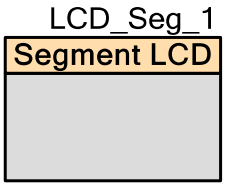
Segment LCD (SegLCD)

1.20



Features

- 2 to 768 pixels or symbols
- 1/3, 1/4 and 1/5 bias supported
- 10 to 150 Hz refresh rate
- Integrated bias generation between 2.0V and 5.2V 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; 5×7 or 5×8 dot matrix; and bar graph calculation routines.

General Description

The Segment LCD (LCD_Seg) component can directly drive 3.3V and 5.0V 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

PRELIMINARY

When to use a Segment LCD

Use the Direct Segment Drive LCD component when you need to directly drive 3.3V or 5.0V 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.

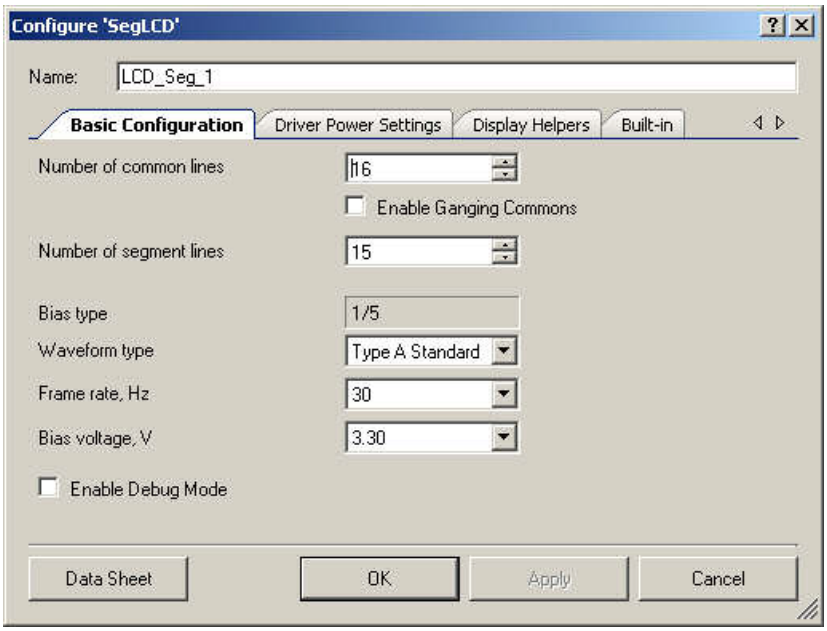
Input/Output Connections

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.

Basic Configuration Tab



Number of Common Lines

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

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.

Number of Segment Lines

Defines the number of segment signals required by the display. The range of possible values is from (2 to 62) - Number Of Common Lines. The default is 8.

Bias Type

This value is read only. It is automatically set by the **Number of Common Lines** selected. It 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. The range of possible values is from 10Hz to 150Hz. The default is 30Hz.

