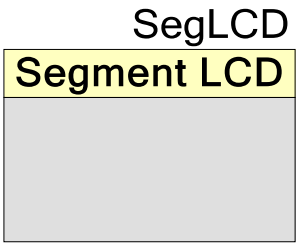
Segment LCD (SegLCD)

1.0



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
- Configurable low power LCD drivers for long battery life and operation during sleep
- 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
- User-defined pixel or symbol map with optional 7 segment, 14 segment, 16 segment, 5×7 or 5×8 dot matrix, and bar graph calculation routines.

General Description

The Segment LCD (SegLCD) 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 for 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 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
- 1-255 element bar graphs

The LCD is configurable to optimize power in portable applications and provides display updates in sleep modes.

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 provide LCD direct drive hardware resources.

Use the Static Drive Segment LCD component when you need to directly drive 3.3V or 5.0V LCD glass at multiplex ratios of 1x. The Static Drive Segment LCD component does not require any LCD specific hardware and may be used on devices that do not support LCD segment drive hardware.

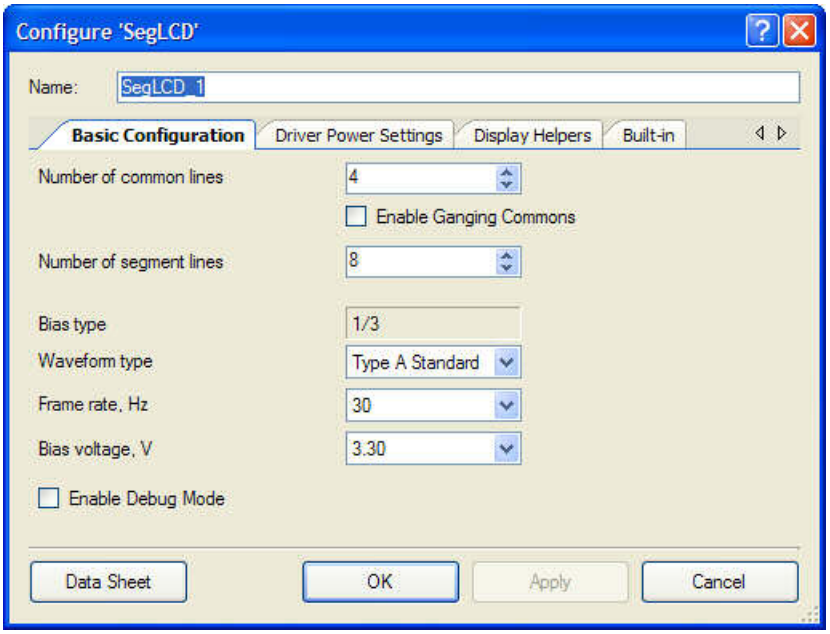
Input/Output Connections

Not applicable

Parameters and Setup

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

Basic Configuration Tab



NumCommonLines(uint8)

Defines the number of common lines in the user-defined display (default is 4). If the number of common lines selected is between the standard 2-16, then use the next highest multiplex ratio and bias ratio but skip the unused subframes.

Enable Ganging Commons

Select this check box to gang PSoC pins to drive common signals.

NumSegmentLines(uint8)

Defines the number of segment lines in the user-defined display. Possible values are in range from 2 to 62 NumCommonLines. The default is 8.

BiasType(uint8)

Read only, automatically set by the number of common lines selected. Determines the proper bias mode for the set of common and segment lines. The default is 1/3.

WaveformType(enum)

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

FrameRate(uint8)

Determines refresh rate of the display. Possible values are in range from 10Hz to 150Hz. The default is 30Hz.

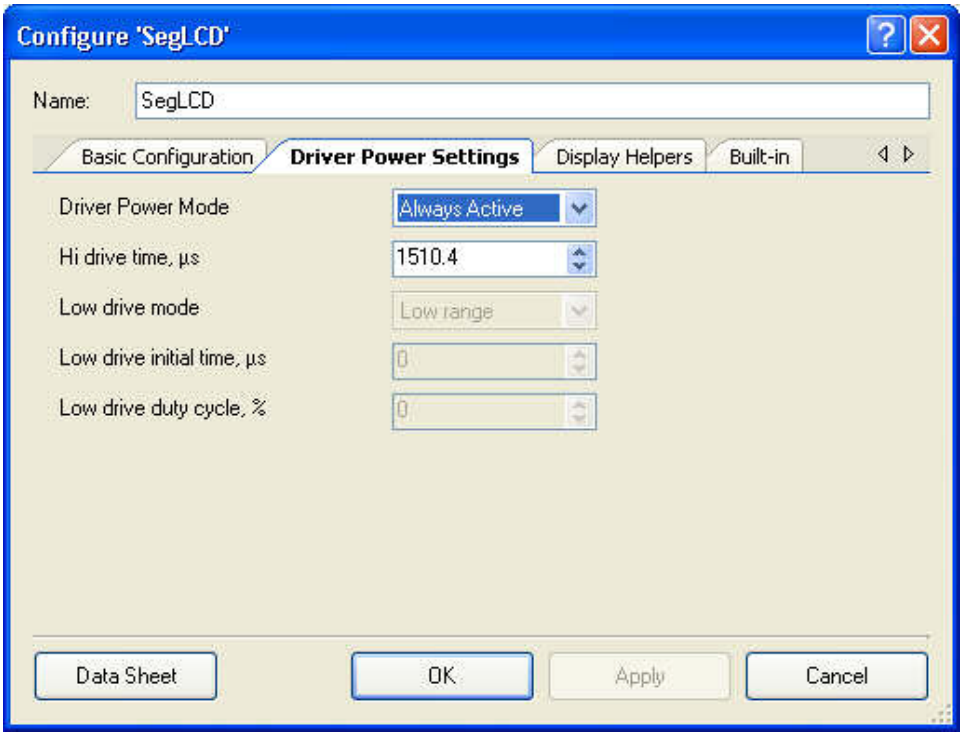
BiasVoltage(uint8)

Determines bias voltage level for the LCDDAC block. Possible values are in range from 2V to 5.2V, default is 3.3 V.

