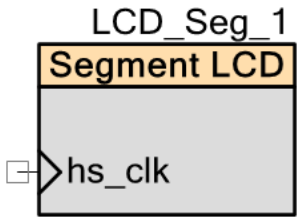
PSoC 4 Segment LCD (SegLCD)

1.30



Features

- Digital Correlation and PWM at 1/2, 1/3, 1/4 and at 1/5 bias modes supported
- Drive STN/TN LCD glass with up to eight COMs
- 30 to 150 Hz refresh rate
- Supports both type A (standard) and type B (low power) waveforms
- Pixel state of the display may be inverted for a negative image
- User-defined pixel or symbol map with optional 7-, 14-, or 16-segment character; 5x7 or 5x8 dot matrix, and bar graph calculation routines.
- Operation in Deep Sleep Mode from the ILO
- All-digital contrast control using "Dead Period" digital modulation

General Description

The Segment LCD component can directly drive a variety of LCD glass at different voltage levels with multiplex ratios. This component provides an easy method of configuring the PSoC device to drive your custom or standard glass. You can use the component to drive STN or TN LCD glass with up to 4 COMs for PSoC 4100/PSoC 4200 and PSoC 4100 BLE/PSoC 4200 BLE device families, and up to 8 COMs for PSoC 4100M/PSoC 4200M/PSoC 4200L device families.

Each LCD pixel/symbol may be either on or off. The Segment LCD component also has advanced support to simplify these 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- to 255-element bar graphs

When to Use a Segment LCD

Use the Direct Segment Drive LCD component when you need to directly drive a variety of LCD glass at different voltage levels with multiplex ratios. The Direct Segment Drive LCD component requires that the target PSoC device supports LCD direct drive.

Input/Output Connections

This section describes the various input and output connections for the Segment LCD component. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

hs_clk – Input

This is the source for the high speed LCD clock. This input always requires connection of the external clock component. The source for the high speed LCD clock is a clock divided from the HFCLK. The recommended input frequency for the HFCLK is in the MHz range. Setting the frequency too high causes the entire display to partially darken. Setting it too low causes desired root-mean-square (RMS) voltage across the LCD segments attenuation.

The external high speed LCD clock is intended for High Speed LCD mode. The divided HFCLK clock should always be connected to the hs_clk component input to be able to switch between Low Speed and High Speed LCD modes at run-time by using LCD_Seg_SetSpeedMode() API.

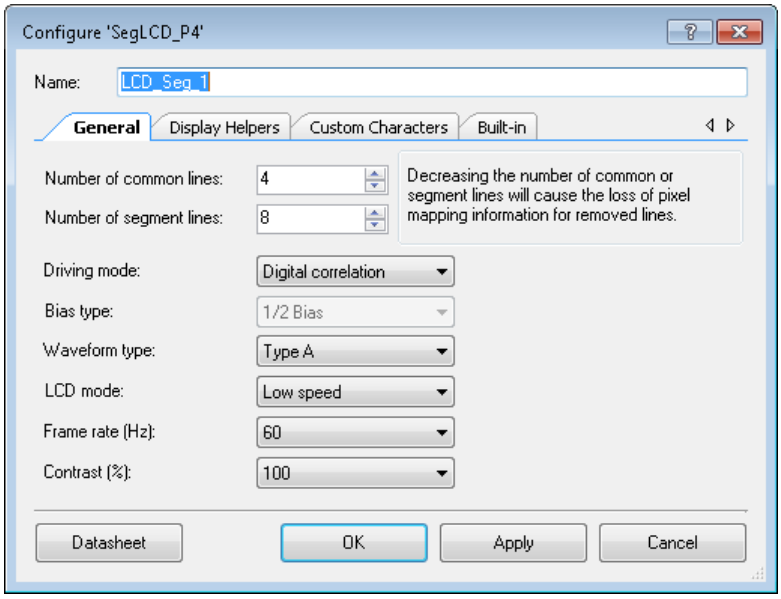
For more details, refer to LCD Mode, Clock Selection, and Functional Description sections.

There are no other visible connections for the component on the schematic canvas. The component does however include the pin connections to the segments and commons and these can be assigned to physical pins using the Design-Wide Resources Pin Editor.

Parameters and Setup

Drag a Segment LCD component onto your design and double click it to open the **Configure** dialog. The **Configure** dialog contains several tabs with different types of parameters to set up the Segment LCD component.

Basic Configuration Tab



Number of common lines

Defines the number of common signals required by the display. The default is 4. The range of possible values include:

- from 1 to 4 for PSoC 4100/PSoC 4200 and PSoC 4100 BLE/PSoC 4200 BLE device families
- from 1 to 8 for PSoC 4100M/PSoC 4200M/PSoC 4200L device families

Number of segment lines

Defines the number of segment signals required by the display. The default is 8. The range of possible values is from 2 to the max Number of available LCD pins, where:

max Number of available LCD pins = (max Number of LCD pins - Number of common lines)

Refer to the device datasheet for the number of LCD pins available for a particular device.

Driving Mode

This parameter defines the driving mode configuration: Digital correlation or PWM. Refer to the Functional Description section for more information.

Bias type

This value determines the proper bias mode for the set of common and segment lines. The following Bias type settings are available: 1/2, 1/3, 1/4 Bias and 1/5 Bias. 1/4 Bias and 1/5 Bias options are supported only in High speed mode. Refer to the Functional Description section for more information.

Waveform type

This determines the waveform type: Type A – each frame addresses each COM pin only once with a balanced (DC = 0) waveform (default) or Type B - each frame addresses each COM pin twice in sequence with a positive and negative waveform that together are balanced (DC = 0).

LCD Mode

This parameter defines the High Speed/Low Speed mode selection. The following LCD Mode settings are available:

- Low Speed: used for Active and Deep Sleep power modes. The clock source for Low Speed mode is the 32 kHz ILO. LCD block connected to ILO clock internally, so no external clock component connection required for the operation in Low Speed mode.
- High Speed: used for Active power mode. The clock source for High Speed mode is a divided HFCLK clock, which should always be connected to the hs_clk input.

Preference between Low Speed and High Speed modes should be based on required power mode, Frame rate, and Contrast parameters values. High Speed mode provides more variety of Frame rate and Contrast values, while Low Speed mode could be limited for corner cases due to implementation specifics.

LCD modes could be switched at run-time by using LCD_Seg_SetSpeedMode() API.

Refer to the Clock Selection and Functional Description sections for more information about mode selection.

Frame rate (Hz) [1]

This determines the refresh rate of the display. The range of possible values is selectable from 30 Hz to 150 Hz, in increments of 10. The default is 60 Hz.