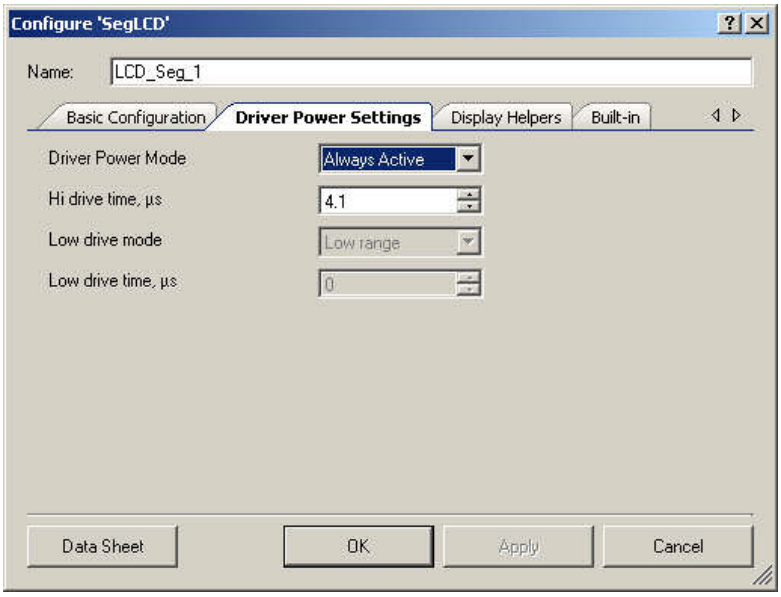
Bias Voltage

This determines the bias voltage level for the LCD DAC. The range of possible values is from 2V to 5.2V. The default is 3.3V.

Enable Debug Mode

If enabled, a DebugPort is added to the component, which allows you to look at the signals that drive the LCD Drivers. It is useful for seeing the Hi and Lo drive times, for example.

Driver Power Settings Tab



Driver Power Mode

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

- "AlwaysActive": LCD DAC will be always be turned on
- "LowPower": LCD DAC will be turned off between voltage transitions

