reinitialization after the first call to the LCD_Seg_Start() routine. If reinitialization of the component is required, then the LCD_Seg_Init() function can be called before the LCD_Seg_Start() or LCD_Seg_Enable() function.

Macros

- LCD_Seg_COMM_NUM – Defines the number of common lines in the user-defined display for the current configuration of the component.
- LCD_Seg_SEG_NUM – Defines the number of segment lines for the user-defined display for the current configuration of the component.
- LCD_Seg_BIAS_TYPE – Defines the bias type for the user-defined display current for the configuration of the component.
- LCD_Seg_BIAS_VOLTAGE – Defines default bias voltage level for the user-defined display. This value will be set in the LCDDAC control register during the initialization process.
- LCD_Seg_FRAME_RATE – Defines the refresh rate for the user-defined display for the current configuration of the component.
- LCD_Seg_EXTRACT_ROW – Calculates the row of the specific pixel in the frame buffer.
- LCD_Seg_EXTRACT_PORT – Calculates the byte offset of the specific pixel in the frame buffer.
- LCD_Seg_EXTRACT_PIN – Calculates the bit position of the specific pixel in the frame buffer.
- LCD_Seg_WRITE_PIXEL – This is a macro define of the LCD_Seg_WritePixel() function with void type.
- LCD_Seg_READ_PIXEL – This is a macro define of the LCD_Seg_ReadPixel() function.
- LCD_Seg_FIND_PIXEL – This macro calculates pixel location in the frame buffer. It uses information from the customizer pixel table and information about the physical pins that will be dedicated for the LCD. This macro is the base of the pixel mapping mechanism. Every pixel name from the pixel table is defined with a calculated pixel location in the frame buffer. APIs use pixel names to access the respective pixel.

Macro Callbacks

Macro callbacks allow users to execute code from the API files that are automatically generated by PSoC Creator. Refer to the PSoC Creator Help and *Component Author Guide* for the more details.

In order to add code to the macro callback present in the component's generated source files, perform the following:

- Define a macro to signal the presence of a callback (in *cyapicallbacks.h*). This will "uncomment" the function call from the component's source code.
- Write the function declaration (in *cyapicallbacks.h*). This will make this function visible by all the project files.
- Write the function implementation (in any user file).

Callback Function [1]	Associated Macro	Description
LCD_Seg_ISR_InterruptCallback	LCD_Seg_ISR_INTERRUPT_CALLBACK	Used in the LCD_Seg_ISR() interrupt handler to perform additional application-specific actions.
LCD_Seg_WAKEUP_ISR_InterruptCallback	LCD_Seg_WAKEUP_ISR_INTERRUPT_CALLBACK	Used in the LCD_Seg_WAKEUP_ISR() interrupt handler to perform additional application-specific actions.
LCD_Seg_SaveConfig_Callback	LCD_Seg_SAVE_CONFIG_CALLBACK	Used in the LCD_Seg_SaveConfig() function to perform additional application-specific actions.
LCD_Seg_RestoreConfig_Callback	LCD_Seg_RESTORE_CONFIG_CALLBACK	Used in the LCD_Seg_RestoreConfig() function to perform additional application-specific actions.

Sample Firmware Source Code

PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

1 The callback function name is formed by component function name optionally appended by short explanation and "Callback" suffix.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The Segment LCD component has the following specific deviations:

MISRA-C: 2004 Rule	Rule Class (Required/ Advisory)	Rule Description	Description of Deviation(s)
8.7	R	Objects shall be defined at block scope if they are only accessed from within a single function.	The following objects are defined as static const in C file scope: SegLCD_charDotMatrix[], SegLCD_7SegDigits[], SegLCD_16SegChars[], SegLCD_14SegChars[]. These arrays hold look-up tables for different helpers. Each helper uses the look-up table for its type. Depending on the configuration a helper can have multiple instances that will share access to its look-up table.
10.1	R	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: a) it is not a conversion to a wider integer type of the same signedness, or b) the expression is complex, or c) the expression is not constant and is a function argument, or d) the expression is not constant and is a return expression.	The DMA component provides general integer type definitions.
13.2	A	Tests of a value against zero should be made explicit, unless the operand is effectively Boolean.	The DMA component provides general integer type definitions which are ORed together to provide the correct function argument.
13.7	R	Boolean operations whose results are invariant shall not be permitted.	Applies for LCD_Seg_WriteBargraph_n() when the number of pixels used in this bargraph helper is less than 2. In this

