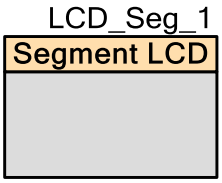
Segment LCD (LCD_Seg)
2.0



Features

- 2 to 768 pixels or symbols
- Static, 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.
- Supports PSoC 3 ES3 silicon revisions and above

General Description

The Segment LCD (LCD_Seg) component can directly drive 3.3 V and 5.0 V LCD glass at 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.)
- 1-255 element bar graphs

When to use a Segment LCD

Use the Direct Segment Drive LCD component when you need to directly drive 3.3 V or 5.0 V LCD glass at multiplex ratios from 2x to 16x. The Direct Segment Drive LCD component requires that the target PSoC device supports LCD direct drive.

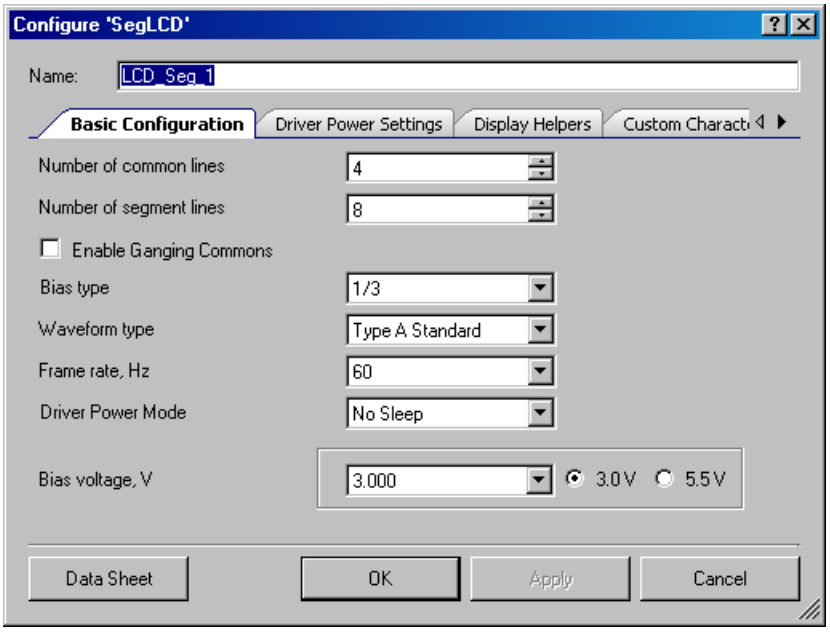
Input/Output Connections

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 check box to gang PSoC pins to drive common signals. Two PSoC pins will be 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

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 it is unique for every configuration. For a detailed description, refer to "Driver Power Modes" in the Functional Description section later in this data sheet.

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 will be always be turned on and chip will not enter to a sleep mode
- "Low Power using ILO": LCD DAC will be turned on but the chip will enter to a sleep between voltage transactions. As a wakeup source the component will use 1 KHz internal ILO.
- "Low Power using Ext 32 KHz crystal": LCD DAC will be turned on but the chip will enter to a sleep between voltage transactions. As a wakeup source the component will use 8 K tap from OPPS timer.