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

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 to its minimum value. 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
```

Low Drive Mode

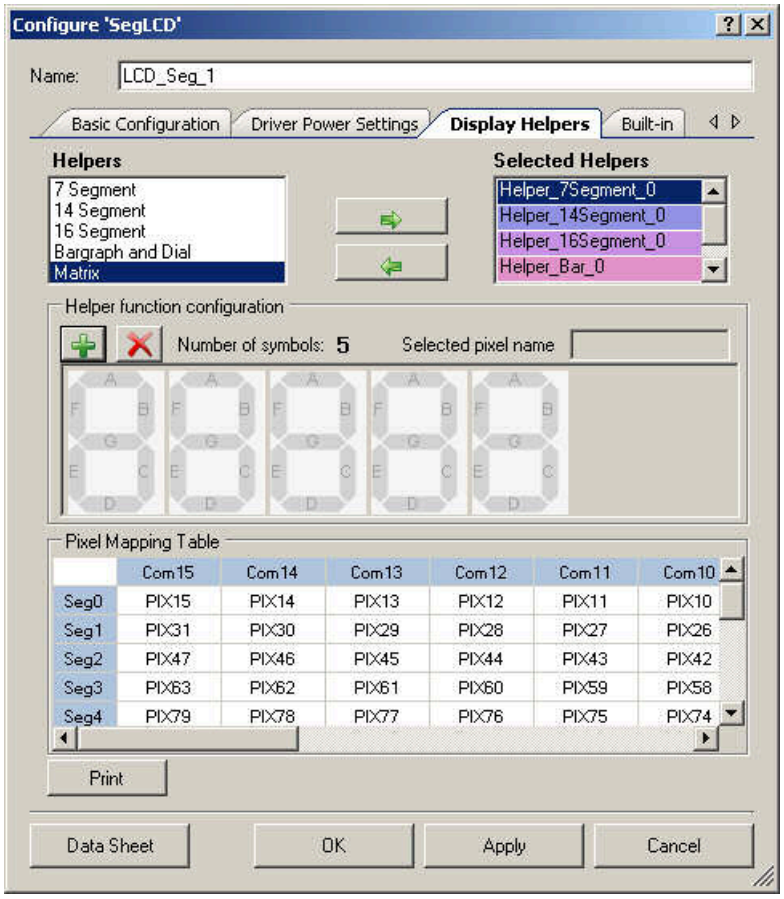
This parameter is available when you set **Driver Power Mode** to "Low Power." Two Low Drive Mode values are available:

- "Low range" – activates Low Drive Mode
- "High range" – activates second low drive high current mode (Lo2).

Low Drive Time

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

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

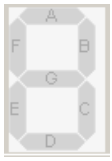
You may add one or more helpers to the **Selected Helpers** list by selecting the desired helper type in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to support the new helper, it will not be added. To delete a helper, select it in the **Selected Helpers** list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

Note: 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.

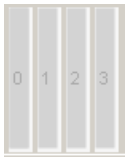
- **7 Segment Helper –** This helper may be 1 to 5 digits in length and can display either hexadecimal digits 0 to F or decimal 16-bit unsigned integer (uint16) values. A decimal point is not supported by the helper functions.



- **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



- **Dot Matrix Helper –** This helper supports up to eight character elements. The component supports 5×7 or 5×8 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.

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

The LCD_Seg component uses two internal clocks and does not require an external clock. Once the component is placed, the two clocks are automatically dedicated to the LCD component. The first clock generates the refresh rate frequency and the second generates a 100 KHz clock for the Low Drive buffers.

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	?	?	?	0	?	?	?