MISRA-C: 2004 Rule	Rule Class (Required/ Advisory)	Rule Description	Description of Deviation(s)
			case the implementation of this function uses an input parameter checking mechanism which leads to invariant boolean operations. This results in unreachable code.
14.1	R	There shall be no unreachable code. This refers to code which cannot, under any circumstances, be reached.	Applies for LCD_Seg_WriteBargraph_n() when the number of pixels used in this bargraph helper is less than 2. In this case the implementation of this function uses an input parameter checking mechanism which leads to invariant boolean operations. This results in unreachable code.
19.7	A	A function should be used in preference to a function-like macro.	The following macros are used for increased performance: SegLCD_FIND_PIXEL(), SegLCD_EXTRACT_ROW(), SegLCD_EXTRACT_PORT(), SegLCD_EXTRACT_PIN(). The macro SegLCD_WRITE_PIXEL() is mapped to the function SegLCD_WritePixel() with the return value cast to void. This macro is used internal to the component where the return value is not used. The macro SegLCD_READ_PIXEL() is mapped directly to the function SegLCD_ReadPixel().
21.1	R	Minimization of run-time failures shall be ensured by the use of at least one of the following: a) Static analysis tools/ techniques b) Dynamic analysis tools/ techniques c) Explicit coding of checks to handle run-time faults	The functions LCD_Seg_Write7SegDigit_n() and LCD_Seg_Write7SegNumber_n() have redundant code when generated for single digit displays. The function LCD_Seg_WriteBargraph_n() has redundant code when generated for less than 3 pixels.

This component has the following embedded components: DMA and Interrupt. Refer to the corresponding component datasheets for information on their MISRA compliance and specific deviations.

Functional Description

The Segment LCD component provides a powerful and flexible mechanism for driving different types of LCD glass. The configuration dialog provides access to the parameters that can be used to customize the component functionality. A standard set of API routines provides control of the display and of specific pixels. Additional display APIs are generated based on the type and number of Display Helpers defined.

Default Configuration

The default configuration of the LCD_Seg component provides a generic LCD Direct Segment drive controller. The default LCD_Seg configuration is:

- Four common lines
- Eight segment lines
- 1/3 bias type
- 60-Hz refresh rate
- No Sleep power mode
- 3-V bias voltage
- Glass Size: 10 sq cm
- 968.99-µs High Drive time
- seg - 1x, com - 2x High Drive strength
- 7.57-µs Low Drive time
- seg – 0.06x, com 0.12x Low Drive strength
- No display helpers are defined. Default API generation does not include functions for any of the supported display elements.

Custom Configuration

A key feature of the Segment LCD component is flexible support for LCDs with different characteristics and layouts.

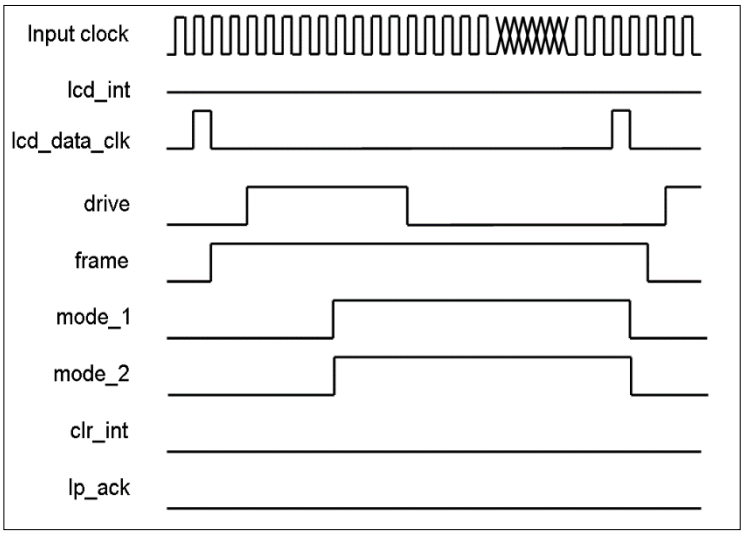
Driver Power Modes

Segment LCD can operate in two power modes: No Sleep and Low Power. No Sleep is the default operation mode.

No Sleep Mode

In this mode, the Segment LCD will never go into a sleep mode, and the LCD is driven throughout the entire frame. Figure 1 shows waveforms for UDB-generated (internal) signals of the LCD_Seg component for No Sleep mode.