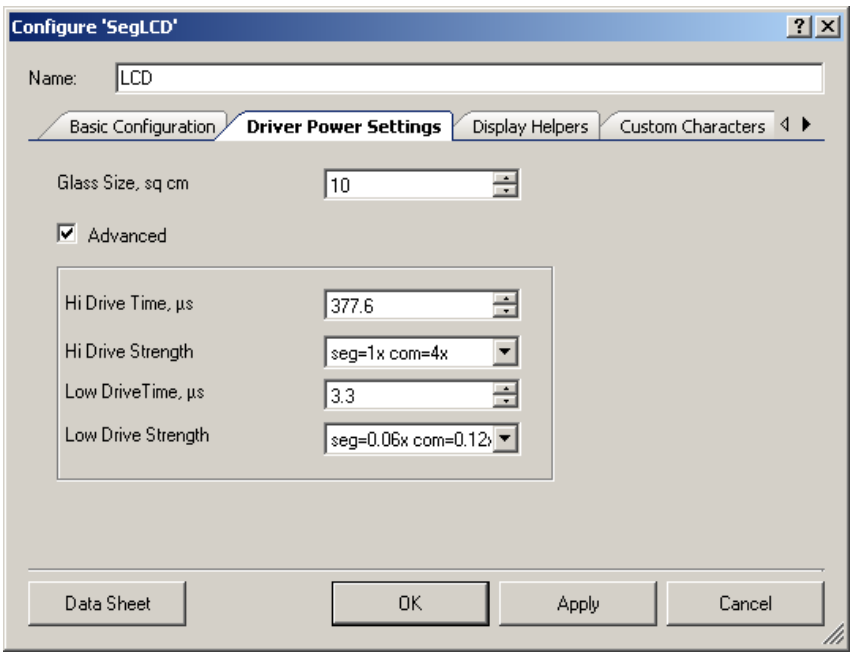
Note Depending on the low power mode you are using, use either an ILO set to 1 KHz or an external 32 KHz crystal connected and enabled. You can enable the 32 KHz or set the frequency of the ILO in Design Wide Resources Clock Editor.

Refer also to "Driver Power Modes" in the Functional Description section later in this data sheet.

Bias Voltage

This determines the bias voltage level for the LCD DAC. The range of possible values is from 2 V to 5 V or from 2 V to 3 V, depending on the supply source.

Driver Power Settings Tab



Glass Size

In this field the user will enter the approximate size of the glass in sq cm. And from that info and the number commons input on the basic configuration tab the approximate capacitive load for the commons will be calculated.

Hi Drive Time

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

Note If you change the **Frame Rate**, **Number of Common Lines**, or **Waveform Type** parameters, the **Hi Drive Time** parameter will be set automatically to half of the frame as it is its default value:

*HiDriveTimedef= 1/(FrameRate * (2 * NumCommonLines) * 256) * 128 (Type A)*

*HiDriveTimedef= 1/(FrameRate * NumCommonLines * 256) * 128 (Type B)*

The calculation of Hi Drive Time minimum value is defined later in this data sheet. The maximum value of Hi Drive Time in the current configuration is:

*HiDriveTimemax= 1/(FrameRate * (2 * NumCommonLines) * 256) * 254 (Type A)*

*HiDriveTimemax= 1/(FrameRate * NumCommonLines * 256) * 254 (Type B)*

If you have selected one of the Low Power modes, it is not recommended to set the Hi Drive Time to its maximum values. The use of Low Power mode will be ineffective because the device will enter Sleep mode for only a short time and in worst case it may not enter to a Sleep at all.

Hi Drive Strength

This parameter selects the drive strength for Hi Drive.