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_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_Seg_Stop	Disables the LCD component and associated interrupts and DMA channels. Does not clear the frame buffer
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	This function sets the bias level for the LCD glass to one of 128 values. Changing the bias level affects the LCD contrast.
LCD_Seg_WriteInvertState	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 page buffer
LCD_Seg_ReadInvertState	This function returns the current value of the display invert state: normal or inverted
LCD_Seg_ClearDisplay	This function clears the display and associated frame buffer RAM.
LCD_Seg_WritePixel	This function sets or clears a pixel based on PixelState. The Pixel is addressed by a packed number.

Function	Description
LCD_Seg_ReadPixel	This function reads the state of a pixel in the frame buffer. The Pixel is addressed by a packed number.
LCD_Seg_SetAwakeMode	This function sets the LCD component to the Awake mode. This is the default mode of the LCD component and is automatically set by the LCD_Seg_Start() function. This function must be called immediately after the device transitions from one of the low power modes to active operation.
LCD_Seg_SetSleepMode	This function puts the LCD component into Sleep mode. This function must be called prior placing the device into a low power mode if LCD component operation is required in the low power mode.
LCD_Seg_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_Seg_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_Seg_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_Seg_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_Seg_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_Seg_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.
LCD_Seg_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

Function	Description
LCD_Seg_PutCharDotMatrix_n	This function displays a character on an array of dot matrix alphanumeric character display elements. Multiple dot matrix 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 dot matrix display element is defined in the component customizer.