Figure 1. Segment LCD Control Signals (No Sleep Mode)



Low Power Mode

In this mode, the LCD is actively driven only at voltage transitions, and the LCD system analog components are entering sleep mode between voltage transitions. The internal LCD timer will wake up the device from sleep mode for a short period of time to update the LCD screen. After that, the internal logic puts the entire device back into Sleep mode until the next refresh sequence.

In low-power mode, the Frame Rate parameter is limited and depends on other component parameters, such as clock source for the LCD timer (either 1 kHz [ILO] or 8 kHz [32-kHz crystal]), number of commons, and waveform type. The Frame Rate limitation can be explained with the limited input frequency of the LCD timer. The following is an example calculation of maximum frame rate for the configuration of eight commons, low Power ILO mode, Type A waveforms:

max FR = 1000/(2 × Num Commons) = 62.5 Hz

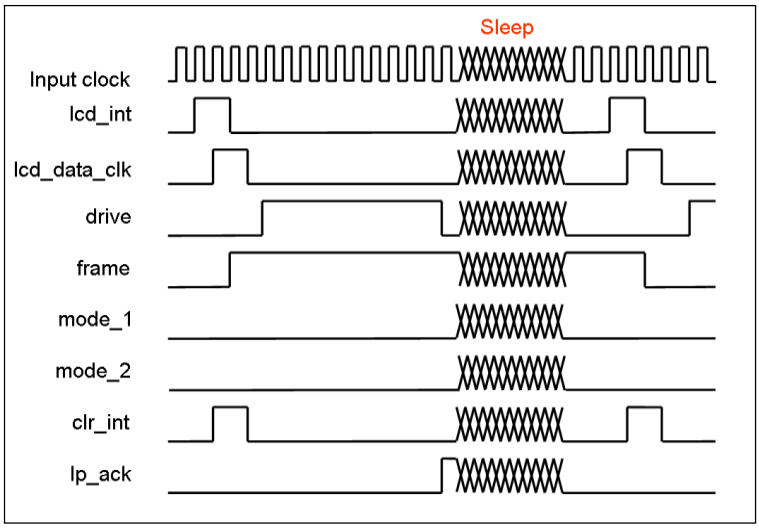
62.5 Hz is the maximum frame rate that the LCD timer can provide in which it generates a wakeup signal on every clock cycle of input frequency.

In the real world, the LCD timer needs one cycle to set the wakeup event and one cycle to clear it. This means the maximum frame rate should be divided by two, which results in 31.25 Hz. From this value, take only the integer which gives 31 Hz as the actual maximum frame rate.

For the Type B waveforms, the actual frame rate is two times larger than for Type A waveforms, because Type B waveforms require two times fewer wakeup events.

Figure 2 shows waveforms for UDB-generated signals of the LCD_Seg component for low-power mode.

Figure 2. Segment LCD Control Signals (Low Power mode)