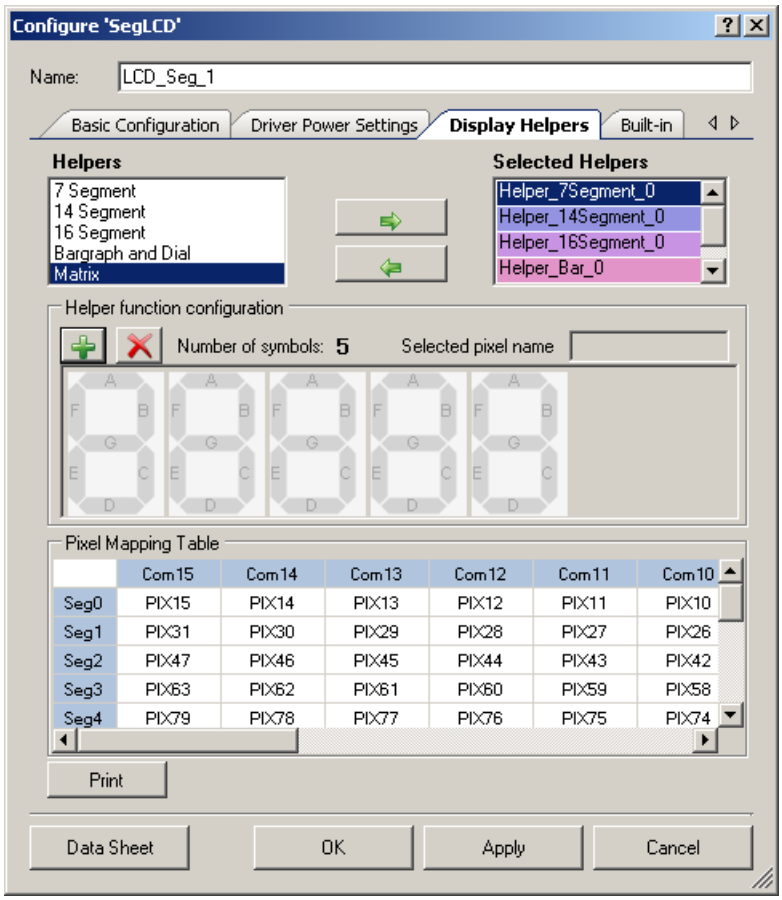
Low Drive Time

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 Hi Drive Strength and it sets automatically depending on the Hi 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

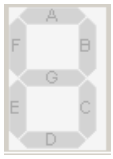
You may add one or more helpers to the **Selected Helpers** list by selecting the desired helper type in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to support the new helper, it will not be added. To delete a helper, select it in the **Selected Helpers** list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

Note: It is important to set the number of common and segment lines for the component prior to 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 API section for more information.

- **7 Segment Helper –** This helper may be 1 to 5 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 the pixels to the left or right of the specified pixel

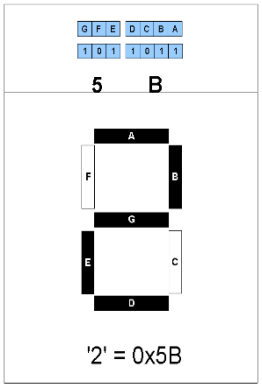
- **Dot Matrix Helper –** This helper supports up to eight character elements. The component supports x5x7 or x5x8 row/column characters. Longer strings of characters can be created by configuring two or more dot matrix helpers to define adjacent dot matrix sections of the display. The helper displays 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 8 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.

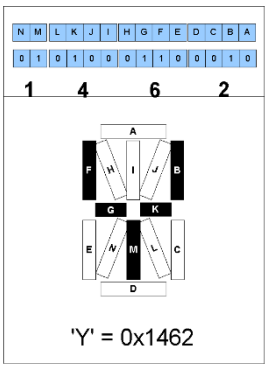
Character Encoding

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 customizer).

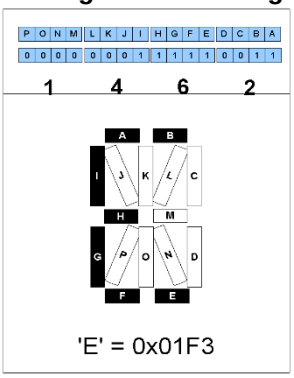
7 Segment encoding



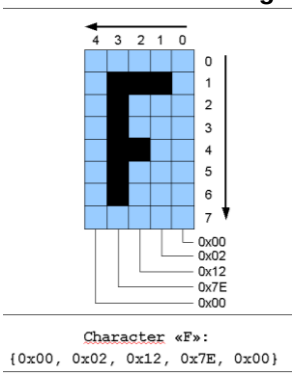
14 Segment encoding



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/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 **Pixel Mapping Table**.

The default naming for pixels associated with a Helper symbol have 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 4th 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