---

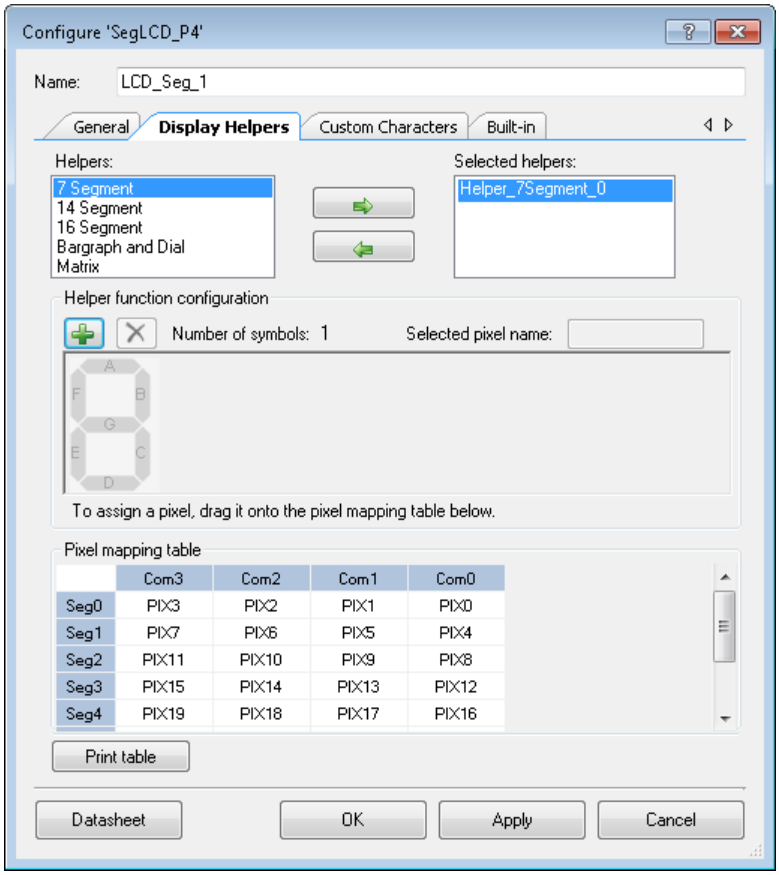
[1] For certain cases, for example at high input frequencies the list of possible values for Frame rate and Contrast may be limited due to the limited divider size (for Low Speed mode 8 bit and for High Speed mode 16 bit) to set the sub-frame and the dead time.

Contrast (%) [1]

This parameter defines the contrast control using "Dead Period" digital modulation. The range of possible values is selectable from 10% to 100% in 10% increments. The default is 100%.

Note If the contrast is set too high, the LCD segments will appear on (dark) at all times. If the contrast is set too low, the LCD segments will appear off at all times. In Low Speed mode, the available contrast settings will be affected by the selected frame rate. Higher frame rates will allow lower contrast settings.

Display Helpers Tab



The Display Helpers tab allows 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

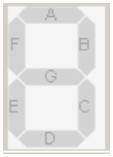
You can add one or more helpers to the **Selected Helpers** list by selecting the desired helper type in the **Helpers** list and clicking the right-arrow button. If there are not enough pins to support the new helper, it is not added. To delete a helper, select it in the **Selected Helpers** list and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

Note It is important to set the number of common and segment lines for the component before you define any display helpers. Remove defined display helpers before you change the number of common or segment lines, because you can lose helper configuration information. If you attempt to change the number of common or segment lines, a warning is displayed indicating that helper pixel mapping configuration can be lost.

The order in which the **Selected Helpers** appear in the list is significant. By default, the first helper of a given type added to the **Selected Helper** list is named with a 0 suffix, the next one of the same type has a suffix of 1, and so on. If a **Selected Helper** is removed from the list, the remaining helpers are not renamed. When a helper is added, the name uses the lowest available suffix.

APIs are provided for each helper. See the Application Programming Interface section for more information.

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



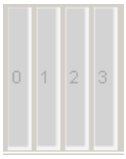
- **14 Segment Helper –** This helper may be up to 20 characters in length. It may display a single ASCII character or a null terminated string. Possible values are standard ASCII printable characters (with codes from 0 to 127).



- **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 of the pixels to the left or right of the specified pixel



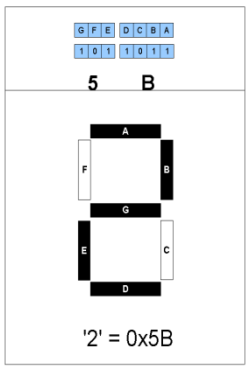
- **Matrix Helper –** This helper supports up to eight character elements. The component supports 5x7 or 5x8 row/column characters. Longer strings of characters can be created by configuring two or more dot-matrix helpers to define adjacent dot-matrix sections of the display. The helper displays a single ASCII character or a null terminated string.

The dot matrix helper has pinout constraints. The dot matrix helper must use seven or eight sequential common drivers for the matrix rows and 5 to 40 sequential segment drivers for the matrix columns. The component supports the standard Hitachi HD44780 character set.

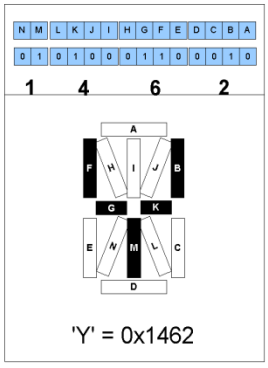
Character Encoding

All high-level Helper APIs have their own look-up tables. The tables include a set of encoded pixel states, which construct a specific character reflection. These examples show how the specific character can be encoded (segment names may be different than shown in the **Configure** dialog).

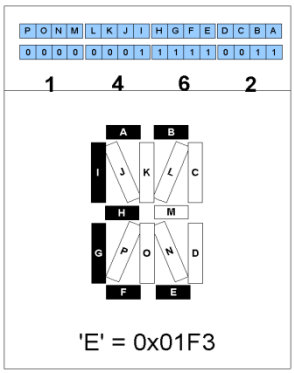
7-Segment Encoding



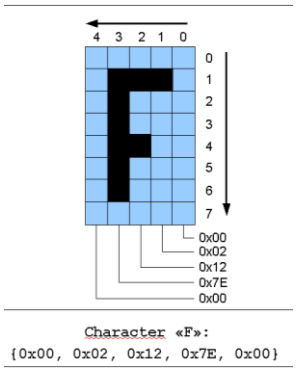
14-Segment Encoding



16-Segment Encoding



Dot-Matrix Encoding



Helper Function Configuration

This section of the dialog allows you to configure a helper; this includes adding or removing symbols to or 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. The pixel name can be renamed to give it a meaningful name for the design. 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 the right upper corner of the **Pixel Mapping Table**.