The customizer uses this value to calculate a level value that is set in the LCDDAC during the initialization process.

Note: Following parameter values will be correct when you operating at 5.2 V.

Driver Power Settings Tab



DriverPowerMode (enum)

The Driver Power Mode parameter defines the power mode of component. This value is an enumerated type – “Waveform_Type” – and can be set to any of the following values:

- “AlwaysActive”: LCD DAC will be always turn on
- “LowPower”: LCD DAC will be turn off between voltage transitions

HiDriveTime (uint8)

This parameter defines the time HIDrive mode will be active during each voltage transaction.

Note Because of internal implementation when you change Frame Rate, Number of Common lines or Waveform Type parameter the HiDriveTime will also be changed to it minimum value in current configuration. Minimum value if HiDriveTime is different for different sets of Frame Rate, Number of Common lines and Waveform Type. The maximum value of HiDriveTime in current configuration is:

```
HiDriveTimemax= HiDriveTimemin * 254
```

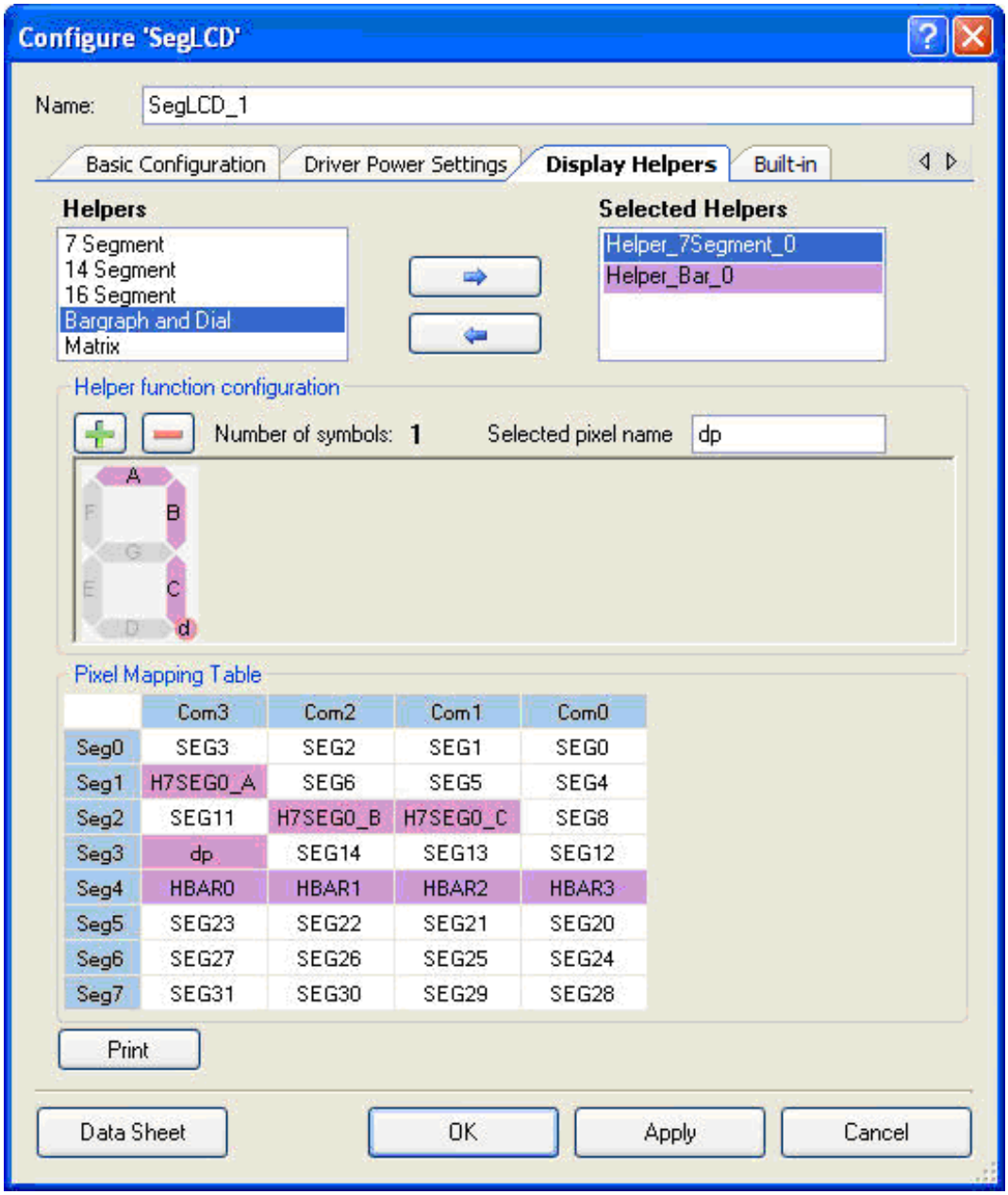
LowDriveMode (uint8)

The LowDriveMode parameter defines the type of Low Drive Mode. Possible values is 0 – “Low range” – activates Low Drive Mode; 1 – “High range” – activates second low drive high current mode (Lo2).

