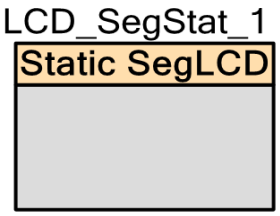
Static Segment LCD (LCD_SegStat) 1.50



Features

- 1 to 61 pixels or symbols
- 10 to 150 Hz refresh rate
- User-defined pixel or symbol map with optional 7 segment, 14 segment, 16 segment and bar graph calculation routines
- Direct drive static (1 common) LCDs

General Description

The Static Segment LCD (LCD_SegStat) component can directly drive 3.3 V and 5.0 V LCD glass. This component provides an easy method of configuring the PSoC device for your custom or standard glass.

Each LCD pixel/symbol may be either on or off. The Static 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
- 1-255 element bar graphs

When to Use a Static Segment LCD

Use the Static Segment Drive LCD component when you need to directly drive 3.3 V or 5.0 V LCD glass in a static mode (for glass configurations that have only one common line). The Static Segment LCD component does not require the target PSoC device to provide LCD drive hardware resources. The component can be used on any target chip that has sufficient pins to support the required number of common and segment lines.

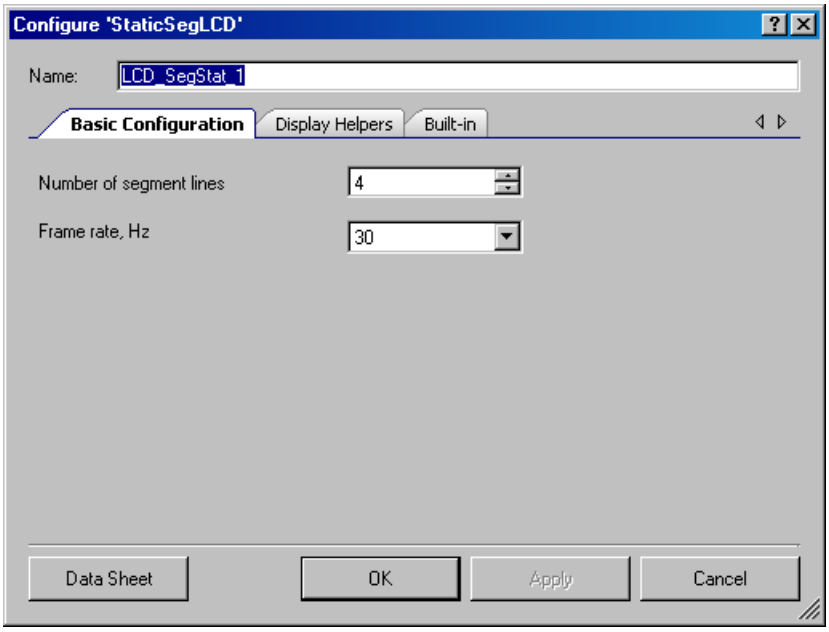
Input/Output Connections

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 Static Segment LCD component onto your design and double-click it to open the Configure dialog.