The default naming for pixels associated with a helper symbol has 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 fourth 7-segment symbol in the project
A	identifies the unique segment within the 7 segment symbol

For default pixel names, only the unique portion of the pixel name is shown on the symbol image. If you modify a pixel name, then the entire name will be shown on the symbol image even if they have a common prefix.

Note All pixel names must be unique.

When a helper function symbol element is assigned to a pixel in the **Pixel Mapping Table** (described below), the pixel assumes the name of the helper symbol element. The helper symbol element name supersedes the default pixel name, but does not replace it. You cannot reuse the default pixel name of pixels that are associated with a helper function.

Pixel Mapping Table

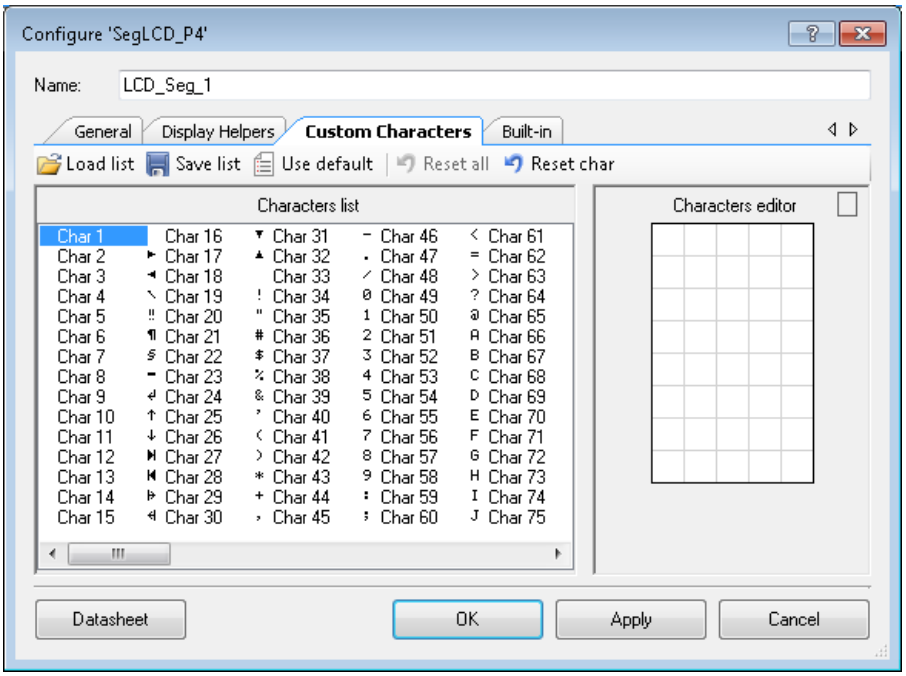
The **Pixel Mapping Table** is a representation of the frame buffer. For the API functions to work properly, each pixel from the **Helper function configuration** must be assigned to a pixel location in the **Pixel Mapping Table**. See the datasheet for your LCD glass for the information you need to 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 the pixel in the table display and entering the desired name. Use this method to name a pixel that is not associated with one of the available helper types.

The **Print table** button creates an image of the **Pixel Mapping Table** for printing.

Custom Characters Tab



This tab allows you to create custom characters for 5x8 dot-matrix displays. You may also use it to store a custom character look-up table as an XML file.

By default, the **Characters List** field contains 255 ASCII characters that have reflection as the standard Hitachi HD44780 character set. You can access and modify any of those characters using the **Character Editor**.

The **Reset char** command allows you to reset unsaved characters to a default reflection. The **Reset all** command brings all unsaved characters to the standard reflection.

After you save your character set, you can save it as an XML file using the **Save list** command. The **Load list** command allows you to load your character list from an XML file. You can go back to a standard character set using the **Use default** option.

Clock Selection

There are two clock sources for the LCD block. These are referred to as the high speed and low speed clocks. The high speed clock is a clock divided off of HFCLK. The low speed is the 32 kHz ILO. The high speed clock should always be connected to the hs_clk component input. The low speed clock is internally connected to the component. A default clock is specified using the customizer via LCD mode parameter (High Speed LCD mode fed by high speed clock and Low Speed LCD mode fed by 32 kHz low speed clock) and this clock may be changed at run-time by using LCD_Seg_SetSpeedMode() API.

Note Generally the recommended input frequency for the high speed clock is in the MHz range.

For large values of the input high speed clock frequency, the required divider for the correct refresh rate generation could be greater than max value of the SUBFR_DIV divider (16 bit for HS mode) and could not be correctly applied by the component customizer.

The value of the input frequency determines the possible values for frame rate and contrast. For certain cases at high input frequencies the list of possible values may be limited, due to limited dividers size (for Low Speed mode 8 bit and for HS mode 16 bit), to set the sub frame and the dead time.

Digital correlation mode can be used with either low speed or high speed clocks. In this mode the outputs change at a small multiple of the frame rate, slow enough that analog distortion should be minimal.

PWM mode uses a square wave output to approximate an analog voltage. This requires the high speed clock and system level tuning of the following parameters:

- Display capacitance (or possibly an additional load cap)
- Series resistance
- High speed clock frequency into the block
- GPIO supply voltage (to a lesser extent)

The display capacitance and series resistor provide a simple RC low pass filter network. The block frequency must be a decade or so above the filter cutoff frequency to control the ripple on the output, setting a practical minimum frequency. Increasing the series resistance provides a lower RC filter cutoff frequency which reduces ripple. It also reduces the effective load capacitance seen by the GPIO, improving rise/fall times and reducing distortion. The GPIO supply voltage has an impact on rise/fall times.

Guidelines for high speed clock frequency selection in PWM mode

Definitions:

F_{CLK} = High speed clock frequency

Bias = one of 1/2, 1/3, 1/4, or 1/5

$F_{PWM} = F_{CLK} * Bias$

F_{PWM} is the maximum toggle frequency of the LCD controlled segment or common output in PWM mode.

F_{DATA} is the maximum toggle rate of the data being passed to the LCD.