LowDriveTime (uint8)

The LowDriveTime parameter defines the time during which LowDriveMode will be active within 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 (5 x 7 or 5 x 8) or a linear or circular bar graph display. The character based display helpers can be

used to combine multiple display symbols in one display helper to create multi-character display elements.

Helpers / Selected Helpers

You may add one or more helpers to the **Selected Helpers** list by selecting one in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to manage a 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 and segment lines. So it is important to set the number of common and segment lines first. If you do want to change the number of common and segment lines, you will have to remove the helpers.

The order in which the **Selected Helpers** appear in the list has meaning. The first **Selected Helper** of a specific type added first is marked with index 0, the next one of the same type is marked with index 1, and so on. If a **Selected Helper** is removed from the list, the following indexes move up, but retain their index number. If you add another index, it will obtain the first available index.

Helper Function Configuration

This section of the dialog allows you to configure a helper, which includes adding and removing symbols to the helper as well as renaming pixels.

1. Select a helper from the **Selected Helpers** list.
2. Click the [+] or [–] 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 symbols supported by the component. If the number of available pins is not enough to manage a new symbol, it will not be added.
3. To rename a pixel, select it in the LCD image and the name will display in the **Selected pixel name** field.

   The name includes two parts: a prefix and a unique name. The prefix includes a display number, helper type, and symbol number. It is common for all pixels in the symbol. The last part is a unique name of the pixel in the symbol.

   If a prefix is not changed, the pixel name shown on the symbol includes only the unique name. If else, the full name would be displayed. The names of all pixels must be unique.

The **Print** button prints a pixel mapping table.

Pixel Mapping Table

The pixel mapping table is a representation of the Frame Buffer. For correct work of API functions added by Helper functions you need to assign pixels from Helper Function Configuration to pixels in the Pixel mapping table. In other words you need to locate helper

function pixel in the frame buffer. You will need a datasheet of your LCD glass also to make correct pin assignment. To assign pixels just select desired pixel in the Helper Function Configuration panel and drag it to the Pixel Mapping Table and drop in the position defined in LCD glass datasheet.

7 Segment helper

The 7 Segment Helper may be 1 to 5 digits in length and can display either hexadecimal digits 0 to 15 (0 to F) or 16-bit unsigned integer (uint16) values. The decimal point is not provided by the helper function. The decimal point, if required, should be handled in software. APIs are provided for each helper. Refer to API section for more information.

14 Segment helper

The 14 Segment Helper may be up to 20 characters in length. The 14 segment helper displays a single ASCII character or a null terminated string. Possible values are standard ASCII printable characters (with codes from 0 to 127). APIs are provided for each helper. Refer to API section for more information.

16 Segment helper

The 16 Segment Helper may be up to 20 characters in length. The 16 segment helper may display possible single ASCII characters 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. APIs are provided for each helper. Refer to API section for more information.

Bargraph and Dial helper

The Bargraph Helper is used for bar graphs and dial indicators with 1 to 255 segments. The bargraph may be a single selected pixel or the selected pixel and all pixels to the left or right of the selected pixel. Possible values are 0-255 with zero representing all pixels off. APIs are provided for each helper. Refer to API section for more information.

Dot Matrix helper

The Dot Matrix Helper generates up to 8 5×7 or 5×8 characters. Longer strings of characters are created by placing two or more dot matrix helpers in a row. The helper displays a single ASCII character or a null terminated string.

The dot matrix helper has pinout constraints to reduce code overhead. Rather than use lookup tables for arbitrary common and segment drivers the dot matrix helper must use 7 or 8 sequential common drivers for the dot rows and 5 to 40 sequential segment drivers for the dot columns. Supported characters are the same as the Hitachi HD44780 standard character set. APIs are provided for each helper. Refer to API section for more information.

Clock Selection

Segment LCD 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 input frequency and the second generates a 100 KHz clock for the Low Drive buffers.

Placement

The SegLCD 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. UCB resources are automatically placed in the UDB array during the project generation process.

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	?	?	?

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 "SegLCD_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 "SegLCD".

Function	Description
SegLCD_Start	Starts the LCD component and enables any required interrupts, DMA channels frame buffer and hardware. Does not clear the frame buffer SRAM if previously defined.

Function	Description
SegLCD_Stop	Disables the LCD component and disables any required interrupts and DMA channels. Automatically blanks the display to avoid damage from DC offsets. Does not clear the frame buffer
SegLCD_ EnableInt	Enables the LCD interrupt/s. Not required if Start called
SegLCD_DisableInt	Disables the LCD interrupt. Not required if Stop called
SegLCD_SetBias	This function sets the bias level for the LCD glass to one of up to 128 values. The actual number of values is limited by the Analog supply voltage Vdda as the bias voltage can not exceed Vdda. Changing the bias level affects the LCD contrast.
SegLCD_WriteInvertState	This function inverts the display based on InvertState. The inversion occurs in hardware and no change is required to the display RAM in the page buffer
SegLCD_ReadInvertState	This function returns the current normal or inverted state of the display
SegLCD_ClearDisplay	This function clears the display RAM of the frame buffer.
SegLCD_WritePixel	This function sets or clears a pixel based on PixelState in a frame buffer. The Pixel is addressed by a packed number. User code can also directly write the frame buffer RAM to create optimized interactions.
SegLCD_ReadPixel	This function reads a pixels state in a frame buffer. The Pixel is addressed by a packed number. User code can also directly read the frame buffer RAM to create optimized interactions.
SegLCD_SetAwakeMode	This function is the default mode of the LCD component and is automatically set by the SegLCD_Start() fuction. This function must be called immediately after the device transitions from one of the low power modes to active operation.
SegLCD_SetSleepMode	This function must be called just prior to the device entering a low power mode if LCD operation must continue in the low power mode. This function insures that the LCD subframe clock will wake the device as required to update the LCD display.
SegLCD_Write7SegDigit_n	This function displays a digit on a 7 segment display element. Digits can be hexadecimal values in the range of 0-9 and A-F. A 7 segment display helper must have been configured in the customizer to define which display segments make up the 7 segment element(s). Multiple, 7 segment displays can be created in the frame buffer and are addressed through the index (n) in the function name. This function is only available if a 7 segment display element is defined in the component customizer.
SegLCD_Write7SegNumber_n	This function displays an integer value on a 1 to 5 digit 7 segment display. You must define what portion of the displays segments make up the 7 segment display portion In the customizer. Multiple, separate 7 segment displays can be created in the frame buffer and are addressed through the index (n) in the function name. Function/s only included if component 7 segment customizer defines the 7 segment option. Sign conversion, sign display, decimal points and other custom features must be handled by application specific user code.