Basic Configuration Tab



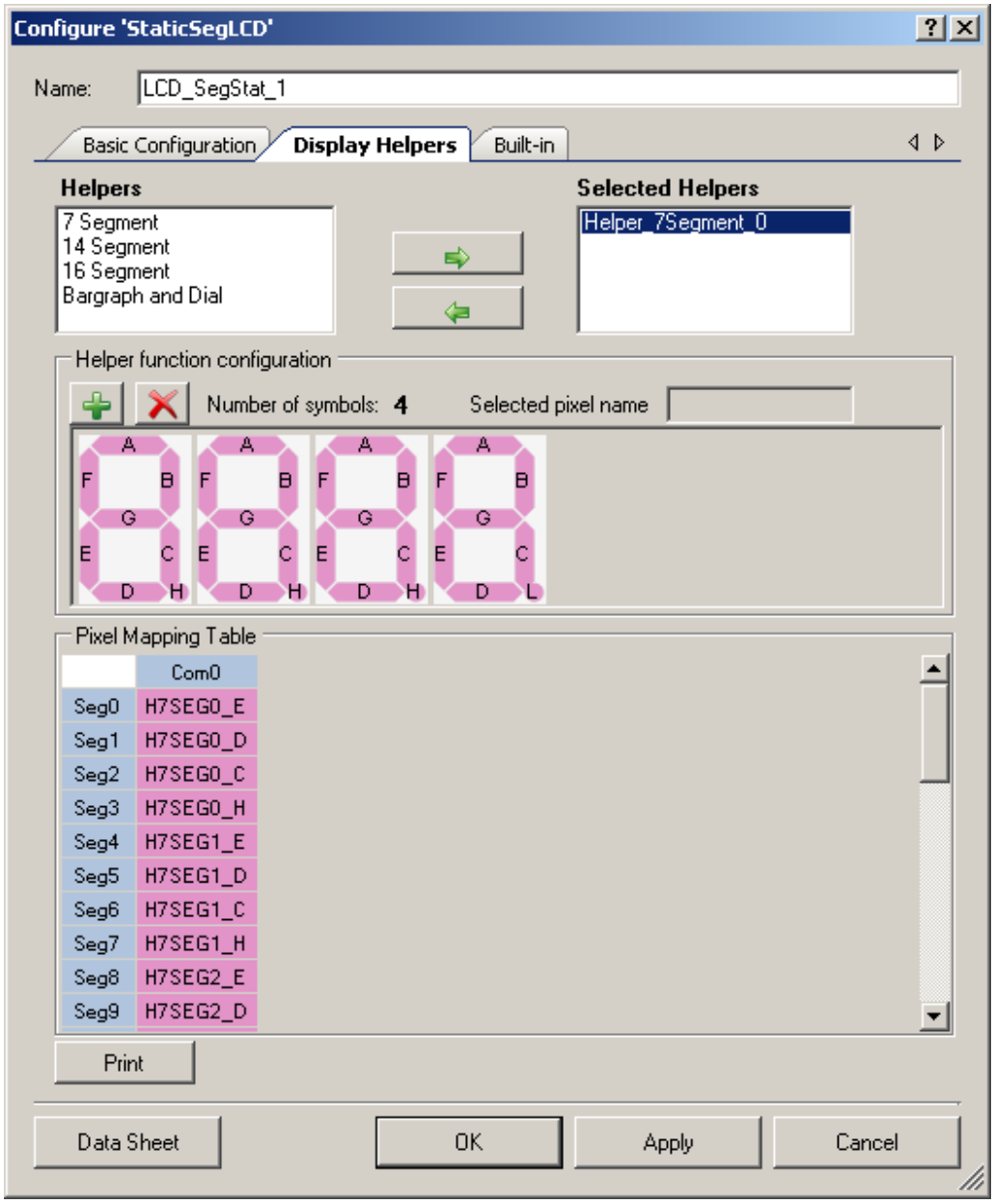
Number of Segment Lines

Defines the number of segment lines in the user-defined display. The default is 4.

Frame Rate

Determines the refresh rate of the display. Possible values include 10 Hz to 150 Hz. The default is 30 Hz.

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
- 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

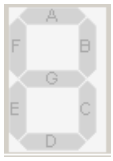
You may add one or more helpers to the **Selected Helpers** list by selecting the desired helper type in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to support the new helper, it will not be added. To delete a helper, select it in the **Selected Helpers** list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

Note: Once you have added a Display Helper to the component, you will not be able to change the number of common or segment lines. 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 can change the number of common or segment lines.

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



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 it 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 the pixel name, then the entire name will be shown on the symbol image even if they will have common prefix.

Note All pixel names must be unique.