Equations:

$$F_{DATA} = \frac{F_{PWM}}{2 * (SUBFR_DIV + 1)}$$

F_{FRAME} is the frame rate of the LCD driver. A typical frame is shown in the Functional description section.

$$F_{FRAME} = \frac{F_{DATA}}{2 * (\#\ of\ commons) + DeadTime}$$

The circuit relies on an RC low-pass filter for proper operation. The components of this are:

$$R_T = R_{GPIO} + R_S + R_{ITO}$$

Where:

- R_{GPIO} is the output impedance of the GPIO output driver
- R_S is an optional series resistance, most often put near the source
- R_{ITO} is the resistance of routing within the LCD display

To estimate the value of R_{GPIO}, we can use the I_{OH}/I_{OL} values from the datasheet. For example, SID59 shows that for VDD = 3V, I_{OH} = 4mA for V_{OH} = -0.6V. We can then calculate R_{GPIO} = 0.6V/4mA = 150 Ω.

The RC Cutoff Frequency is defined as:

$$F_C = \frac{1}{2\pi R_T C_{LCD}}$$

To ensure that the desired root-mean-square (RMS) voltage reaches the LCD display without attenuation:

$$F_{C_MIN} \geq 10 * F_{DATA}$$

To avoid the LCD to see the PWM output toggling, that would cause the entire display to partially darken:

$$F_{PWM} \geq 10 * F_{C_MAX}$$

There is another constraint that sets the maximum F_{PWM} that can be used. Long term exposure to DC average voltage offsets between segment and common lines can cause the display to darken over time. To avoid this it is required that the average bias levels (i.e. 1/3*Vdd) be accurate within 20mV. Requirement for bias error drops below 20mV is the following:

$$F_{PWM} \leq \frac{1}{8 R_{GPIO} C_{TH}}$$

Where: C_{TH} = the equivalent parallel capacitance from the GPIO driver perspective:

$$C_{TH} = \frac{C_{LCD}}{(\pi F_{CLK} C_{LCD} (R_S + R_{ITO}))^2 + 1}$$

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. This table lists and describes the interface to each function. The following sections covers 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."

Functions

Function	Description
LCD_Seg_Start()	Initializes the Segment LCD with default customizer values and enables the Segment LCD.
LCD_Seg_Stop()	Disables the Segment LCD.
LCD_Seg_SetSpeedMode()	Sets High Speed or Low Speed LCD speed mode.
LCD_Seg_SetOperationMode()	Sets PWM or Digital Correlation LCD operation mode.
LCD_Seg_SetBiasType()	Sets bias type for PWM operation mode.
LCD_Seg_SetWaveformType()	Sets the LCD driving Waveform Type.
LCD_Seg_SetContrast()	Sets the contrast control using "Dead Period" digital modulation.

Function	Description
LCD_Seg_WriteInvertState()	Inverts the display based on an input parameter.
LCD_Seg_ReadInvertState()	Returns the current value of the display invert state: normal or inverted
LCD_Seg_ClearDisplay()	Clears the display and associated frame buffer RAM.
LCD_Seg_WritePixel()	Sets or clears a pixel based on PixelState.
LCD_Seg_ReadPixel()	Reads the state of a pixel in the frame buffer.
LCD_Seg_Sleep()	Prepares the component to enter Sleep mode.
LCD_Seg_Wakeup()	Restores component after exiting Sleep mode.
LCD_Seg_SaveConfig()	Saves user data.
LCD_Seg_RestoreConfig()	Restores user data.
LCD_Seg_Init()	Initializes/Restores default Segment LCD configuration.
LCD_Seg_Enable()	Enables the Segment LCD.

void LCD_Seg_Start(void)

Description: Initializes the Segment LCD with default customizer values and enables the Segment LCD.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_Stop(void)

Description: Disables the Segment LCD.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_SetSpeedMode(uint32 mode)

Description: Sets High Speed or Low Speed LCD speed mode.

Parameters: uint32 mode: Sets the mode of the LCD clock speed operation.

Name	Description
LCD_Seg_SPEED_LS	Low Speed mode.
LCD_Seg_SPEED_HS	High Speed mode.

Return Value: None

Side Effects: None

void LCD_Seg_SetOperationMode(uint32 mode)

Description: Sets PWM or Digital Correlation LCD operation mode.

Parameters: uint32 mode: Sets the mode of the LCD operation.

Name	Description
LCD_Seg_MODE_PWM	PWM operation mode.
LCD_Seg_MODE_DIG_COR	Digital Correlation mode.

Return Value: None

Side Effects: None

void LCD_Seg_SetBiasType(uint32 bias)

Description: Sets bias type for PWM operation mode.

Parameters: uint32 bias: Sets the bias type for PWM operation mode.

Name	Description
LCD_Seg_BIAS_1_2	1/2 Bias.
LCD_Seg_BIAS_1_3	1/3 Bias.
LCD_Seg_BIAS_1_4	1/4 Bias. Not supported in Low Speed mode.
LCD_Seg_BIAS_1_5	1/5 Bias. Not supported in Low Speed mode.

Return Value: None

Side Effects: None

void LCD_Seg_SetWaveformType(uint32 type)

Description: Sets the LCD driving Waveform Type.

Parameters: uint32 type: Sets the Waveform Type.

Name	Description
LCD_Seg_TYPE_A	Waveform type A.
LCD_Seg_TYPE_B	Waveform type B.

Return Value: None

Side Effects: None

uint32 LCD_Seg_SetContrast(uint32 contrast)

Description: Sets the contrast control using "Dead Period" digital modulation.

Parameters: uint32 contrast: Sets the contrast for the LCD glass in percentage. Valid range from 10% to 100% in 10% increments. Valid range can be restricted because of dividers size (for Low Speed mode 8 bit and for HS mode 16 bit). For greater frequencies, certain ratios between contrast and frame rate cannot be achieved.

Return Value: Pass or fail based on a validity check of the contrast value.

Name	Description
CYRET_SUCCESS	Function completed successfully.
CYRET_BAD_PARAM	Evaluation of contrast parameter failed.

Side Effects: None

void LCD_Seg_WriteInvertState(uint32 invertState)

Description: Inverts the display based on an input parameter.

Parameters: uint32 invertState: Sets the invert state of the display.

Name	Description
LCD_Seg_STATE_NORMAL	Normal non inverted display.
LCD_Seg_STATE_INVERTED	Inverted display.

Return Value: None

Side Effects: None

uint32 LCD_Seg_ReadInvertState(void)

Description: Returns the current value of the display invert state: normal or inverted.

Parameters: None

Return Value: The invert state of the display.

Name	Description
LCD_Seg_STATE_NORMAL	Normal non inverted display.
LCD_Seg_STATE_INVERTED	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

uint32 LCD_Seg_WritePixel(uint32 pixelNumber, uint32 pixelState)

Description: Sets or clears a pixel based on the input parameter PixelState. The pixel is addressed by a packed number. This packed number comes from the pixel mapping table of Display helpers. The pixel mapping table is used to map the helper function pixel to the actual frame buffer pixel. All pixels have packed number defines provided in the component header file.