LCD_Seg_WriteStringDotMatrix_n	This function displays a null terminated character string on an array of dot matrix alphanumeric character display elements. Multiple dot matrix 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 dot matrix 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" suffix in the function name.

uint8 LCD_Seg_Start (void)

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

Parameters: None

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

Return Value	Description
CYRET_BAD_PARAM	One or more function parameters were invalid
CYRET_SUCCESS	Function completed successfully

Side Effects: None

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: None

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.

Value	Description
NORMAL_STATE	Normal display
LCD_INVERTED_STATE	Inverted display

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

Return Value	Description
CYRET_BAD_PARAM	One or more parameters to the function were invalid
CYRET_SUCCESS	Function completed successfully

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.

Return Value	Description
NORMAL_STATE	Normal display
LCD_INVERTED_STATE	Inverted 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.

Value	Description
LCD_Seg_PIXEL_STATE_OFF	Set the pixel to off.
LCD_Seg_PIXEL_STATE_ON	Set the pixel to on.
LCD_Seg_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_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.

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

Side Effects: None

void LCD_Seg_SetAwakeMode(void)

Description: This function sets the LCD component to the Awake mode. This is the default mode of the LCD component and is automatically set by the LCD_Seg_Start() function. This function must be called immediately after the device transitions from one of the low power modes to active operation.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_SetSleepMode(void)