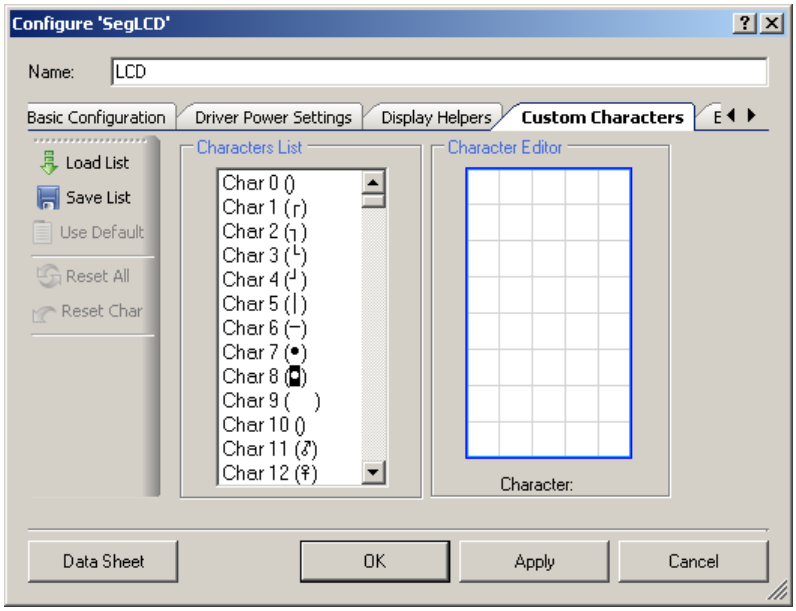
The Pixel Mapping Table is a representation of the frame buffer. For the API functions to work properly, each pixel from the Helper Function Configuration must be assigned to a pixel location in the Pixel Mapping Table. Refer to the data sheet for your LCD glass for the information you will need to make 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 on 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 **Character 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.

Placement

The LCD_Seg component implementation consists of two parts. The LCDDAC is a fixed function hardware block in the PSoC that is used by this component. Additional timing logic for the drive signals is implemented in UDBs. UDB resources are automatically placed in the UDB array during the project generation process.

Note: Only one instance of the component can be used in a project. A placement error will be generated during the build process if more than one instance of the component is used in a project.

Default pin assignments are made during the build process and can be modified using the Pin Editor in the PSoC Creator Design Wide Resources tool.

Resources

Digital Blocks					API Memory (Bytes)		
Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	Pins (per External I/O)
0	TBD	TBD	TBD	0	TBD	TBD	TBD

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 cover 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".

Function	Description
LCD_Seg_Init	Perform initialization of the component.
LCD_Seg_Enable	Enables the power to LCD fixed hardware and enables generation of UDB signals.
LCD_Seg_Start	Starts the LCD component and enables the required interrupts,
LCD_Seg_Stop	Disables the LCD component and associated interrupts and DMA channels.
LCD_Seg_EnableInt	Enables the LCD interrupt/s. Not required if LCD_Seg_Start called
LCD_Seg_DisableInt	Disables the LCD interrupt. Not required if LCD_Seg_Stop called
LCD_Seg_SetBias	Sets the bias level for the LCD glass to one of 128 values.

Function	Description
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	Prepare the component to enter a Sleep mode.
LCD_Seg_Wakeup	Wakes up components from a Sleep mode.
LCD_Seg_SaveConfig	Saves component configuration.
LCD_Seg_RestoreConfig	Restores component configuration.

Global Variables

Variable	Description
LCD_Seg _initVar	Indicates whether the LCD_Seg 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 in after the first call to the LCD_Seg_Start() routine. If reinitialization of the component is required the variable should be set to 0 before the LCD_Seg_Start() routine is called. Alternately, the LCD_Seg can be reinitialized by calling the LCD_Seg_Init() and LCD_Seg_Enable() functions.

void LCD_Seg_Init(void)

Description: Initialize or restore the component parameters per the Configure dialog settings. 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 32KHz External Xtal component mode.

void LCD_Seg_Enable(void)

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

Parameters: None

Return Value: None

Side Effects: None

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.

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 Init() API. This only valid for Low Power 32KHz External Xtal component mode.

void LCD_Seg_EnableInt(void)

Description: Enables the LCD interrupts. Not required if LCD_Seg_Start is called. An interrupt occurs after every LCD update (TD completion).

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_DisableInt(void)

Description: Disables the LCD interrupts. Not required if LCD_Seg_Stop is called.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_SetBias(uint8 biasLevel)

Description: This function sets the bias level for the LCD glass to one of 128 values. The actual number of values is limited by the Analog supply voltage, Vdda. The bias voltage can not exceed Vdda. Changing the bias level affects the LCD contrast.

Parameters: (uint8) biasLevel: bias level for the display

Return Value: None

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: is the packed number that points to the pixels 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 (4 bits) is the byte address in the multiplex row and the MSB low nibble (4 bits) is the multiplex row number. The generated component .h file includes a #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 pixels 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 (4 bits) is the byte address in the multiplex row and the MSB low nibble (4 bits) is the multiplex row number. The generated component .h file includes a #defines of this format for each pixel.