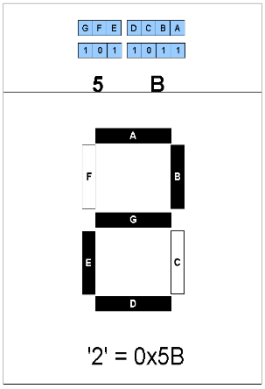
When a Helper function symbol element is assigned to a pixel in the Pixel Mapping Table (described below), the pixel assumes the name of the helper symbol element. The helper symbol

element name supersedes the default pixel name, but does not replace it. You cannot re use the default pixel name of pixels that are associated with a Helper function.

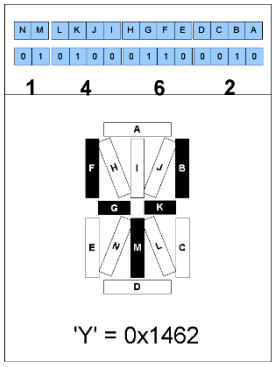
Character Encoding

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).

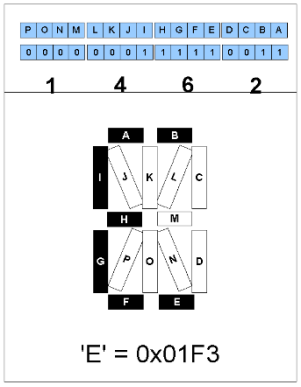
7 Segment encoding



14 Segment encoding



16 Segment encoding



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.

Clock Selection

Segment LCD component uses one internal clock and does not require an external clock. Once the component is placed, the clock is automatically dedicated to the LCD component. The clock generates frame rate frequency

Placement

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

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, Bar Graph helper: Low level API functions set + Bar Graph helper High Level API.

Resources	Resource Type						API Memory (Bytes)		Pins (per External I/O)
	Datapath Cells	PLDs	Status Cells	Control/ Count7 Cells	Sync Cells	Interrupts	Flash	RAM	
Basic	0	0	0	0	0	1	1977	94	2-62
Basic, 7-Segment helper	0	0	0	0	0	1	2473	104	2-62
Basic, 14-Segment helper	0	0	0	0	0	1	3305	106	2-62
Basic, 16-Segment helper	0	0	0	0	0	1	2322	104	2-62
Basic, Bar Graph helper	0	0	0	0	0	1	3705	103	2-62

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_SegStat_1" to the first instance of a component in a given design. You can rename the instance to any unique value that follows the

syntactic 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_SegStat".

Function	Description
LCD_SegStat_Start	Starts the LCD component and DMA channels. Initializes the frame buffer. Does not clear the frame buffer RAM if previously defined.
LCD_SegStat_Stop	Disables the LCD component and associated interrupts and DMA channels. Does not clear the frame buffer.
LCD_SegStat_EnableInt	Enables the LCD interrupts. Not required if LCD_SegStat_Start called.
LCD_SegStat_DisableInt	Disables the LCD interrupt. Not required if LCD_SegStat_Stop called.
LCD_SegStat_ClearDisplay	This function clears the display RAM of the frame buffer.
LCD_SegStat_WritePixel	This function sets or clears a pixel based on PixelState. The pixel is addressed by a packed number.
LCD_SegStat_ReadPixel	This function reads the state of a pixel in the frame buffer. The pixel is addressed by a packed number.
LCD_SegStat_Sleep	Stop the LCD and save the user configuration.
LCD_SegStat_Wakeup	Restore and enable the user configuration.
LCD_SegStat_Init	Configures every-frame interrupt and initializes the frame buffer.
LCD_SegStat_Enable	Enables clock generation for the component.
LCD_SegStat_SaveConfig	Save the LCD configuration.
LCD_SegStat_RestoreConfig	Restore the LCD configuration.

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” index value in the function name.

