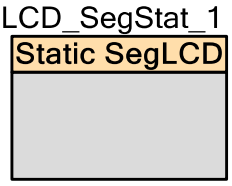
Static Segment LCD (LCD_SegStat) 1.20



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.

General Description

The Static Segment LCD (LCD_SegStat) component can directly drive 3.3V and 5.0V 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 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 3.3V or 5.0V 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.

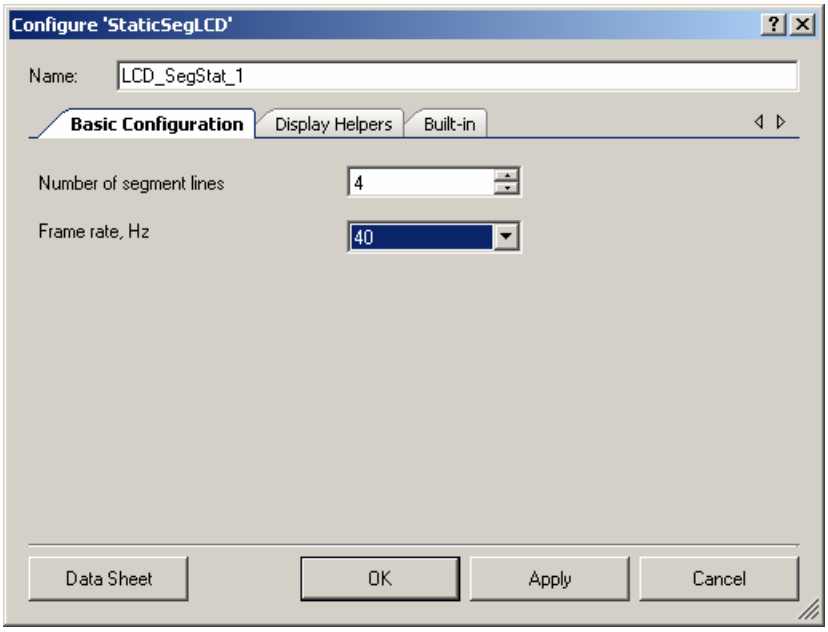
Input/Output Connections

Not applicable

Parameters and Setup

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

Basic Configuration Tab



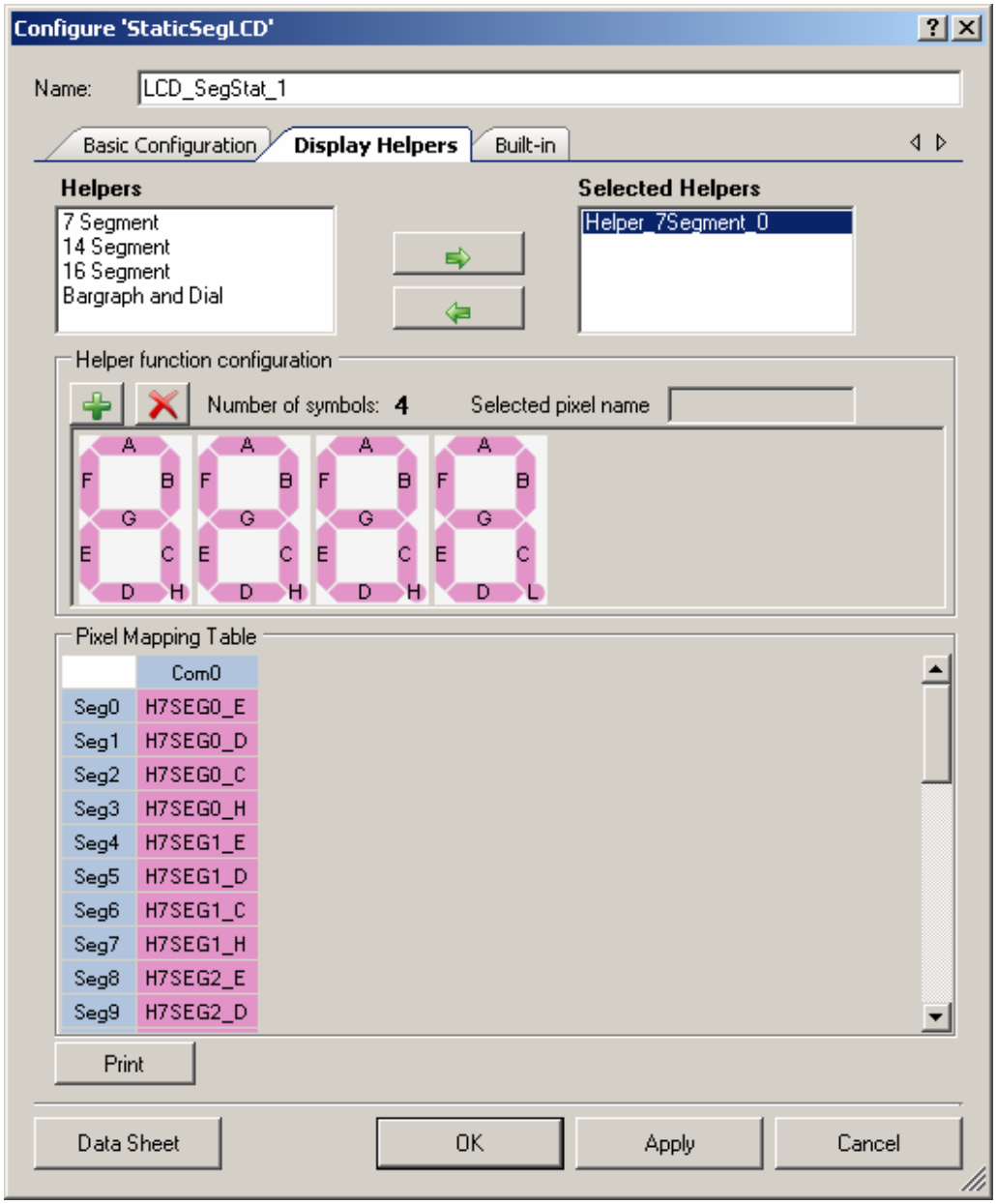
Number of Segment Lines

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 10Hz to 150Hz. The default is 30Hz.

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

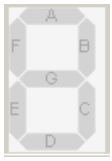
You may add one or more helpers to the **Selected Helpers** list by selecting the desired helper type in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to support the new helper, it will not be added. To delete a helper, select it in the **Selected Helpers** list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



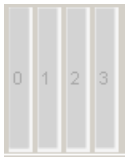
- **14 Segment Helper –** This helper may be up to 20 characters in length. It may display a single ASCII character or a null terminated string. Possible values are standard ASCII printable characters (with codes from 0 to 127).



- **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 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

If a common prefix is maintained for all of the pixels in a helper, only the unique portion of the pixel name is shown on the symbol image. If a common prefix is not used for all of the pixels in the helper, the full pixel name will be displayed. All pixel names must be unique.

Note: 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.

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

Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
0	0	0	0	0	?	?	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 enables the required interrupts, DMA channels, frame buffer and hardware. 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 interrupt/s. 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_Write7SegDigit_n	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. This function is only included if a 7 segment display element is defined in the component customizer.