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

Side Effects: None

void LCD_Seg_Sleep(void)

Description: Prepare the component to enter a Sleep mode.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_Wakeup(void)

Description: Wakes up components from a Sleep mode.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_SaveConfig(void)

Description: Saves component configuration.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_RestoreConfig(void)

Description: Restores component configuration.

Parameters: None

Return Value: None

Side Effects: None

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 0-9 and A-F. 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 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 7segment 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.

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 element(s) 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	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.
2-10	Display the Location segment and 2-10 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 char 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: ASCII value of the character to display (printable characters with ASCII values 0 – 127)

(uint8) position: 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 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 element(s) 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: 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: None

void LCD_Seg_PutChar16Seg_n(uint8 character, uint8 position)

Description: This function displays an 8-bit char 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: ASCII value of the character to display (printable ASCII and table extended characters with values 0 – 255)

(uint8) position: 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 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 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: 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: None

void LCD_Seg_PutCharDotMatrix_n(uint8 character, uint8 position)

Description: This function displays an 8-bit char 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 element(s). 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 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 element(s). 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: 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: None

Defines

- LCD_Seg_COMM_NUM – Defines the number of common lines in the user-defined display for current configuration of the component.
- LCD_Seg_SEG_NUM – Defines the number of segment lines for the user-defined display current configuration of the component.
- LCD_Seg_BIAS_TYPE – Defines the bias type for the user-defined display current configuration of the component.
- LCD_Seg_BIAS_VOLTAGE – Defines default bias voltage level for user-defined display. This value will be set in LCDDAC control Register during Initialization process.
- LCD_Seg_FRAME_RATE – Defines the refresh rate for the user-defined display 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 WritePixel function.
- LCD_Seg_READ_PIXEL – This is a macro define of the ReadPixel function.
- LCD_Seg_FIND_PIXEL – This macro calculates pixel location in the frame buffer. It uses information from customizer pixel table and information of physical pins which will be dedicated for the LCD. This macro is the base of pixel mapping mechanism. Every pixel name from the pixel table will be defined with calculated pixel location in the frame buffer and APIs will use pixel names to access the respective pixel.

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the Segment LCD component. This example assumes the component has been placed in a design with the default name LCD_Seg_1.