Function	Description
SegLCD_WriteBargraph_n	This function displays an integer value on a linear or circular bar-graph The bar graph may be any user-defined size between 1 and 255 segments using a customizer display helper.
SegLCD_PutChar14Seg_n	This function displays a character on a 14 segment alphanumeric character display. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.
SegLCD_WriteString14Seg_n	This function displays a null terminated character string on a set of 14 segment alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.
SegLCD_PutChar16Seg_n	This function displays a character on a 16 segment alphanumeric character display. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.
SegLCD_WriteString16Seg_n	This function displays a null terminated character string on a set of 16 segment alphanumeric character display symbols. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.
SegLCD_PutCharDotMatrix_n	This function displays a character on a dot matrix alphanumeric character display. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.
SegLCD_WriteStringDotMatrix_n	This function displays a null terminated character string on a set of dot matrix alphanumeric character display symbols. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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 SegLCD_Start (void)

Description: Starts the LCD component and enables any required interrupts, DMA channels, frame buffers, and hardware. Does not clear the frame buffer SRAM if previously defined. Sign conversion, sign display, decimal points and other custom features must be handled by application specific user code.

Parameters: None

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

void SegLCD_Stop(void)

Description: Disables the LCD component and disables any required 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 SegLCD_EnableInt(void)

Description: Enables the LCD interrupts. Not required if SegLCD_Start is called.

Parameters: None

Return Value: None

Side Effects: None

void SegLCD_DisableInt(void)

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

Parameters: None

Return Value: None

Side Effects: None

void SegLCD_SetBias(uint8 BiasLevel)

Description: This function sets the bias level for the LCD glass to one of up to 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 SegLCD_WriteInvertState(uint8 InvertState)

Description: This function inverts the display based on InvertState. The inversion occurs in hardware and no change is required to the display RAM in the page buffer

Parameters: (uint8) InvertState: Sets the invert state of the display. Symbolic names for the invert state are provided, and their associated values are shown here:

Return 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 SegLCD_ReadInvertState(void)

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

Parameters: None

Return Value: (uint8) InvertState: Sets the invert state of the display. Symbolic names for the invert state are provided, and their associated values are shown here:

Return Value	Description
NORMAL_STATE	Normal display
LCD_INVERTED_STATE	Inverted display

Side Effects: None

void SegLCD_ClearDisplay(void)

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

Parameters: None

Return Value: None

Side Effects: None

uint8 SegLCD_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.

(uin8) 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
PIXEL_STATE_OFF	Set the pixel to off.
PIXEL_STATE_ON	Set the pixel to on.
PIXEL_STATE_INVERT	Invert the pixel's current state.

Return Value: (uint8) Status returns the standardized return value for pass or fail on a range check of the byte address and multiplex row number. No check is performed on bit position.

Side Effects: None

uint8 SegLCD_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 SegLCD_SetAwakeMode(void)

Description: This function is the default mode of the LCD component and is automatically set by the SegLCD_Start() fuction. 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 SegLCD_SetSleepMode(void)

Description: This function must be called just prior to the device entering a low power mode if LCD operation must continue 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 SegLCD_Write7SegDigit_n(uint8 Digit, uint8 Position)

Description: This function displays a 4-bit Hex digit in the range of 0-9 and A-F 7 segment display. You must define what portion of the displays segments make up the 7 segment display portion in the component customizer. Multiple, separate 7 segment displays can be created in the frame buffer and are addressed through the index (n) in the function name. Function/s only included if component 7 segment customizer defines the 7 segment option.

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 defined display does not contain a digit in the Position then the digit will not be displayed.

Return Value: None

Side Effects: None

void SegLCD Write7SegNumber_n(uint8 Value, uint8 Position, uint8 Mode)

Description: This function displays a 16-bit integer value on a 1 to 5 digit 7 segment display. You must define what portion of the displays segments make up the 7 segment display portion In the customizer. Multiple, separate 7 segment displays can be created in the frame buffer and are addressed through the index (n) in the function name. Function/s only included if component 7 segment customizer defines the 7 segment option.
Sign conversion, sign display, decimal points and other custom features must be handled by application specific user code.

Parameters: (uint8) 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 contains fewer digits then the Value requires for display for 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 SegLCD_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. You must define what portion of the displays segments make up the bar-graph portion. Multiple, separate bar graph displays can be created in the frame buffer and are addressed through the index (n) in the function name. Function/s only included if component bar-graph customizer defines the 7 segment option

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 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 bargraph display mode.

Value	Description
0	Only the Location segment is turned on
1	The Location segment 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 2-254 segments to the right to create wide indicators.