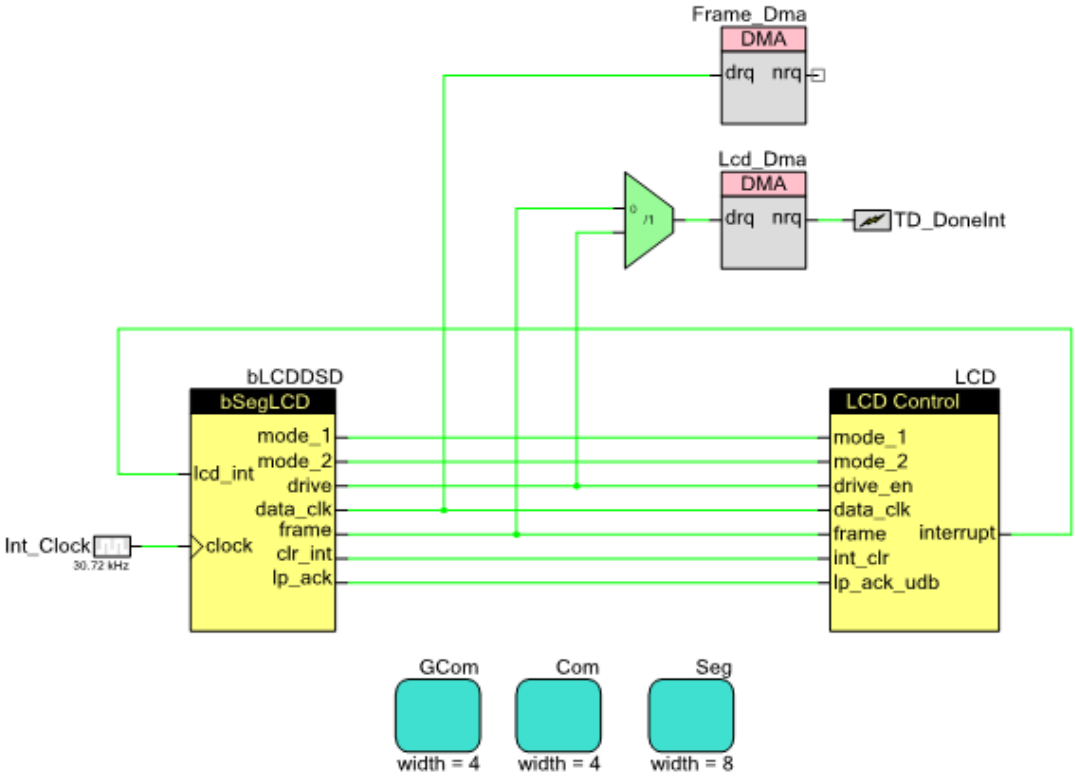


Note When you using the PSoC 5LP device, it is required to have an interrupt with an ISR (even if it is empty) to wake up the CPU from sleep. Therefore, the LCD_Seg_EnableInt() function should always be called prior putting the device into sleep mode.

Block Diagram and Configuration

Figure 3 shows the internal schematic for the Segment LCD component. It consists of a basic Segment LCD component, LCD Control block (LCD) component, DMA component, three LCD ports, one digital port, an ISR component, and a clock component.

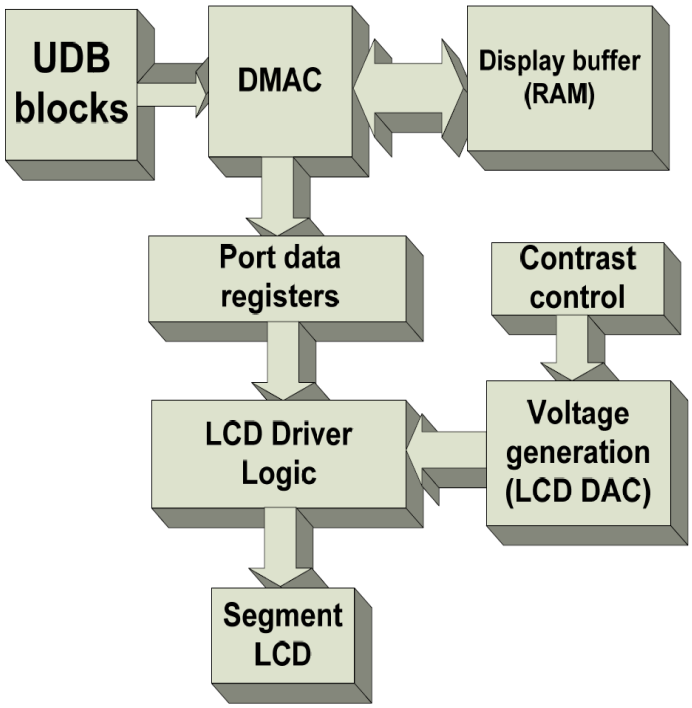
Figure 3. Segment LCD Component Schematic



- The basic Segment LCD component is responsible for generating the proper timing signals for the LCD Port and DMA components.
- The DMA component is used to transfer data from the frame buffer to the LCD data registers through the aliased memory area.
- The LCD component handles the required DSI routing. This block also provides the required register names as defined in *cyfitter.h*.
- The LCD ports (GCom, Com, and Seg) are used to map the logical signals to physical pins. There are two instances of the LCD Port: one for common lines and one for segment lines. The LCD port for the common signals is limited to 16 pins wide, and the LCD Port for segment signals is limited to 48 pins wide. There is also an additional common port, GCom, which is only included if the Ganging option is enabled.