Parameters: uint32 pixelNumber: The predefined packed number pointing to the pixel's location in the frame buffer. Each pixel has a define for the pixel number based on the name of the pixel. For example the default define for segment B in the first 7-segment display helper is LCD_Seg_H7SEG0_B.

uint32 pixelState: The pixelNumber specified is set to this pixel state.

Name	Description
LCD_Seg_PIXEL_STATE_ON	Pixel on.
LCD_Seg_PIXEL_STATE_OFF	Pixel off.
LCD_Seg_PIXEL_STATE_INVERT	Pixel invert.

Return Value: Pass or fail based on a range check of the pixelNumber address.

Name	Description
CYRET_SUCCESS	Function completed successfully.
CYRET_BAD_PARAM	Evaluation of pixelNumber parameter failed.

Side Effects: None

uint32 LCD_Seg_ReadPixel(uint32 pixelNumber)

Description: Reads the state of a pixel in the frame buffer. The pixel is addressed by a packed number. This packed number comes from pixel mapping table of Display helpers. Pixel mapping table used to map the helper function pixel to the actual frame buffer pixel. All pixels have packed number defines which are resides in component header file.

Parameters: uint32: pixelNumber: The predefined packed number pointing to the pixel's location in the frame buffer. Each pixel has a define for the pixel number based on the name of the pixel. For example the default define for segment B in the first 7-segment display helper is LCD_Seg_H7SEG0_B.

Return Value: Returns the current status of the PixelNumber specified.

Name	Description
LCD_Seg_PIXEL_STATE_ON	Pixel on.
LCD_Seg_PIXEL_STATE_OFF	Pixel off.
LCD_Seg_PIXEL_UNKNOWN_STATE	Evaluation of pixelNumber parameter failed.

Side Effects: None

void LCD_Seg_Sleep(void)

Description: Prepares the component to enter Sleep mode.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_Wakeup(void)

Description: Restores component after exiting Sleep mode.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_SaveConfig(void)

Description: Saves user data. This API is called by LCD_Seg_Sleep() function.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_RestoreConfig(void)

Description: Restores user data. This API is called by LCD_Seg_Wakeup() function.

Parameters: None

Return Value: None

Side Effects: None

void LCD_Seg_Init(void)

Description: Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call LCD_Seg_Init() because the LCD_Seg_Start() API calls this function and is the preferred method to begin component operation.

Parameters: None

Return Value: None

Side Effects: Block will be stopped to change settings.

void LCD_Seg_Enable(void)

Description: Activates the hardware and begins component operation. It is not necessary to call LCD_Seg_Enable() because the LCD_Seg_Start() API calls this function, which is the preferred method to begin component operation.

Parameters: None

Return Value: None

Side Effects: None

Global Variables

Variable	Description
LCD_Seg_initVar	LCD_Seg_InitVar indicates whether the Segment LCD has been initialized. The variable is initialized to 0 and set to 1 the first time LCD_Seg_Start() is called. This allows the component to restart without reinitialization after the first call to the LCD_Seg_Start() routine. If reinitialization of the component is required, then the LCD_Seg_Init() function can be called before the LCD_Seg_Start() or LCD_Seg_Enable() function.

Optional Helper APIs

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

Function	Description
LCD_Seg_Write7SegDigit_n	Displays a hexadecimal digit on an array of 7-segment display elements.
LCD_Seg_Write7SegNumber_n	Displays an integer value on a 1- to 5-digit array of 7-segment display elements.
LCD_Seg_WriteBargraph_n	Displays an integer location on a linear or circular bar graph.
LCD_Seg_PutChar14Seg_n	Displays a character on an array of 14-segment alphanumeric character display elements.
LCD_Seg_WriteString14Seg_n	Displays a null terminated character string on an array of 14-segment alphanumeric character display elements.
LCD_Seg_PutChar16Seg_n	Displays a character on an array of 16-segment alphanumeric character display elements.
LCD_Seg_WriteString16Seg_n	Displays a null terminated character string on an array of 16-segment alphanumeric character display elements.
LCD_Seg_PutCharDotMatrix_n	Displays a character on an array of dot-matrix alphanumeric character display elements.
LCD_Seg_WriteStringDotMatrix_n	Displays a null terminated character string on an array of dot-matrix alphanumeric character display elements.

Note Each display helper is given a unique number starting at 0. The suffix "n" in the function name must be replaced with the number for the display helper.

void LCD_Seg_Write7SegDigit_n(uint32 digit, uint32 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 to 9 and A to F. The customizer Display Helpers facility must be used to define the pixel set associated with the 7-segment display elements. 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: uint32 digit: Unsigned integer value in the range of 0 to 15 to be displayed as a hexadecimal digit.

uint32 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(uint32 value, uint32 position, uint32 mode)

Description: This function displays a 16-bit integer value on a 1- to 5-digit array of 7-segment 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: uint32 value: The unsigned integer value to be displayed.

uint32 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.

uint32 mode: Sets the display mode.

Value	Description
LCD_Seg_NO_LEADING_ZEROES	No leading zeroes.
LCD_Seg_ LEADING_ZEROES	Leading zeroes are displayed

Return Value: None

Side Effects: None

void LCD_Seg_WriteBargraph_n(uint32 location, int32 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 elements 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: uint32 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.

int32 mode: Sets the bar graph display mode.

Value	Description
0	The 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 to 10	Displays the Location segment and 2 to 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, uint32 position)

Description: This function displays an 8-bit character 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: The ASCII value of the character to display (printable characters with ASCII values 0 to 127)

uint32 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_WriteString14Seg_n(*uint8 character, uint32 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 elements. 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: The pointer to the null terminated character string.

uint32 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: Doesn't clear the display prior to data output. All the locations that weren't affected will remain in their previous pixel states.

void LCD_Seg_PutChar16Seg_n(uint8 character, uint32 position)

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

uint32 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_WriteString16Seg_n(*uint8 character, uint32 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 elements. 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: The pointer to the null terminated character string.

uint32 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: Doesn't clear the display prior to data output. All the locations that weren't affected will remain in their previous pixel states.

void LCD_Seg_PutCharDotMatrix_n(uint8 character, uint32 position)

Description: This function displays an 8-bit character 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 elements. 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.

uint32 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, uint32 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 elements. 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 pointer to the null terminated character string.

uint32 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

Macros

- LCD_Seg_EXTRACT_ROW(pixel) – Calculates the row of the specific pixel in the frame buffer.
  - pixel – the predefined packed number that points to the bit's location in the frame buffer.
- LCD_Seg_EXTRACT_PORT(pixel) – Calculates the byte offset of the specific pixel in the frame buffer.
  - pixel – the predefined packed number that points to the bit's location in the frame buffer.
- LCD_Seg_EXTRACT_PIN(pixel) – Calculates the bit position of the specific pixel in the frame buffer.
  - pixel – the predefined packed number that points to the bit's location in the frame buffer.
- LCD_Seg_FIND_PIXEL(port, pin, row) – This macro calculates pixel location in the frame buffer. It uses information from the customizer pixel table and information about the physical pins that will be dedicated for the LCD. This macro is the base of the pixel mapping mechanism. Every pixel name from the pixel table is defined with a calculated pixel location in the frame buffer. APIs use pixel names to access the respective pixel.
  - port – the port number of the particular LCD Common or Segment line.
  - pin – the pin number of the particular LCD Common or Segment line.
  - row – the frame buffer row number of the particular LCD Common or Segment line.