Description: This function puts the LCD component into Sleep mode. This function must be called prior to placing the device into a low power mode if LCD component operation is required in the low power mode. This function insures that the LCD subframe clock will wake the device as required to update the LCD display.

Parameters: None

Return Value: None

Side Effects: None

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.

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

Return Value: None

Side Effects: None

void LCD_Seg_WriteBargraph_n(uint8 location, uint8 mode)

Description: This function displays an 8-bit integer Location on a 1 to 255 segment bar graph (numbered left to right).The bar graph may be any user-defined size between 1 and 255 segments. A bar graph may also be created in a circle to display rotary position. The customizer Display Helpers facility must be used to define the pixel set associated with the bar graph display 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_WRITE_PIXEL

A macro define of the WritePixel function.

LCD_Seg_READ_PIXEL

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
- 100 Hz refresh rate
- Always Active power mode
- No display helpers are defined. Default API generation will not include functions for any of the supported display elements.

Custom Configuration

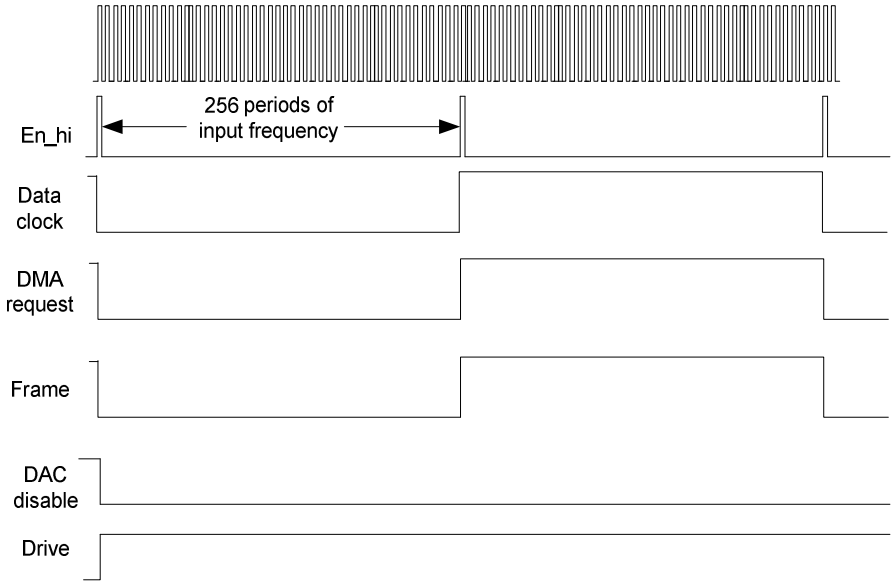
A key feature of the Segment LCD component is flexible support for LCDs with different characteristics and layouts.

Driver Power Modes

Always Active Mode

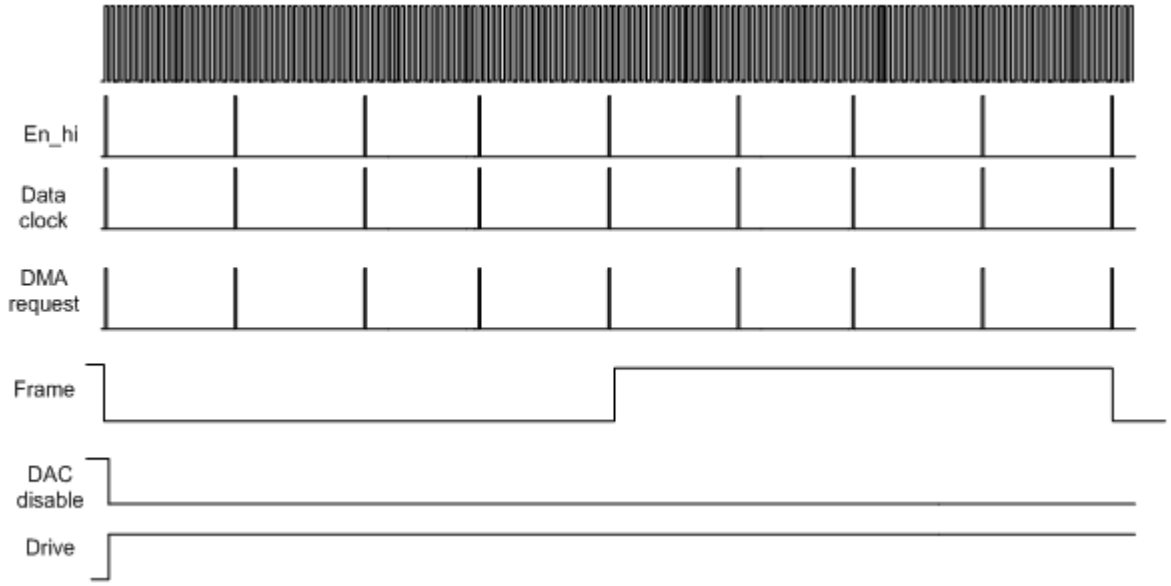
In this use model, the LCD is driven throughout the entire frame. This means that the LCD DAC is powered and the internal signal drive is asserted high whenever the component is enabled.

The following shows waveforms for UDB-generated (internal) signals of the LCD_Seg component for Always Active mode (Type A):



Note Refer to "Timing Calculations" below for more information.

The following shows waveforms for UDB-generated signals of the LCD_Seg component for Always Active mode (Type B):

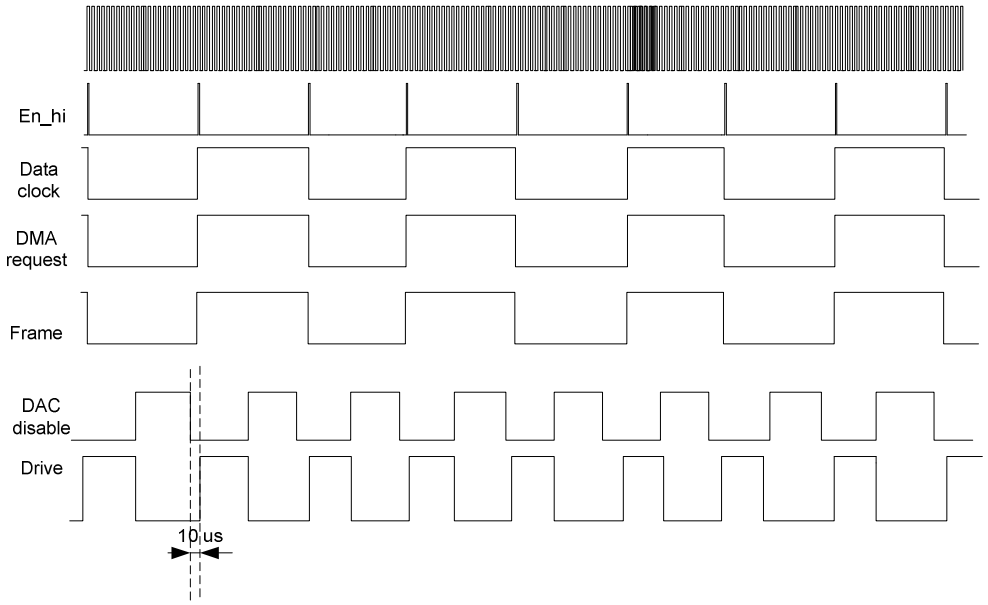


Signals shown are for the ¼ multiplex ratio case.