Global Variables

Variable	Description
LCD_SegStat_initVar	Indicates whether the LCD_SegStat has been initialized. The variable is initialized to 0 and set to 1 the first time LCD_SegStat_Start() is called. This allows the component to restart without reinitialization after the first call to the LCD_SegStat_Start() routine. If reinitialization of the component is required, then the LCD_SegStat_Init() function can be called before the LCD_SegStat_Start() or LCD_SegStat_Enable() function.

uint8 LCD_SegStat_Start (void)

Description: Starts the LCD component, 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_BAD_PARAM	One or more function parameters were invalid
CYRET_SUCCESS	Function completed successfully

Side Effects: None

void LCD_SegStat_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: LCD_SegStat_Stop API doesn't release the DMA channel and transfer descriptors. For the component to work properly, you must do that manually. In the header of your C file add the following declaration "extern void LCD_SegStat_DmaDispose(void);". Then, each time you call LCD_SegStat_Stop() or LCD_SegStat_Sleep() in your source code, place a call to the LCD_SegStat_DmaDispose() routine.

void LCD_SegStat_EnableInt(void)

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

Parameters: None

Return Value: None

Side Effects: None

void LCD_SegStat_DisableInt(void)

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

Parameters: None

Return Value: None

Side Effects: None

void LCD_SegStat_ClearDisplay(void)

Description: This function clears the display RAM of the page buffer.

Parameters: None

Return Value: None

Side Effects: None

uint8 LCD_SegStat_WritePixel(uint16 PixelNumber, uint8 PixelState)

Description: This function sets or clears a pixel in the frame buffer based on the PixelState parameter. The Pixel is addressed with 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.

(uint8) PixelState: The PixelNumber specified is set to this pixel state. Symbolic names for the pixel state are provided, and their associated values are shown here:

Value	Description
LCD_SegStat_PIXEL_STATE_OFF	Set the pixel to off.
LCD_SegStat_PIXEL_STATE_ON	Set the pixel to on.
LCD_SegStat_PIXEL_STATE_INVERT	Invert the pixel's current 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.

Return Value	Description
CYRET_BAD_PARAM	Packed byte address or row value was invalid
CYRET_SUCCESS	Function completed successfully

Side Effects: None

uint8 LCD_SegStat_ReadPixel(uint16 PixelNumber)

Description: This function reads a pixel's state 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

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

Value	Description
0	The pixel is off.
1	The pixel is on.

Side Effects: None

void LCD_SegStat_Sleep(void)

Description: This is the preferred routine to prepare the component for sleep. The LCD_SegStat_Sleep() routine saves the current component state. Then it calls the LCD_SegStat_Stop() function and calls LCD_SegStat_SaveConfig() to save the hardware configuration.
Call the LCD_SegStat_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 pins drive modes.

void LCD_SegStat_Wakeup(void)

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

Parameters: None

Return Value: None

Side Effects: Calling the LCD_SegStat_Wakeup() function without first calling the LCD_SegStat_Sleep() or LCD_SegStat_SaveConfig() function may produce unexpected behavior.

void LCD_SegStat_Init(void)

Description: Initializes or restores the component parameters per the Configure dialog settings. Configures every frame interrupt and initializes the frame buffer.

Parameters: None

Return Value: None

Side Effects: None

void LCD_SegStat_Enable(void)

Description: Enables clock generation for the component.

Parameters: None

Return Value: None

Side Effects: None

void LCD_SegStat_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_SegStat_Sleep() function.

Parameters: None

Return Value: None

Side Effects: None

void LCD_SegStat_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_SegStat_Sleep() function.

Parameters: None

Return Value: None

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

Optional Helper APIs

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

Function	Description
LCD_SegStat_Write7SegDigit_n	Displays a hexadecimal digit on an array of 7 segment display elements.
LCD_SegStat_Write7SegNumber_n	Displays an integer value on a 1 to 5 digit array of 7 segment display elements.
LCD_SegStat_WriteBargraph_n	Displays an integer location on a linear or circular bar graph
LCD_SegStat_PutChar14Seg_n	Displays a character on an array of 14 segment alphanumeric character display elements.
LCD_SegStat_WriteString14Seg_n	Displays a null terminated character string on an array of 14 segment alphanumeric character display elements.
LCD_SegStat_PutChar16Seg_n	Displays a character on an array of 16 segment alphanumeric character display elements.
LCD_SegStat_WriteString16Seg_n	Displays a null terminated character string on an array of 16 segment alphanumeric character display elements.

void LCD_SegStat_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_SegStat Write7SegNumber_n(uint8 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.