LCD_SegStat_Write7SegNumber_n	This function displays an integer value on a 1 to 5 digit array of 7 segment display elements. This function is only included if a 7 segment display element is defined in the component customizer.
LCD_SegStat_WriteBargraph_n	This function displays an integer location on a linear or circular bar graph This function is only included if a bar graph display element is defined in the component customizer
LCD_SegStat_PutChar14Seg_n	This function displays a character on an array of 14 segment alphanumeric character display elements. Multiple, 14 segment alphanumeric display element groups can be defined 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.
LCD_SegStat_WriteString14Seg_n	This function displays a null terminated character string on an array of 14 segment alphanumeric character display elements. Multiple 14 segment alphanumeric display element groups can be defined 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.
LCD_SegStat_PutChar16Seg_n	This function displays a character on an array of 16 segment alphanumeric character display elements. Multiple,16 segment alphanumeric display element groups can be defined and are addressed through the suffix (n) in the function name. This functions is only included if a 16 segment display element is defined in the component customizer.

Function	Description
LCD_SegStat_WriteString16Seg_n	This function displays a null terminated character string on an array of 16 segment alphanumeric character display elements. Multiple 16 segment alphanumeric display element groups can be defined and are addressed through the suffix (n) in the function name. This functions is only included if a 16 segment display element is defined in the component customizer

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.

uint8 LCD_SegStat_Start (void)

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

Parameters: None

Return Value: (uint8) cystatus: Standard API return values.

Return Value	Description
CYRET_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: None

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 paramater. 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_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

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 just a macro define of the WritePixel function.