Top Level Architecture

Figure 4. Segment LCD Top Level



Registers

LCD_Seg_CONTROL_REG

Bits	7	6	5	4	3	2	1	0
Value	frame	reserved			frame done	mode 2	mode 1	clock enable

- clock enable: Enables generation of all internal signals described in the previous sections.
- mode 1: The middle bit of mode[2:0] bit field, which defines High and Low drive strength.
- mode 2: The higher bit of mode[2:0] bit field, which defines High and Low drive strength.
- frame done: Generates a synchronous pulse after completion of a DMA Frame transaction.
- frame: Generates Frame signal for the LCD Driver.

LCD_Seg_CONTRAST_CONTROL_REG

Holds bias-voltage level, which is used by LCD DAC to generate proper bias voltage. An API is provided to change bias-voltage level.

Bits	7	6	5	4	3	2	1	0
Value	reserved	reserved	contrast level					

- contrast level: Bias-voltage level described above.

LCD_Seg_LCDDAC_CONTROL_REG

Bits	7	6	5	4	3	2	1	0
Value	lp enable	reserved			continuous drive	reserved	bias select	

- bias select: Selects bias.
- continuous drive: Allows the LCDDAC to remain active when the chip goes to sleep.
- lp enable: Allows the UDB to gate the Low Power Ack for the LCD subsystem.

LCD_Seg_TIMER_CONTROL_REG

Bits	7	6	5	4	3	2	1	0
Value	period						clk select	enable timer

- enable timer: Enables LCD Timer.
- clk select: LCD Timer source selection clock.
- period: LCD Timer period.

LCD_Seg_DRIVER_CONTROL_REG

Bits	7	6	5	4	3	2	1	0
Value	reserved			bypass enable	pts	invert	mode 0	sleep mode

- sleep mode: When in a low-power mode, sets the output buffer in LCD Drivers to ground if set to ‘1’, otherwise sets LCD Drivers to HI-Z.
- mode 0: Lower bit of mode[2:0] bit field, which defines High and Low drive strength.
- invert: If set, inverts data on the segment pins.
- pts: If cleared to ‘0’ - normal operation, VOUT = VIO-0.5V. If set to ‘1’, VOUT = VIO.

- bypass_enable: If set to ‘1’, bypasses the drive buffer within the LCD Driver and instead connects directly to the voltage input selected. If cleared to ‘0’, normal operation, the voltages are passed through driver buffer.

LCD_Seg_LCDDAC_SWITCH_REG[0..4]

Bits	7	6	5	4	3	2	1	0
Value	reserved						switch control[0..4]	

- switch control[0..4]: This set of bit-fields selects voltage sources for LCD Driver.

Resources

The Segment LCD component is a dedicated piece of hardware (LCD Fixed Block) in PSoC 3 and PSoC 5LP that is used in conjunction with UDBs to allow direct driving of external LCD glass. The component utilizes the following resources:

Configuration	Resource Type					
	Datapath Cells	Macrocells	Status Cells	Control Cells	DMA Channels	Interrupts
No Sleep Mode	1	5	–	1	2	1
Low Power Mode	1	8	–	1	2	2

The table below shows all possible configurations of the Static Segment LCD component. The meaning of the configuration names is as follows:

Basic: Low-level API functions set without any high-level helper API

Basic, 7-Segment helper: Low-level API functions set + 7-Segment helper API

Basic, 14-Segment helper: Low-level API functions set + 14-Segment helper API

Basic, 16-Segment helper: Low-level API functions set + 16-Segment helper API

Basic, Dot Matrix helper: Low-level API functions set + Dot Matrix helper high-level API.

Basic, Bar Graph helper: Low-level API functions set + Bar Graph helper high-level API.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Basic	2254	86	1560	50
Basic, 7-segment helper	2574	87	1698	90
Basic, 14-segment helper	2748	87	1928	90

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Basic, 16-segment helper	2752	87	1932	90
Basic, Dot Matrix helper	3881	106	3032	106
Basic, Bar Graph helper	2794	87	1632	90

DC and AC Electrical Characteristics

LCD Direct Drive DC Specifications001-88604.docx

Parameter	Description	Conditions	Min	Typ	Max	Units
I_{CC}	LCD system operating current	Device sleep mode with wakeup at 400-Hz rate to refresh LCDs, bus clock = 3 MHz, $V_{DDIO} = V_{DDA} = 3$ V, 4 commons, 16 segments, 1/4 duty cycle, 50-Hz frame rate, no glass connected	–	38	–	µA
I_{CC_SEG}	Current per segment driver	Strong drive mode	–	260	–	µA
V_{BIAS}	LCD bias range (V_{BIAS} refers to the main output voltage(V0) of LCD DAC)	$V_{DDA} \geq 3$ V and $V_{DDA} \geq V_{BIAS}$	2	–	5	V
	LCD bias step size	$V_{DDA} \geq 3$ V and $V_{DDA} \geq V_{BIAS}$	–	$9.1 \times V_{DDA}$	–	mV
	LCD capacitance per segment/common driver	Drivers may be combined	–	500	5000	pF
	Long-term segment offset		–	–	20	mV
I_{OUT}	Output drive current per segment driver	$V_{DDIO} = 5.5$ V, strong drive mode	355	–	710	µA