Return Value: None

Side Effects: None

void SegLCD_PutChar14Seg_n(uint8 Character, uint8 Position)

Description: This function displays an 8-bit char on an array of alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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

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

Return Value: None

Side Effects: None

void SegLCD_WriteString14Seg_n(*uint8 Character, uint8 Position)

Description: This function displays an 8-bit null terminated character string on an array of alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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 defined display contains fewer characters then the string requires for display, the extra characters will not be displayed.

Return Value: None

Side Effects: None

void SegLCD_PutChar16Seg_n(uint8 Character, uint8 Position)

Description: This function displays an 8-bit char on an array of alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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

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

Return Value: None

Side Effects: None

(void) SegLCD_WriteString16Seg_n(*uint8 Character, uint8 Position)

Description: This function displays an 8-bit null terminated character string on an array of alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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 defined display contains fewer characters then the string requires for display, the extra characters will not be displayed.

Return Value: None

Side Effects: None

void SegLCD_PutCharDotMatrix_n(uint8 Character, uint8 Position)

Description: This function displays an 8-bit char on an array of alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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 display range, the character will not be displayed.

Return Value: None

Side Effects: None

void SegLCD_WriteStringDotMatrix_n(*uint8 Character, uint8 Position)

Description: This function displays an 8-bit null terminated character string on an array of alphanumeric character displays. You must define what portion of the displays segments make up the alphanumeric display portion in the customizer. Multiple, separate alphanumeric displays can be created in the frame buffer and are addressed through the index (n) in the function name. Functions are only included if the component alphanumeric customizer defines the alphanumeric option.

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 defined display contains fewer characters then the string requires for display, the extra characters will not be displayed.

Return Value: None

Side Effects: None

Defines

SegLCD_COMM_NUM

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

SegLCD_SEG_NUM

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

SegLCD_BIAS_TYPE

Defines the bias type for the user-defined display current configuration of the component.

SegLCD_BIAS_VOLTAGE

Defines default bias voltage level for user-defined display. This value will be set in LCDDAC control Register during Initialization process.

SegLCD_FRAME_RATE

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

SegLCD_WRITE_PIXEL

This is just a macro define of the WritePixel function.

SegLCD_READ_PIXEL

A macro define of the ReadPixel function.

SegLCD_FIND_PIXEL

This macros calculates pixel location in the frame buffer. It uses an information from customizer pixel table and information of physical pins which will be dedicated for the LCD. This macros 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 SegLCD_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>
#include <SegLCD_1.h>