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

void main()
{
   LCD_Seg_1_Start();
   LCD_Seg_1_SetBias(127);
   LCD_Seg_1_WritePixel(LCD_Seg_1_PIX0,1);
   LCD_Seg_1_ReadPixel(LCD_Seg_1_PIX0);
   LCD_Seg_1_ClearDisplay();
   LCD_Seg_1_Stop();
}
```

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 provide 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. By default LCD_Seg configuration is:

- 4 common lines
- 8 Segment lines
- 1/3 Bias Type
- 60 Hz refresh rate
- No Sleep power mode
- 3 V Bias Voltage
- Glass Size – 10 sq sm
- 1041.7 us Hi Drive time
- seg - 1x, com - 2x Hi Drive strength

- 8.1 us Low Drive time
- seg – 0.06x, com 0.12x Low Drive strength
- No display helpers are defined. Default API generation will 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.

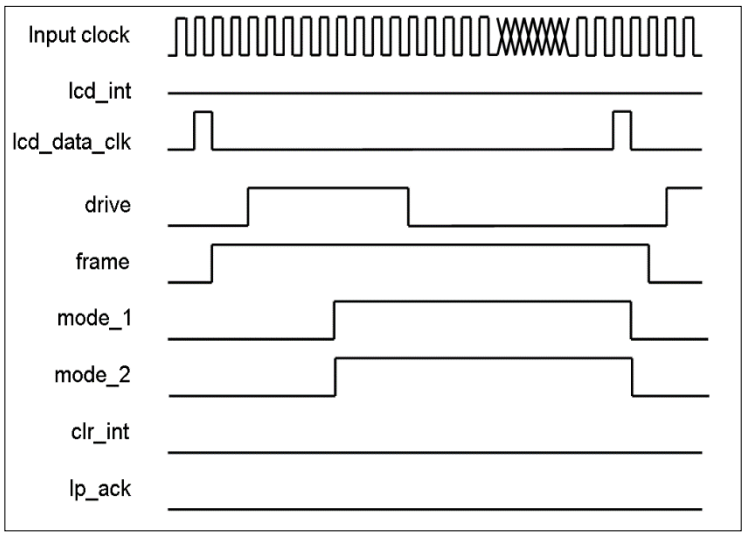
Driver Power Modes

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. The following 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 will put back the entire device into Sleep mode until the next refresh sequence.

In this 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 K XTAL)], 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*

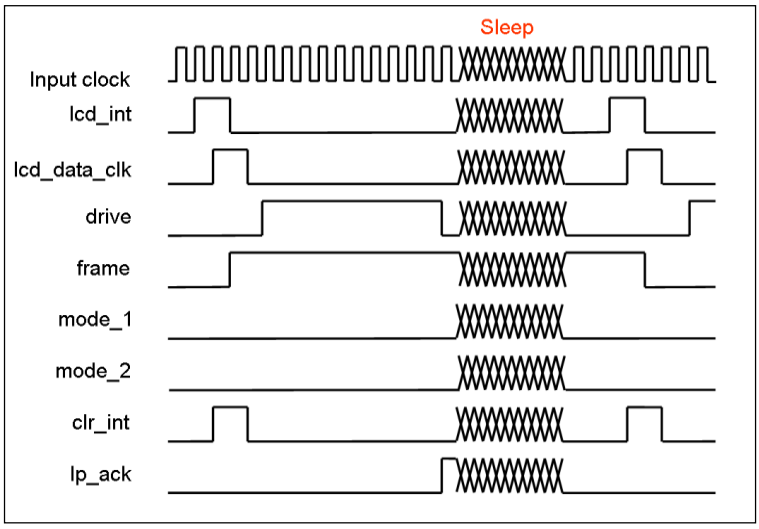
62.5 Hz is the maximum Frame Rate that the LCD timer can provide when it will generate a wake-up signal on every clock cycle of input frequency.

In the real world, the LCD timer needs one cycle to set the wake-up event and one cycle to clear it. So it means the maximum Frame Rate should divided by two, which results in 31.25 Hz. From this value, we take only the integer and get 31 Hz as the actual maximum Frame Rate.

For the Type B waveforms the actual Frame Rate will be two times larger than for Type A waveforms, because Type B waveforms require two times fewer wake-up events.

The following shows waveforms for UDB-generated signals of the LCD_Seg component for Low Power mode.

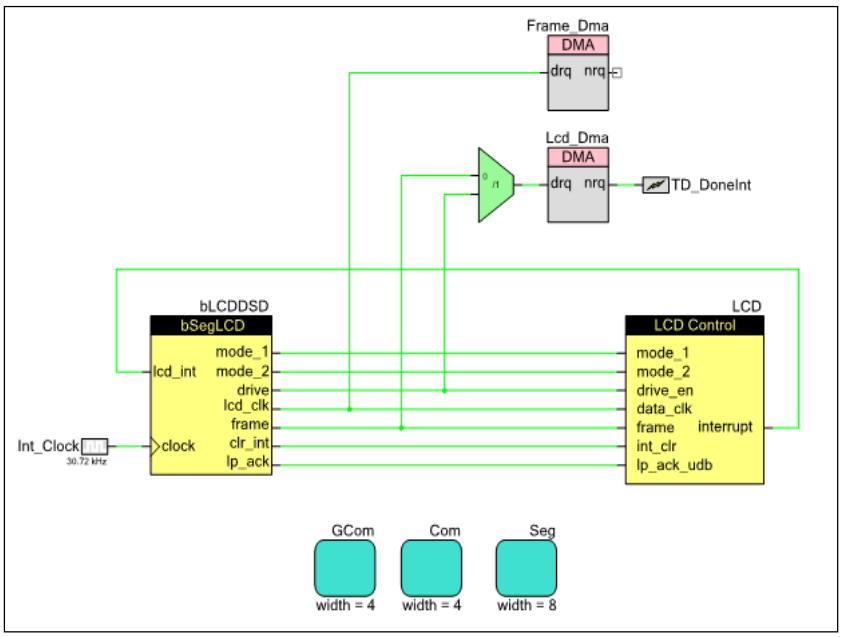
Figure 2 Segment LCD control signals (Low Power mode)



Block Diagram and Configuration

The following diagram 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.

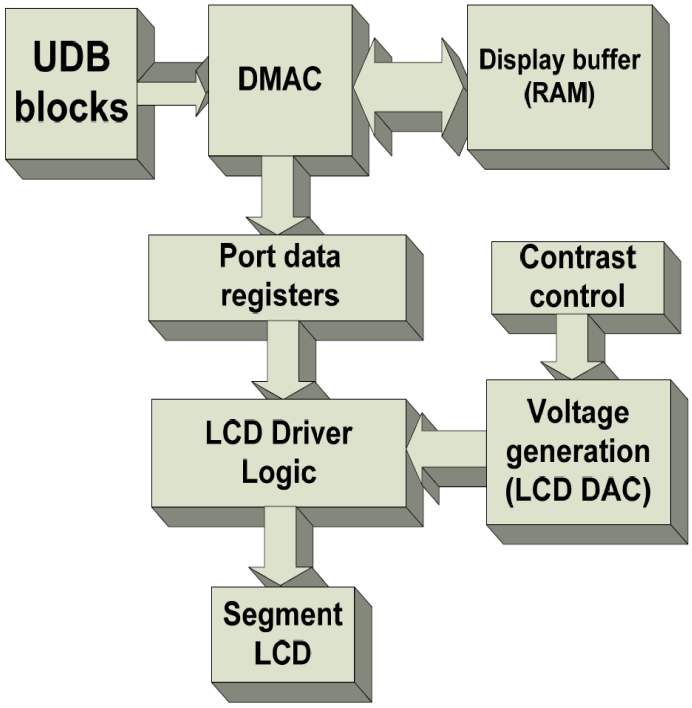
Figure 3 Segment LCD Component Schematic



- 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. Also, there is 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: This bit enables generation of all internal signals described in above sections.