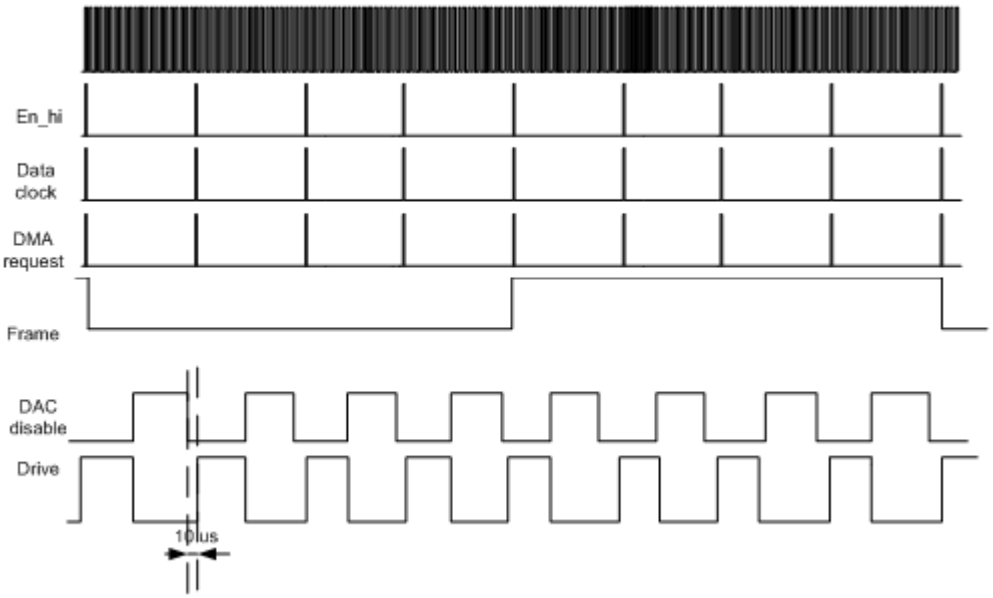
Low Power Mode

In this use model the LCD is actively driven only at voltage transitions and the LCD System analog components are powered down between voltage transitions.

The following shows waveforms for UDB-generated signals of the LCD_Seg component for Transition Active mode (Type A):



The following shows waveforms for UDB-generated signals of the LCD_Seg component for Transition Active mode (Type B):

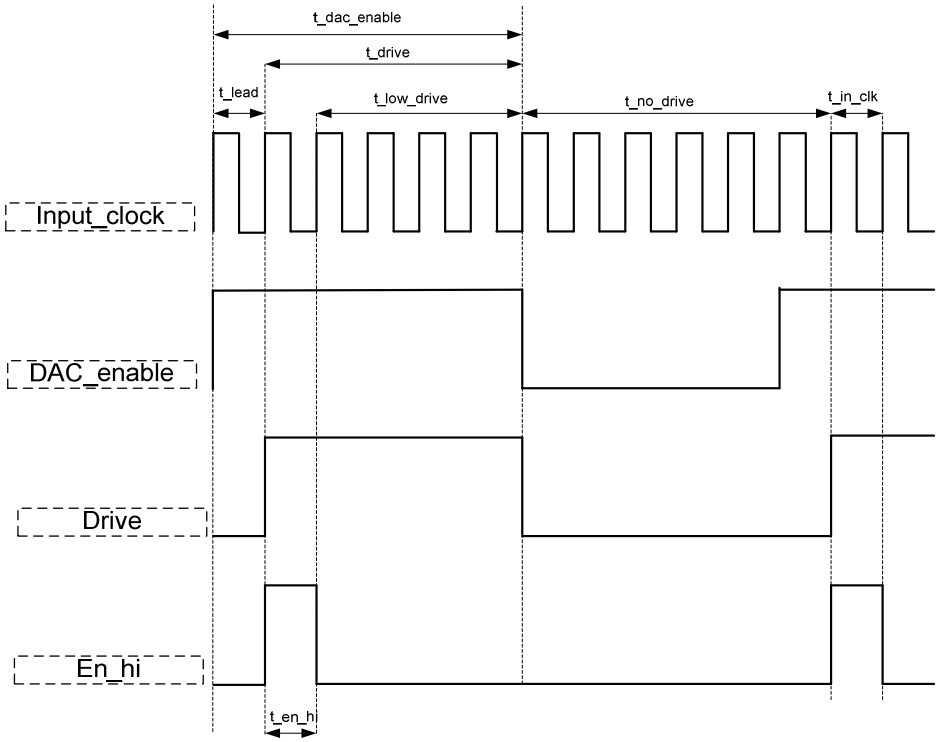


Signals shown are for the ¼ multiplex ratio case.

Timing Calculations

The following diagram and table show the timing information for the UDB generated signals. In the diagram, only three internal signals are represented. The other signals can be derived from the previous diagrams. The following diagram is based on Low Power mode and a Type B waveform.

The DAC_disable signal is generated by inverting the internal DAC_enable signal.



Parameter	Time	Description	Calculation
Input clock frequency	-	Input clock frequency value (set automatically by customizer)	f_in = f_frame_rate * N_common_lines * 256 * 2
Input clock period	t_in_clk	Specifies period of input clock	t_in_clk = 1/(f_in)
Hi Drive time	t_en_hi	Specifies period of time HiDrive will be active.	t_en_hi = t_in_clk
	t_lead	Specifies the time.	10 µs
	-	Hi Drive inactive time	t_low_drive (Always Active) t_low_drive + t_no_drive (Low Power)
	t_drive	Specifies drive time	t_drive = t_en_hi + t_low_drive

Parameter	Time	Description	Calculation
	t_drive	Specifies drive time	t_drive = t_en_hi + t_low_duty_cycle
	t_low_drive	Specifies Low Drive time	t_low_drive = t_drive - t_en_hi
	t_no_drive	Specifies the time when drive signal will be asserted LOW	t_no_drive = t_in_clk * 256 - t_drive
	t_dac_enable	Specifies the time during which LCD DAC will be turned on (only for Low Power mode)	t_dac_enable = t_drive + t_lead

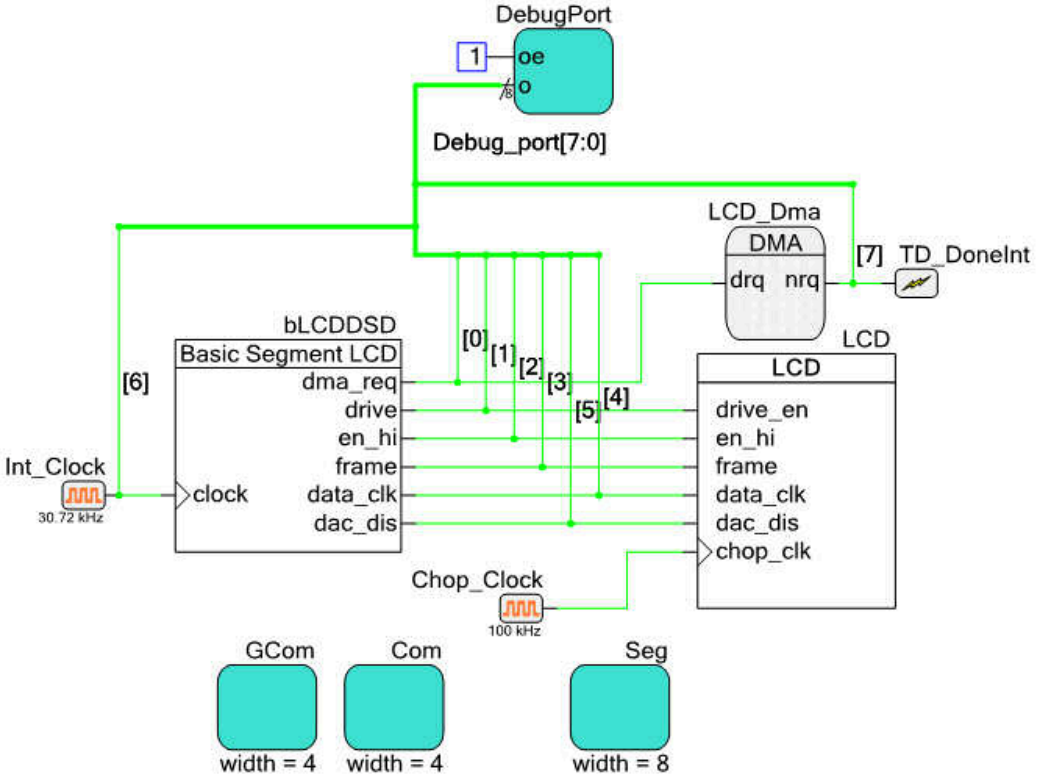
User-Specific Configuration

Signal timing can be adjusted by changing the timing parameters in the component customizer.

Hi Drive Time