Sample Firmware Source Code

PSoC Creator has many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

See the "Find Example Project" topic in the PSoC Creator Help for more information.

MISRA Compliance

This section describes the MISRA-C: 2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the System Reference Guide along with information on the MISRA compliance verification environment.

The Segment LCD component has the following specific deviations:

Rule	Rule Class	Rule Description	Description of Deviation(s)
19.7	Advisory	A function should be used in preference to a function-like macro.	The following macros are used for increased performance: LCD_Seg_FIND_PIXEL(), LCD_Seg_EXTRACT_ROW(), LCD_Seg_EXTRACT_PORT(), LCD_Seg_EXTRACT_PIN().

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and the component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

The table below shows all possible configurations of the Segment LCD component. The meaning of the configuration names are:

- **Default**: Low-level API functions set without any high-level helper API
- **7-Segment helper**: Low-level API functions set + 7-Segment helper API

- **14-Segment helper**: Low-level API functions set + 14-Segment helper API
- **16-Segment helper**: Low-level API functions set + 16-Segment helper API
- **Dot Matrix helper**: Low-level API functions set + Dot Matrix helper high-level API
- **Bar Graph helper**: Low-level API functions set + Bar Graph helper high-level API

Configuration	PSoC 4 (GCC)	
	Flash Bytes	SRAM Bytes
Default	1032	16
7-Segment helper	1176	16
14-Segment helper	1428	16
16-Segment helper	1436	16
Dot Matrix helper	2576	16
Bar Graph helper	1264	16

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 provides control of the display and of specific pixels. Additional display APIs are generated based on the type and number of Display Helpers defined.

Note All block configurations can only be modified when the block is disabled. You need to call the LCD_Seg_Stop() function, modify the configuration and then call the LCD_Seg_Enable() function to resume block operations.

Default Configuration

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

- 4 common lines
- 8 segment lines
- Digital Correlation driving mode

- Waveform type A
- Low speed LCD mode
- 60 Hz refresh rate
- 100% contrast
- No display helpers are defined. Default API generation does not include functions for any of the supported display elements.

Custom Configuration

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

Contrast vs. Frame Rate

Some combinations of frame rate and input frequency can restrict the valid contrast range because of limited dividers size (for Low Speed mode 8 bit and for High Speed mode 16 bit). For small values of contrast at small frame rates, the required divider values may be beyond permissible limits of the dividers size. For large High Speed clock frequencies, certain ratios between the contrast and frame rate can't be achieved due to the limited divider size. The component customizer automatically restricts such incorrect combinations. Sub-frame and dead time divider values for High Speed and Low Speed Master Generators are generated by the component customizer and stored in LCD_Seg_dividersHS[] and LCD_Seg_dividersLS[] arrays. For cases when there is a problem with the divider generation, those data arrays are filled with the nearest valid divider values.

LCD Speed Mode Switching

The High Speed and Low Speed Master Generators are essentially duplicates of each other except that the High Speed version has larger frequency dividers to generate the frame and sub-frame periods; this is because the clock of the High Speed block typically has 30-100 times the frequency of the 32 KHz clock fed to the Low Speed block. Switching between High Speed and Low Speed modes via LCD_Seg_SetSpeedMode() function causes the dividers to reconfigure. If there are some restrictions, which are related to certain ratios between contrast and frame rate (see Contrast vs. Frame rate section above), it is possible that switching between High Speed and Low Speed modes via LCD_Seg_SetSpeedMode() function can set new dividers values, which don't give the same contrast value. The divider values generated by the component customizer for High Speed and Low Speed Master Generators attempt to preserve the same frame rate.

Driving Modes

LCD_Seg component supports the following operating modes:

- Digital Correlation
- PWM at 1/2, 1/3, 1/4 or 1/5 Bias

Use Cases:

	Deep Sleep	Sleep/Active	Notes
4 COM, TN Glass	Digital Correlation	PWM, 1/3 Bias	SW configuration is used to switch the mode between deep sleep and active
4 COM, STN Glass	Digital Correlation		No contrast advantage for PWM with STN Glass
8/16 COM, STN	Not supported	PWM 1/4 Bias and 1/5 Bias	Supports only in the High speed LCD mode. The Low speed clock is not fast enough to make the PWM work at high multiplex ratios.

PWM modes driving low capacitance displays may require external resistors.

Conventional LCD Driver Waveforms

Conventional LCD drivers apply waveforms to the COM and SEG electrodes that are generated by switching between multiple different voltages. The following terms are used to define these waveforms:

- Duty: A driver is said to operate in 1/M-th duty when it drives M COM electrodes. Each COM electrode is effectively ‘driven’ 1/M-th of the time.
- Bias: A driver is said to use 1/B-th bias when its waveforms use voltage steps of (1/B)*VDRV.
- Frame: A frame is the length of time needed to address all segments. During a frame the driver cycles through the commons in sequence. All segments receive 0V DC when measured over the length of the entire frame.
- Type-A Waveform: The driver structures the frame into M sub-frames. COMi is addressed in sub-frame i.
- Type-B Waveform: The driver structures the frame into 2M sub-frames. COMi is address in sub-frames i and M+i. The two sub-frames are each other inverse. Type-B waveforms are typically slightly more power efficient because it contains fewer transitions.

The following figures show the conventional waveforms for COM and SEG electrodes for 1/3rd bias and 1/4th duty. Only COM0/COM1 and SEG0/SEG1 are drawn.