LCD_SegStat_READ_PIXEL

A macro define of the 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

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_SegStat_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_SegStat_1_Start();
   LCD_SegStat_1_WritePixel(SEG0,1);
   LCD_SegStat_1_ReadPixel(SEG0);
   LCD_SegStat_1_ClearDisplay();
   LCD_SegStat_1_Stop();
}
```

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. Additioanl display APIs are generated based on the type and number of Display Helpers defined.

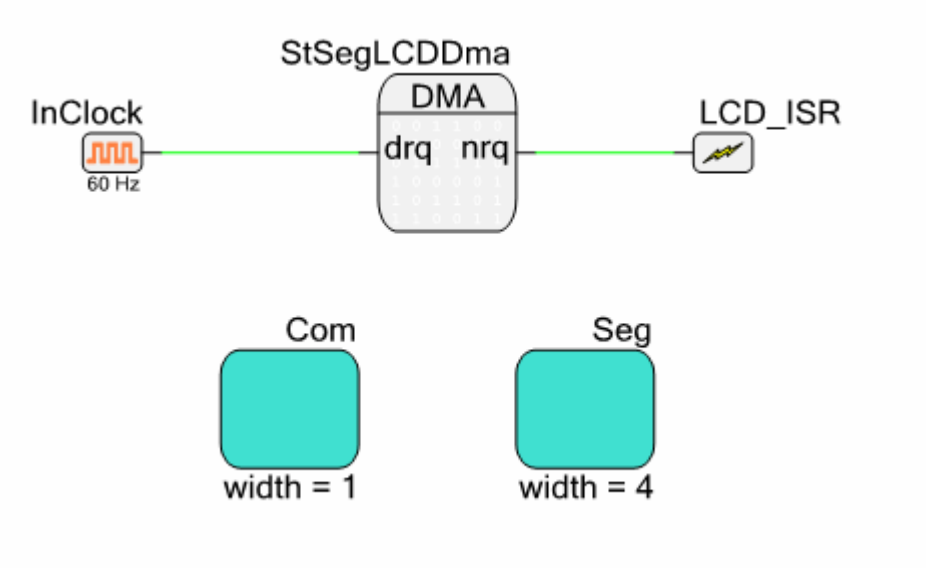
Default Configuration

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

- 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.

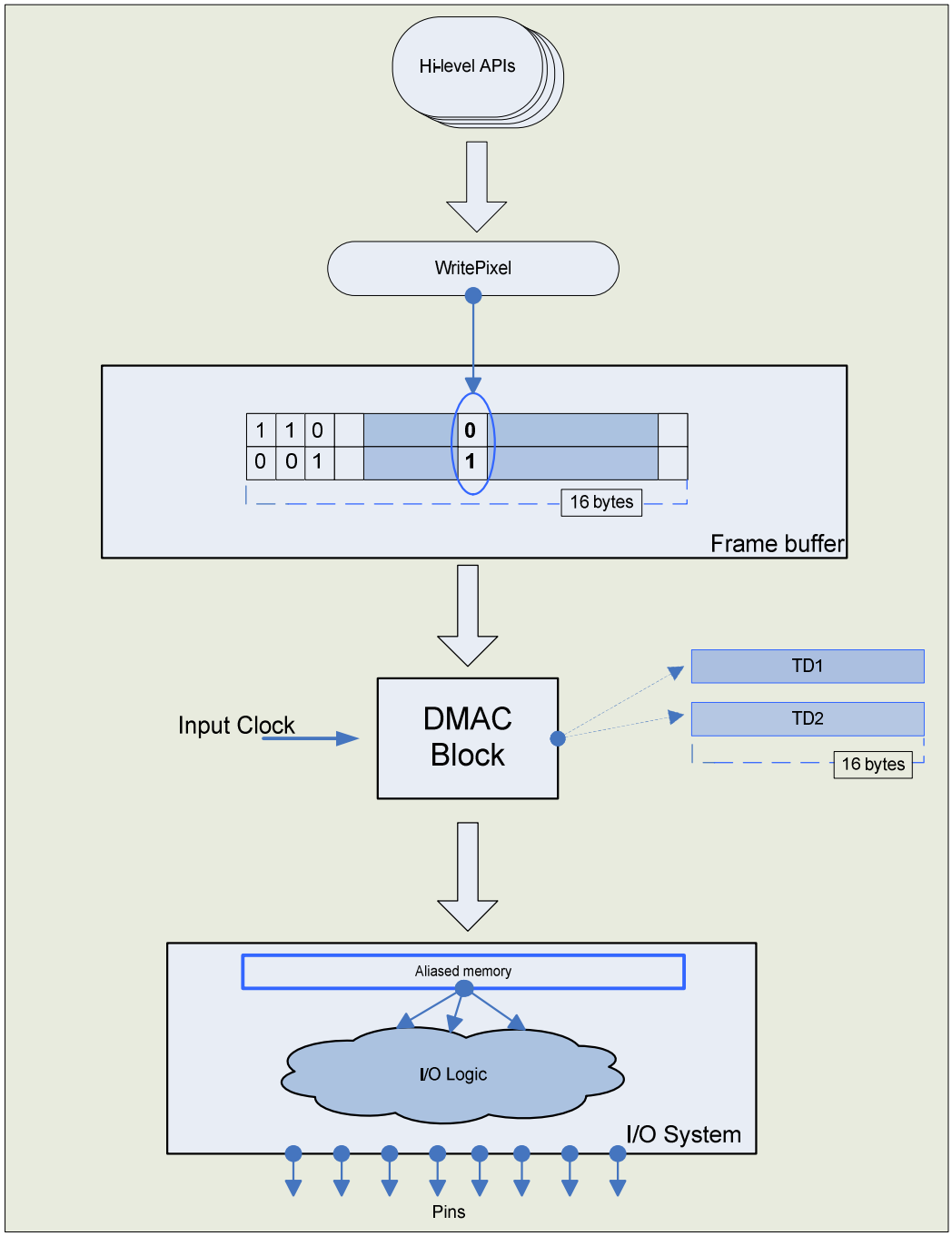
Block Diagram and Configuration

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.



The following figure shows the functional representation of the Segment LCD – Static component.

Top Level Architecture



Registers

None.

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
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	
1.20.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.
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, 2009-2010. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress’ product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.