LCD Direct Drive AC Specifications001-88604.docx

Parameter	Description	Conditions	Min	Typ	Max	Units
f_{LCD}	LCD frame rate		10	50	150	Hz

Component Errata

This section lists known problems with the component.

Cypress ID	Component Version	Problem	Workaround
191257	v3.40	This component was modified without a version number change in PSoC Creator 3.0 SP1. For further information, see Knowledge Base Article KBA94159 (www.cypress.com/go/kba94159).	No workaround is necessary. There is no impact to designs.

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
3.40.e	Minor datasheets edits.	
3.40.d	Enabled component to be nested into other components.	Support for hierarchical component design.
3.40.c	Datasheet update.	Added Macro Callbacks section.
3.40.b	Edited the datasheet.	Added Component Errata section to document that the component was changed, but there is no impact to designs.
3.40.a	Updated the description of LCD_Seg_SetBias().	The description mentioned 128 bias level values but it only 63 values.
3.40	Updated the Custom Characters tab interface and the default character set to match SegLCD_P4 component.	To provide consistency between SegLCD and PSoC 4 SegLCD components.
	Made minor GUI improvements of the Basic Configuration and Driver Power Settings tabs.	Usability improvements.
3.30	Added MISRA Compliance section.	The component has specific deviations described.
	Updated Segment LCD with the latest version of the Clock, Interrupt and DMA components.	
	Fixed issue with LCD_Seg_WriteStringDotMatrix_n() and LCD_Seg_WriteString16Seg_n() API.	These functions may have display junk values sometimes. Due an incorrect condition check in their implementation. Instead of zero terminating character function looked for NULL pointer.
	Fixed issue with LCD_Seg_WriteBargraph_n() API.	This API didn't clear its previous output on display, which sometimes caused unexpected results displayed on the LCD.

Version	Description of Changes	Reason for Changes / Impact
	Fixed issue with LCD_Seg_Write7SegNumber_n() API.	When it was used in "no leading zeros" mode, this API didn't cleared remaining digits on the left hand side. Opposite to "leading zeros" mode this digits were filled with zeroes.
	Fixed description issues in LCD_Seg_SetBias().	Added description of return value of LCD_Seg_SetBias() as previously it was described as such that returns void. Also description of API mentioned 128 bias level values but it has only 63 values.
	Declaration of parameter “character” in LCD_Seg_WriteStringDotMatrix_n(), LCD_Seg_WriteString16Seg_n() and LCD_Seg_WriteString16Seg_n() API was changed from “uint8 * character” to “uint8 const character[]”	This is MISRA related change. MISRA doesn’t allow using array indexing of variables when they are declared as pointers.
3.20	An empty wakeup ISR was added to the component. The change concerns only PSoC 5LP devices.	The implementation of PSoC 5LP requires an interrupt in order to wakeup.
	Added equations for the calculation of DefaultStep.	
	Fixed equations for the HighDriveTimeValdef and HighDriveTimeValmax calculations.	
3.10	Fixed LCD_SegStat_Write7SegNumber() API.	There was a bug in the API which overrode the last digit of the number displayed on the LCD with 0. The issue only occurred when "leading zeroes" mode output was used.
	Added Keil function reentrancy support to the APIs.	Added the capability for customers to specify any individual generated function as reentrant.
3.0	Changed WRITE_PIXEL() macros definition from #define LCD_Seg_WRITE_PIXEL(pixelNumber, pixelState) LCD_Seg_WritePixel(pixelNumber, pixelState) to #define LCD_Seg_WRITE_PIXEL(pixelNumber, pixelState) (void) LCD_Seg_WritePixel(pixelNumber, pixelState)	There is no need to analyze the return value of the LCD_Seg_WritePixel() API inside component realization so the macros were changed to handle that
	Changed component’s verilog implementation to avoid STA combinational cycle warning which was prompted when using component in low-lower mode.	The asynchronous SR latch was replaced with a synchronous logic.