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

Return Value: None

Side Effects: None

void LCD_SegStat_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_SegStat_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_SegStat_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_SegStat_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_SegStat_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

Pins APIs

These API functions are used to change drive mode of pins used by Static Segment LCD component. These APIs are most often used to place Static Segment LCD component pins into the Hi-Z mode to minimize leakage while the device is in a low power mode.

Function	Description
LCD_SegStat_ComPort_SetDriveMode	Sets the drive mode for the pin used by common line of Static Segment LCD component.
LCD_SegStat_SegPort_SetDriveMode	Sets the drive mode for all pins used by segment lines of Static Segment LCD component.

void LCD_SegStat_ComPort_SetDriveMode (uint8 mode)

Description: Sets the drive mode for the pin used by common line of Static Segment LCD component.

Parameters: (uint8) mode: Desired drive mode. See the Pins component datasheet for information on drive modes.

Return Value: None

Side Effects: None

LCD_SegStat_SegPort_SetDriveMode(uint8 mode)

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

Parameters: (uint8) mode: Desired drive mode. See the Pins component datasheet for information on drive modes.

Return Value: None

Side Effects: None

Defines

LCD_SegStat_SEG_NUM

Defines the number of segment lines for the user-defined display current configuration of the component.

LCD_SegStat_FRAME_RATE

Defines the refresh rate for the user-defined display current configuration of the component.

LCD_SegStat_WRITE_PIXEL

This is a macro define of the LCD_SegStat_WritePixel() function.

LCD_SegStat_READ_PIXEL

This is a macro define of the LCD_SegStat_ReadPixel() function.

LCD_SegStat_FIND_PIXEL

This macro calculates pixel location in the frame buffer. It uses information from the customizer pixel table and information of physical pins dedicated for the LCD. This macro is the basis of the 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

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.

Functional Description

The Static 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. 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.

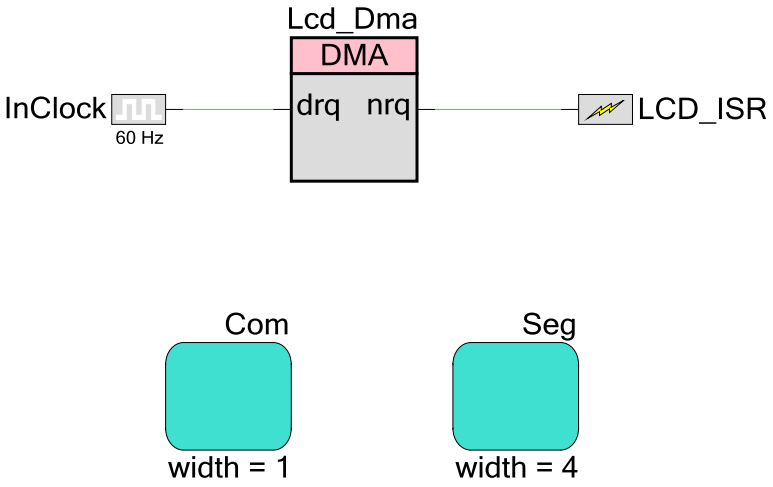
Default Configuration

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

- 4 Segment lines
- 30 Hz refresh rate
- No display helpers are defined. Default API generation will not include functions for any of the supported display elements.

Block Diagram and Configuration

The following diagram shows the schematic representation of the Segment LCD – Static component. The component does not require special purpose LCD drive hardware. The component makes use of the DMA and standard digital port I/Os.



This diagram shows the internal schematic for the Static Segment LCD component. It consists of a DMA component, ISR component, clock component, and two LCD ports.