Figure 1. Conventional Type-A Waveform Example

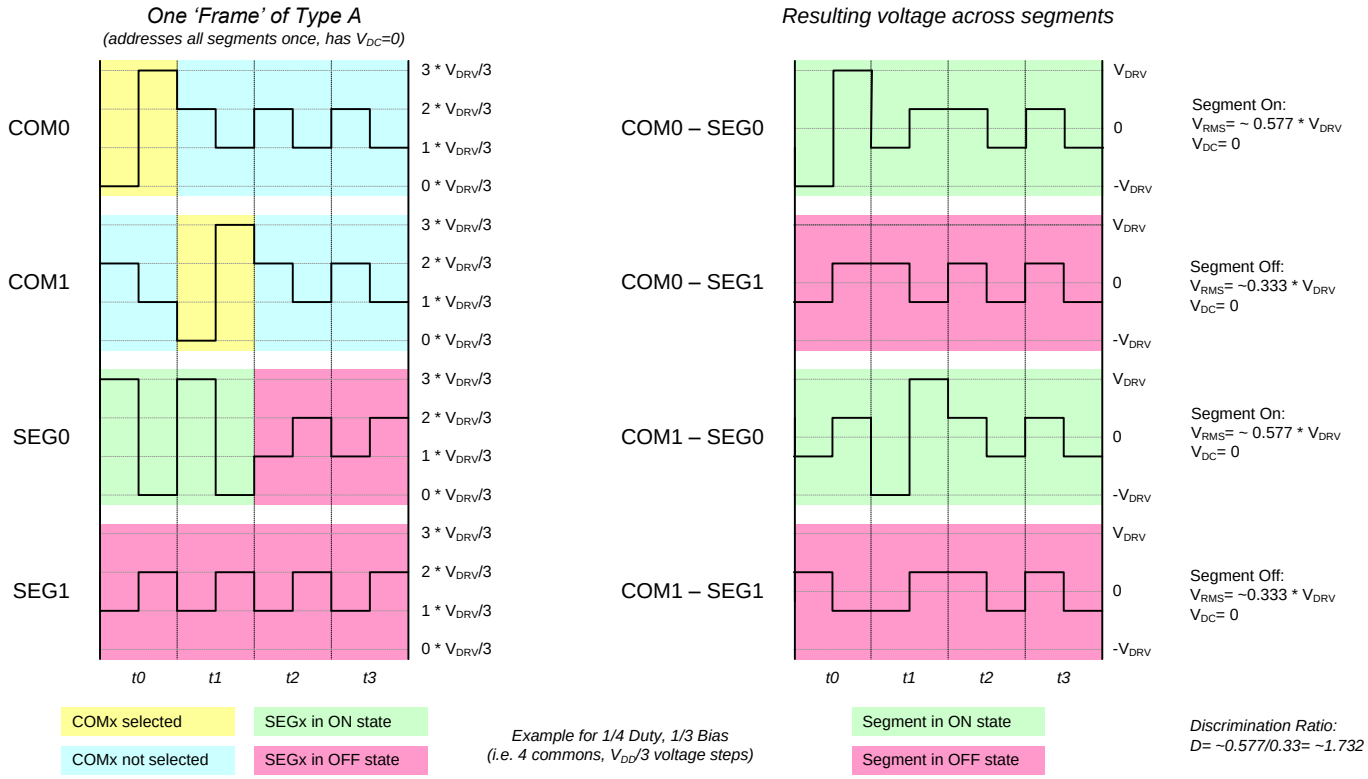
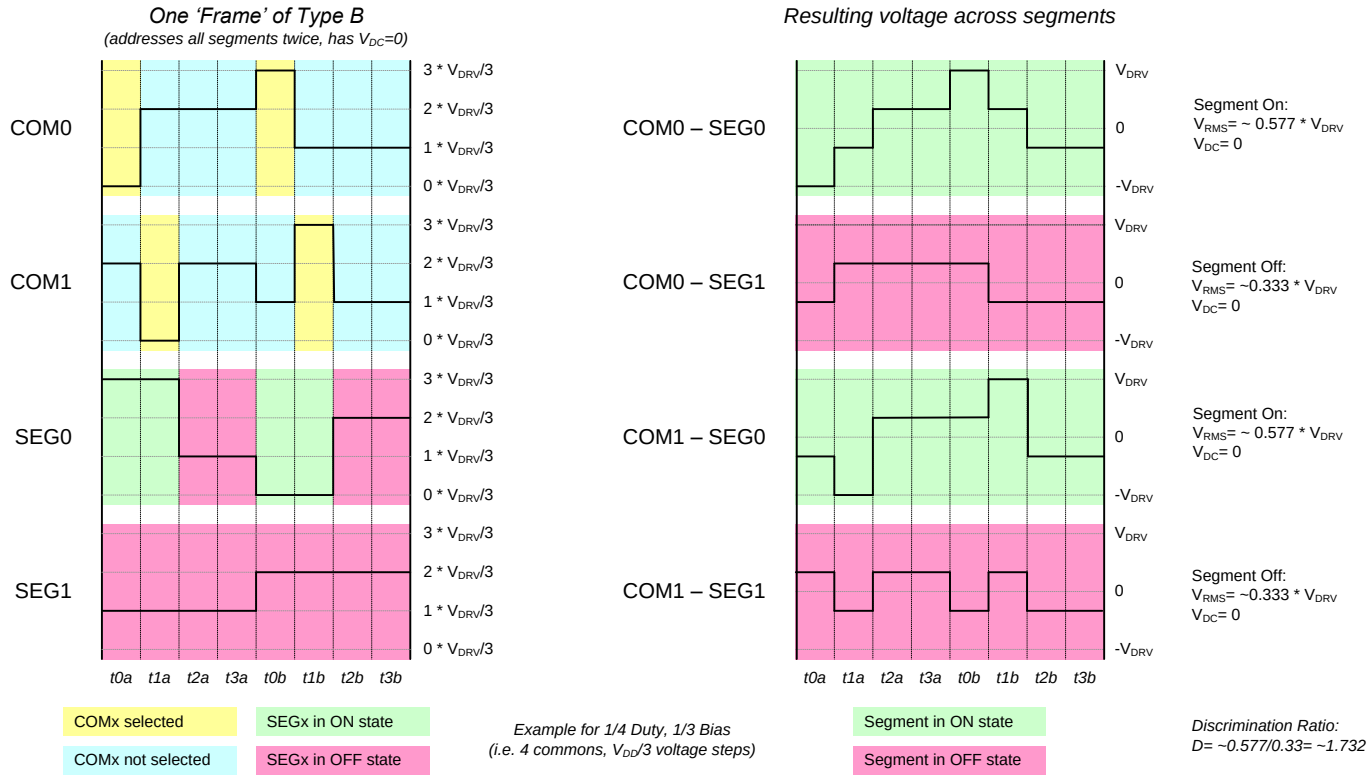
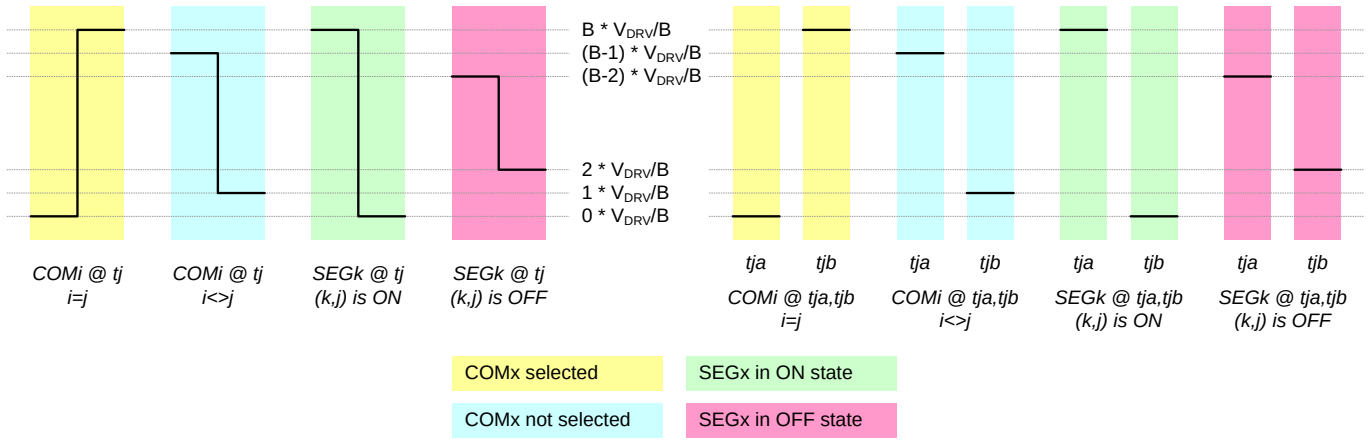