By default t_en_hi = t_in_clk or one cycle of input clock. This value is automatically calculated by the customizer. You can increase this time up to a maximum value of HiDriveTimemax= HiDriveTimemin * 254. The Hi Drive Time value will be increased in increments corresponding to 1 cycle of the input clock. This will result in the active time for the en_hi signal being extended.

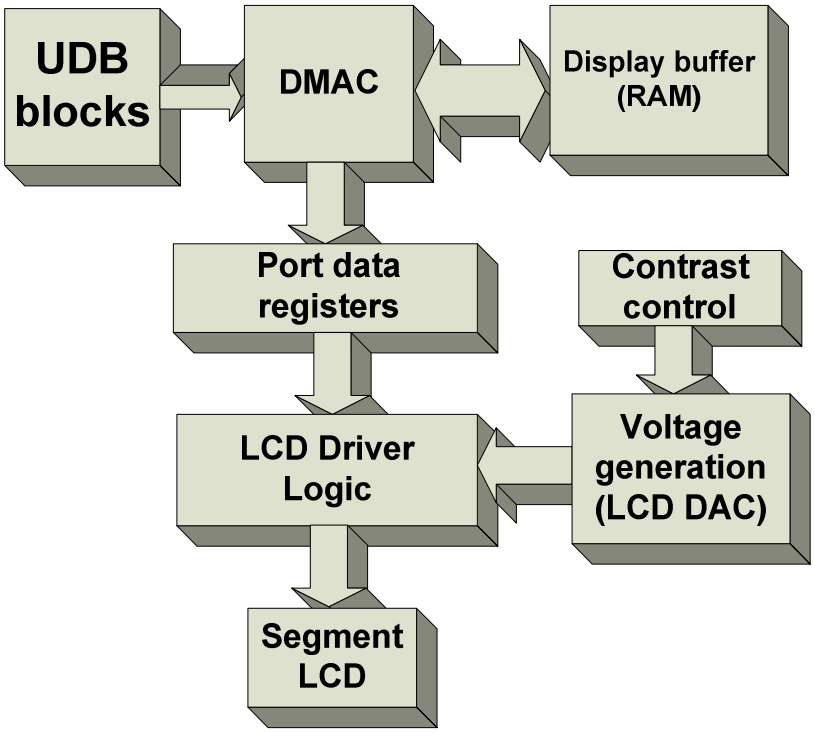
Block Diagram and Configuration



This 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, two LCD ports, one digital port, an ISR component and two clocks.

- 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)** is 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 signalssegments is limited to 48 pins wide. .
- The **DebugPort** is used only for debug purposes. This component is removed by default.

Top Level Architecture



Registers

LCD_Seg_CONTRAST_CONTROL

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	contrast level						

contrast level: bias voltage level described above.

LCD_Seg_LCDDAC_CONTROL

Bits	7	6	5	4	3	2	1	0
Value	reserved					DAC disable	bias select	

bias select: Selects bias.

DAC disable: Disables LCD DAC if contrast control is not needed.

LCD_Seg_DRIVER_CONTROL

Bits	7	6	5	4	3	2	1	0
Value	reserved					invert	lo2	sleep mode

sleep mode: This bit sets the sleep mode for Segment LCD.

lo2: Enables / Disables the high-current mode of loDrive mode of the LCD DRIVER block

invert: If set inverts all data in on the Segment LCD.

LCD_Seg_CONTROL

Bits	7	6	5	4	3	2	1	0
Value	reserved						control reset	clock enable

clock enable: This bit enables generation of all internal signals described in above sections.

control reset: This bit performs initial reset of the digital portion of the component.

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

Parameter	Description	Conditions	Min	Typical	Max	Units
	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 returns an error if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
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.