Version	Description of Changes	Reason for Changes / Impact
	Added a verification for unassigned helper pixels in the LCD_Seg_ReadPixel() API.	The new return status of 0xFF (LCD_Seg_Stat_PIXEL_UNKNOWN_STATE) was added to LCD_Seg_ReadPixel(). This status returned in case when the pixel exists in a helper but it isn't assigned.
	Function LCD_Seg_WriteBargraph() was optimized.	Code refactoring allowed some space savings in flash memory.
	Functions LCD_Seg_WriteString14Seg() and LCD_Seg_WriteString16Seg() were optimized.	Array indexing was changed to pointer incrementing, which saves RAM space and allows better execution time of these functions.
	API to LCD_Seg_Write7SegNumber() was optimized.	There was some ineffective code in the LCD_Seg_Write7SegNumber() API which reflected in the longer execution time of this API.
	Fixes to LCD_Seg_Wakeup() API.	LCD_Seg_Wakeup() API doesn't return any status although it should, so the API was fixed to return proper status.
2.10	Fixed LCD_Seg_Start() routine	It was silently disabling global interrupts, if they were previously enabled, because of an error in its implementation.
	Changed Driver Power Settings tab of the customizer. Added two new fields Custom Step and Default Step.	New options for High Drive Time parameter in low power mode are available. Besides the default step of increment, a custom more precise step could be chosen. It is based on the frequency of the Master Clock.
	Fixed issue which put the SegLCD to a Sleep prior completion of LCD refresh sequence.	Internal component signal lp_ack was going High on the start of the refresh sequence but correctly it should happen in the end of the refresh sequence.
	Fixed the issue which reflected in inverting the LCD glass image if High Drive Strength was set to "seg=2x com=2x" or "seg=1x com=4x"	The issue is in writing of the improper value to LCD Driver Control Register. This value generated by the AND operation with the mask and the mask is not proper. Changing the mask to a correct one fixed the problem.
	LCD_Seg_WriteBargraph() API fix.	Fixed issue with LCD_Seg_WriteBargraph() API which reflected in incorrect output of a bar graph pixels of the LCD when mode parameter was set to 10. The occurred if Position parameter satisfied condition "Position + 10 > Bar Graph pixels". In this case the last pixel of the Bar graph wasn't set but it should.
	Added characterization data to datasheet	
	Added information to the component that advertizes its compatibility with silicon revisions.	The tool returns an error if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
	Minor datasheet edits and updates	

Version	Description of Changes	Reason for Changes / Impact
2.0.a	Added notes to LCD_Seg_Init(), LCD_Seg_Start(), and LCD_Set_Stop() APIs in datasheet	
	Added information to the component that advertizes its compatibility with silicon revisions.	The tool returns an error if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
2.0	Added Sleep/Wakeup and Init/Enable APIs.	To support low power modes, as well as to provide common interfaces to separate control of initialization and enabling of most components.
	Added new API file - *SegLCD_PM.c* which contains declaration of Sleep mode APIs.	New requirement to support low power modes.
	Component was updated to support PSoC 3 ES3 and above. Updated the Configure dialog: ■ Removed obsolete controls: Enable Debug Mode, Low Drive Mode ■ Added new controls: Glass Size, High Drive Strength, Low Drive Strength ■ Changed selections of Driver Power Mode by changing old modes Always Active and Low Power to new ones No Sleep, Low Power ILO Low Power 32XTAL ■ Added custom characters Tab to allow users to create custom character sets for Dot Matrix helper ■ Made Bias type control editable and changed selection values for Bias Voltage	New requirements to support the PSoC 3 ES3 device and LCD HW architecture, thus a new 2.0 version was created. Version 1.xx supports PSoC 3 ES2 and PSoC 5 silicon revisions
	Added `=ReentrantKeil(LCD_Seg_ . "_...")` to the following functions: LCD_Seg_Stop() LCD_Seg_EnableInt() LCD_Seg_DisableInt() LCD_Seg_SetBias() LCD_Seg_WriteInvertState() LCD_Seg_ReadInvertState() LCD_Seg_RedPixel() LCD_Seg_SaveConfig() LCD_Seg_RestoreConfig()	Allows users to make these APIs reentrant if reentrancy is desired.

© Cypress Semiconductor Corporation, 2013-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