- The **DMA** component is used to transfer data from the frame buffer to the LCD data registers through the aliased memory area.

- The **LCD Ports (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.
- **ISR** component available for user interrupt.

Top Level Architecture

The architecture of the component is very simple. There only a few blocks are used in it. The base of the component is a frame buffer that contains LCD data. The frame buffer can be modified using the LCD_SegStat_WritePixel() function or a high-level function that refers to the WritePixel() function. The DMA then transfers the LCD data to port registers via aliased memory. The DMA transactions are triggered by an internal clock set with a precalculated value to provide a proper refresh rate for the LCD.

Registers

None

References

Not applicable

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.50.c	Minor datasheet edit.	
1.50.b	Minor datasheet edit.	
1.50.a	Added manual call workaround to LCD_SegStat_DmaDispose() description in datasheet	Missing call to LCD_SegStat_DmaDispose() in the LCD_SegStat_Stop() routine.
	Added Pin APIs section to data sheet.	
	Minor datasheet edits and updates.	
1.50	Added sleep and wakeup APIs.	To support low power modes.
	Added backup structure which stores the state of the component.	Backup structure stores component state (enabled/disabled) prior to entering sleep mode. When LCD_SegStat_Wakeup() API called to wake the component from sleep this structure puts the component back into the state it was before entering the Sleep mode.

Version	Description of Changes	Reason for Changes / Impact
	Added tooltip in the Helpers tab.	This was done because the default size of the Configure dialog does not show the Pixel Mapping Table.
	Added numeric up-down controls and edit fields for Number of Segment Lines set enabled even if user added helper functions, earlier they were disabled.	Previously, if you wanted to change the number of segment lines after adding helpers, you had to delete all helpers and lose all configuration information. After adding a helper to a segment LCD component, you can go back and change the number of segment lines. If the number of segments increases, everything is clear. If the number of segments decreases, you will see a warning that you may lose configuration information. If you have assigned pixels on segment lines that you want to remove, they will be unassigned.
	Added error provider for selected pixel name text box. If you enter a wrong value in the text box, an error icon appears and you cannot leave the text box until the value is correct. Error icon has a tool tip with a problem description.	Error provider is a better way to handle incorrect values. It is used instead of showing message boxes.
	Following global variables were redefined with low camel case names: #define `$LCD_SegStat`_Buffer #define `$LCD_SegStat`_Channel #define `$LCD_SegStat`_TermOut #define `$LCD_SegStat`_TD #define `$$LCD_SegStat`_GCommons #define `$$LCD_SegStat`_Commons	Coding standards require variables to follow low camel case naming style. These variables became obsolete and will be removed in future.
1.30.b	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
	Made change log cumulative for v1.30 datasheet	
1.30.a	Moved local parameters to formal parameter list.	To address a defect that existed in PSoC Creator v1.0 Beta 4.1 and earlier, the component was updated so that it could continue to be used in newer versions of the tool. This component used local parameters, which are not exposed to the user, to do background calculations on user input. These parameters have been changed to formal parameters which are visible, but un-editable. There are no functional changes to the component but the affected parameters are now visible in the “expression view” of the customizer dialog.

Version	Description of Changes	Reason for Changes / Impact
1.30	Updated generated source code.	Added *StaticSegLCD_INT.c* file to component's library. Component's source code was changed to use *CyLib.h* function for internal interrupt configuration. Added a fix for DMA to work properly in 32-bit address space. Added an equation to component source file which selects proper termination signal from DMA.
1.20.b	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
	Made datasheet change log cumulative	
1.10.b	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
1.10.a	Moved local parameters to formal parameter list.	To address a defect that existed in PSoC Creator v1.0 Beta 4.1 and earlier, the component was updated so that it could continue to be used in newer versions of the tool. This component used local parameters, which are not exposed to the user, to do background calculations on user input. These parameters have been changed to formal parameters which are visible, but un-editable. There are no functional changes to the component but the affected parameters are now visible in the “expression view” of the customizer dialog.

© 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.