void main()
{
   SegLCD_1_Start();
   SegLCD_1_SetBias(127);
   SegLCD_1_WritePixel(SEG0,1);
   SegLCD_1_ReadPixel(SEG0);
   SegLCD_1_ClearDisplay();
   SegLCD_1_Stop();
}
```

Functional Description

Segment LCD component provides a powerful and very flexible mechanism for driving LCD glasses of different types. The GUI of the customizer gives you access to a lot of customization parameters of the component. The API Generator provides you with the access to the different types of APIs that will be generated based on your selections in the “Display Helpers” section o the GUI. The APIs provide required capabilities in LCD's tuning. For example they provide contrast regulation and general operation.

Default Configuration

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

- 4 common lines
- 8 Segment lines, with common lines this provides access to 24 or fewer pixels
- 100 Hz refresh rate
- Always active power mode
- No helpers are chosen. This means no APIs will be provided for a specific type of display.

Custom Configuration

The main feature of the Segment LCD components is that it works with different kinds of LCDs. Users can select any type of display selecting the proper helper function in the customizer.

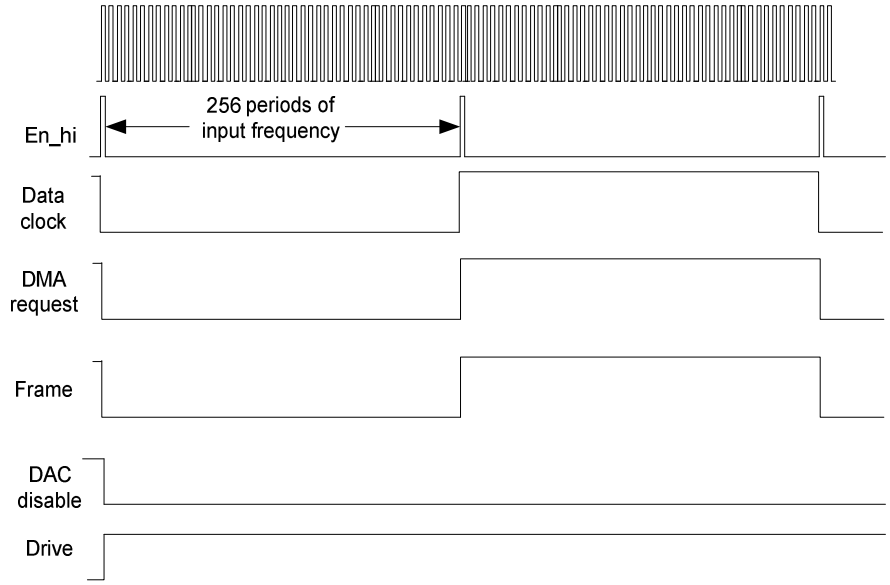
Note: Only the one instance of the component can be used in the project. If you create two instances of the component you will get a placement error or a warning because both will use hardware LCDDAC block.

Driver Power Modes

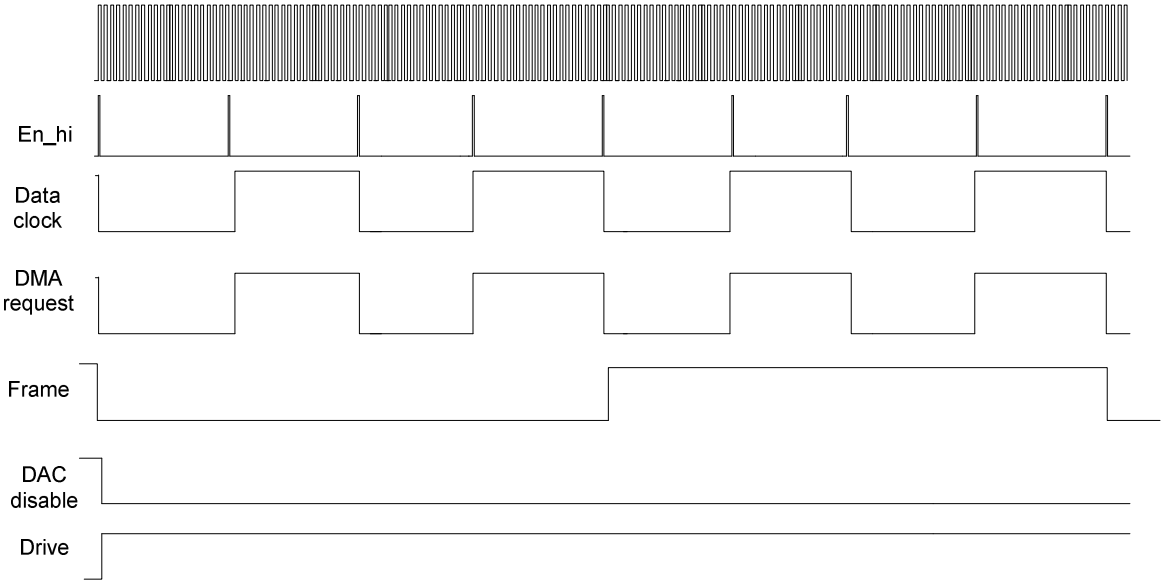
Always Active Mode

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.

Waveforms for UDB-generated (internal) signals of Segment LCD component for Always Active mode (Type A) are presented below:



Waveforms for UDB-generated signals of Segment LCD component for Always Active mode (Type B):

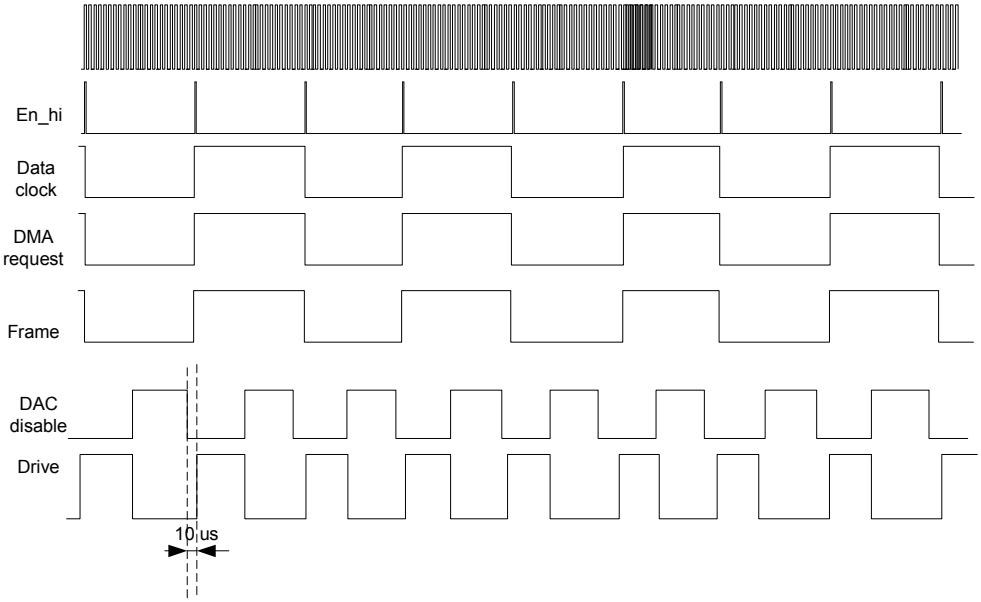


The frame signal is presented for the case of a ¼ multiplex ratio.

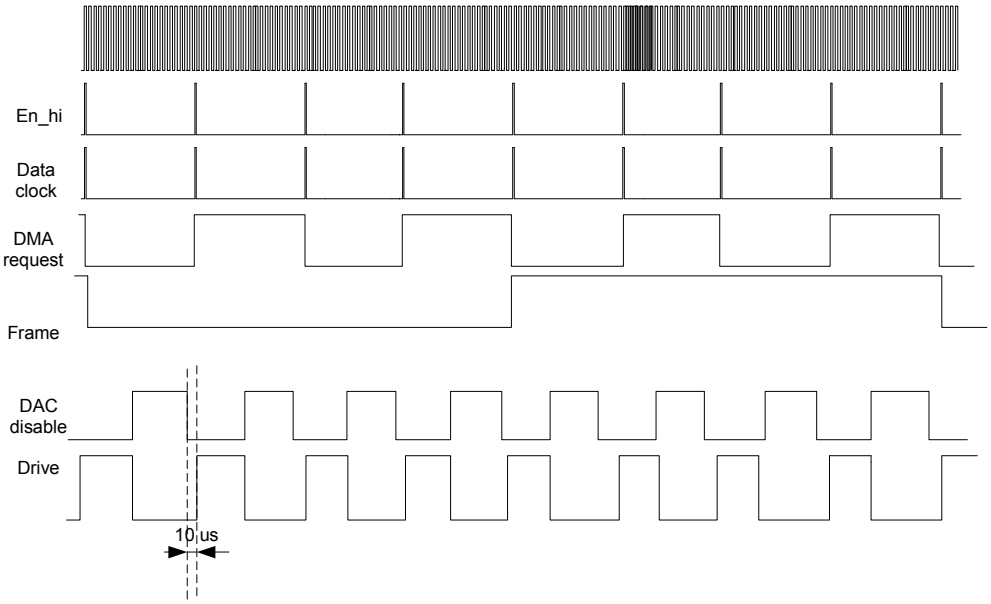
Low Power Mode