- mode 1: middle bit of mode[2:0] bit field which defines Hi and Low drive strength.
- mode 2: higher bit of mode[2:0] bit field which defines Hi and Low drive strength.
- frame done: Generates a synchronous pulse after completion of DMA Frame transaction.
- frame: Generates Frame signal for 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: Enable 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, set output buffer in LCD Drivers to ground if set to "1" otherwise set LCD Drivers to Hi-Z.
- mode 0: lower bit of mode[2:0] bit field which defines Hi and Low drive strength.
- invert: If set inverts data on the segment pins.
- pts: "0" - normal operation, VOUT = VIO-0.5V, "1" - VOUT = VIO.

- bypass_enable: If "1" - bypasses the drive buffer within the LCD Driver and instead connects directly to the voltage input selected. "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.

References

Not applicable

DC and AC Electrical Characteristics

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Description	Conditions	Min	Typical	Max	Units
Input Voltage Range	LCD operating current					µA
	LCD Bias Range		2	---	5.2	V
	LCD Bias Step Size		---	25.2	---	mV
	LCD Capacitance per Segment/Common Driver	Drivers may be ganged	---	500	5000	pF
	Iout Per Segment Driver					
	Strong Drive		120	160	200	µA
	Weak Drive		---	0.5	---	µA
	Weak Drive 2			1		µA
	No Drive		---	2	---	µA
	Iout Per Common Driver					
	Strong Drive		160	220	300	µA
	Weak Drive		---	11	---	µA
	Weak Drive 2		---	22	---	µA
	No Drive		---	<25	---	µA

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.0.c	Minor datasheet edit.	
2.0.b	Minor datasheet edit.	
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 - *CharLCD_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, Hi 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 ES1 silicon revisions
	Added `=ReentrantKeil($INSTANCE_NAME . "_...")` to the following functions: Stop() EnableInt() DisableInt() SetBias() WriteInvertState() ReadInvertState() RedPixel() SaveConfig() RestoreConfig()	Allows users to make these APIs reentrant if reentrancy is desired.

© Cypress Semiconductor Corporation, 2010-2016. 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, 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.