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.

Waveforms for UDB-generated signals of Segment LCD component for Transition Active mode (Type A) are presented below:



Waveforms for UDB-generated signals of Segment LCD component for Transition Active mode (Type B):

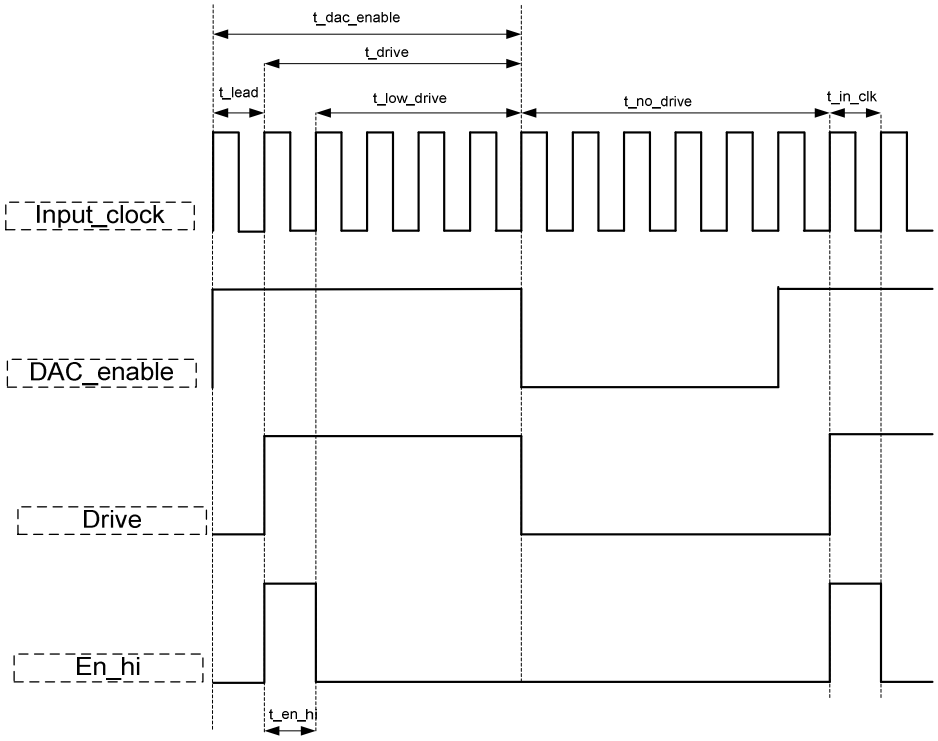


The frame signal is presented for the case of a ¼ multiplex ratio.

Timing Calculations

The diagram and table below show timing for UDB generated signals. In the diagram, only three internal signals are represented. The other signals can be calculated from the previous diagrams. The following diagram represents 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 equation
Input clock frequency	-	Input clock frequency value (set automatically by customizer)	f_in = f_frame_rate * N_common_lines * 256
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
	low_duty_cycle	LoDrive duty cycle from the customizer	50%

Parameter	Time	Description	Calculation equation
Low Drive Duty Cycle	t_low_duty_cycle	Defines low drive active duty cycle	t_low_duty_cycle = (low_duty_cycle/100)*(256 - t_en_hi)
	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_dac_enable
	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

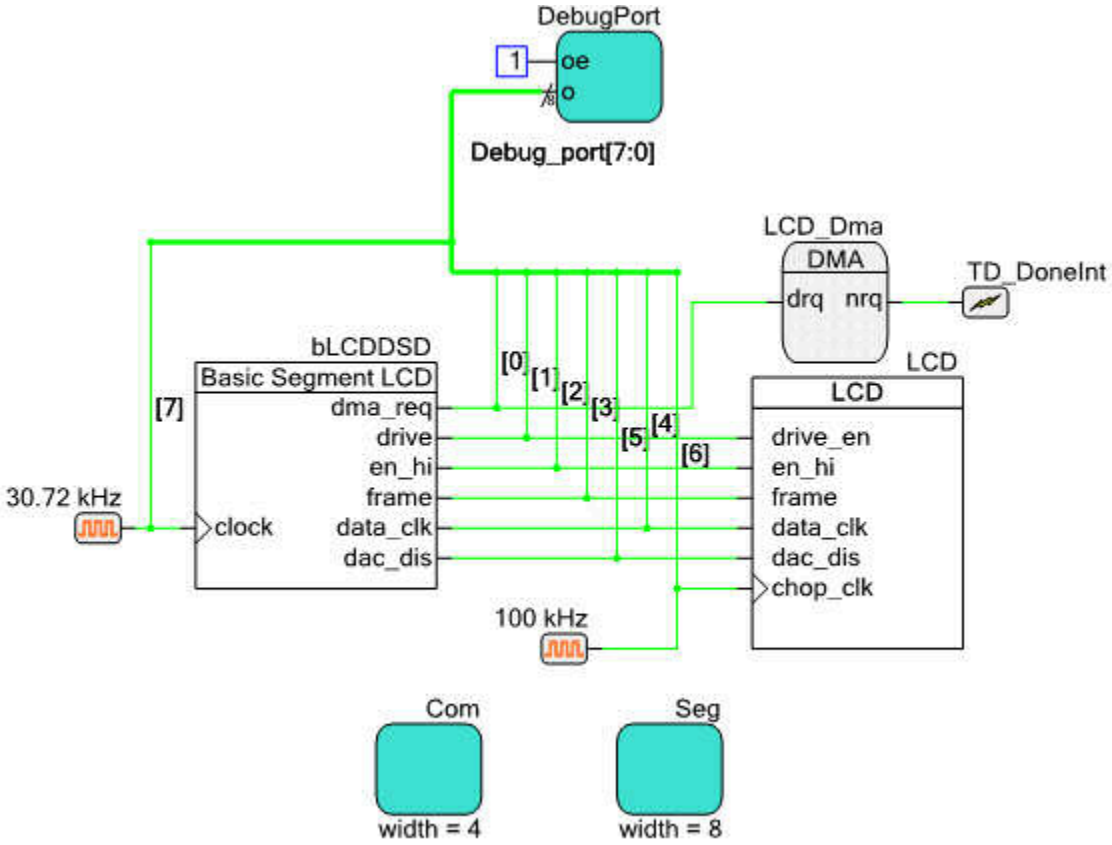
User-Specific Configuration

With the component customizer you can adjust the timing by changing parameters.

Hi Drive time

By default t_en_hi = t_in_clk or one cycle of input clock. The customizer will automatically calculate the time for this parameter. You can increase this time up to maximum value of 100 µs. The time will be increased each time with the value of t_in_clk. Each time you increment the value in the customizer it will extend the duration of the active state of the en_hi signal by the 1 cycle of the input clock.

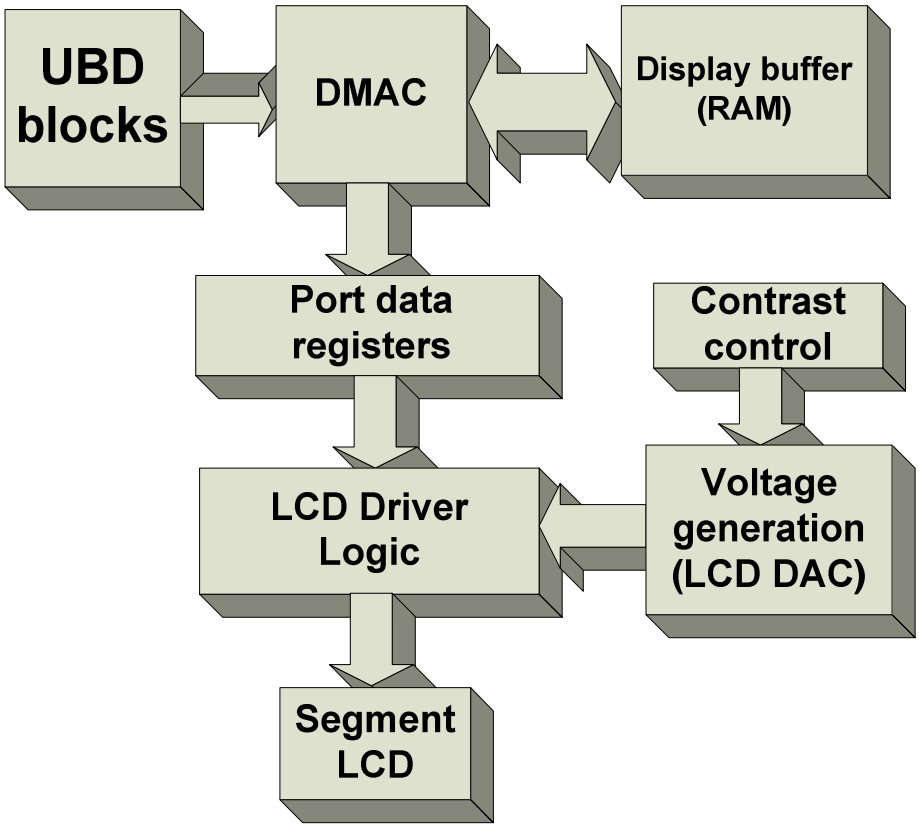
Block Diagram and Configuration



The diagram represents a schematic of an implementation of the Segment LCD component. It consists of a Basic Segment LCD component, an LCD Control block (LCD) component, DMA, two LCD Ports, one digital port, and an ISR component. Clock components provide the required clocks.

The **Basic Segment LCD** is responsible for generating the proper timing signals for the LCD Port and DMA components. **DMA** is used to transfer data from the frame buffer to the LCD data registers through the aliased memory area. The **LCD Control Block** handles all DSI routing required. This block also provides all required register names as defines in cyfitter.h. The **LCD Port** is used to map logical pins to physical. There are two instances of the LCD Port; one for common lines and one for segment lines. The LCD Port for commons is limited to a 16-pin width and the LCD Port for segments is limited to 48. The **DebugPort** is used only for debug purposes. This component is removed by default.

Top Level Architecture



Registers

SegLCD_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.

SegLCD_LCDDAC_CONTROL

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

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

bias select: Selects bias.

SegLCD_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.

SegLCD_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.

SegLCD_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.

Constants

There are several constants defined for the control registers as well as some of the enumerated types. Most of these are described above for the Control and Status Register. However there are more constants needed in the header file to make all of this happen. Each of the register definitions requires either a pointer into the register data or a register address. Because of multiple byte orders of the compilers, it is required that the CY_GET_REGX and CY_SET_REGX macros are used for register accesses.

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
1.0.b	Added information to advertize component 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.
1.0.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.