Figure 2. Conventional Type-A Waveform Example



The generalized waveforms for individual sub-frames for any duty and bias are illustrated in the following figure. Note that these use at most 6 different voltages (including VSS and VDRV).

Figure 3. Conventional Waveforms - Generalized



The effective RMS voltage for on and off segments can be calculated using these waveforms:

$$V_{RMS}\ (OFF\ ) = \sqrt{\frac{2(B-2)^2 + 2(M-1)(1)^2}{2M}} \times (V_{DRV}\ / B)$$

$$V_{RMS}\ (ON\ ) = \sqrt{\frac{2B^2 + 2(M-1)(1)^2}{2M}} \times (V_{DRV}\ / B)$$

The resulting Discrimination Ratio (D) for various bias and duty combinations is illustrated in the following table. The bias choice (B) for each duty (M) with the highest possible value for D is colored green.

Discrimination Ratio Table for Conventional Waveforms

Ratio (D) / Duty (M)	Bias (B) 1/2	1/3	1/4	1/5
1	INF	3.000	2.000	1.667
1/2	2.236	2.236	1.844	1.612
1/3	1.732	1.915	1.732	1.567
1/4	1.528	1.732	1.648	1.528
1/5	1.414	1.612	1.581	1.494
1/6	1.342	1.528	1.528	1.464
1/7	1.291	1.464	1.483	1.438
1/8	1.254	1.414	1.446	1.414
1/9	1.225	1.374	1.414	1.393
1/10	1.202	1.342	1.387	1.374
1/11	1.183	1.314	1.363	1.357
1/12	1.168	1.291	1.342	1.342
1/13	1.155	1.271	1.323	1.327
1/14	1.144	1.254	1.306	1.314
1/15	1.134	1.238	1.291	1.302
1/16	1.125	1.225	1.277	1.291

Digital Correlation

The principles of operation are illustrated by the example waveforms shown in the following figures.

Figure 4. Digital Correlation Example – Type-A

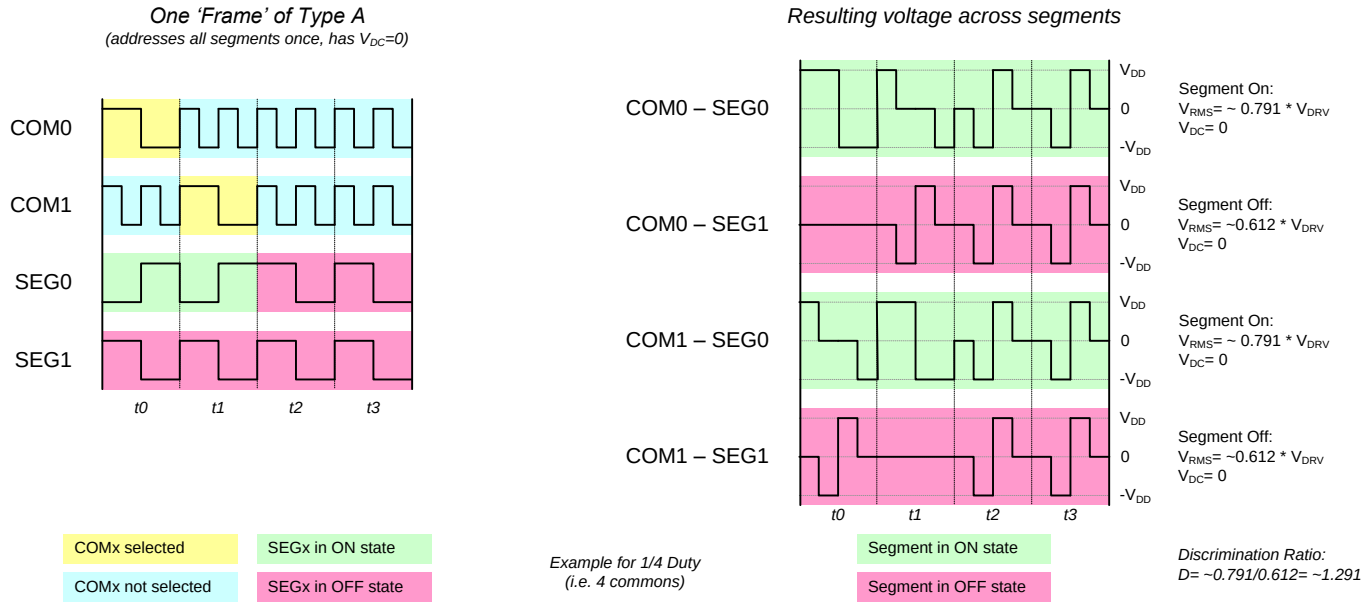
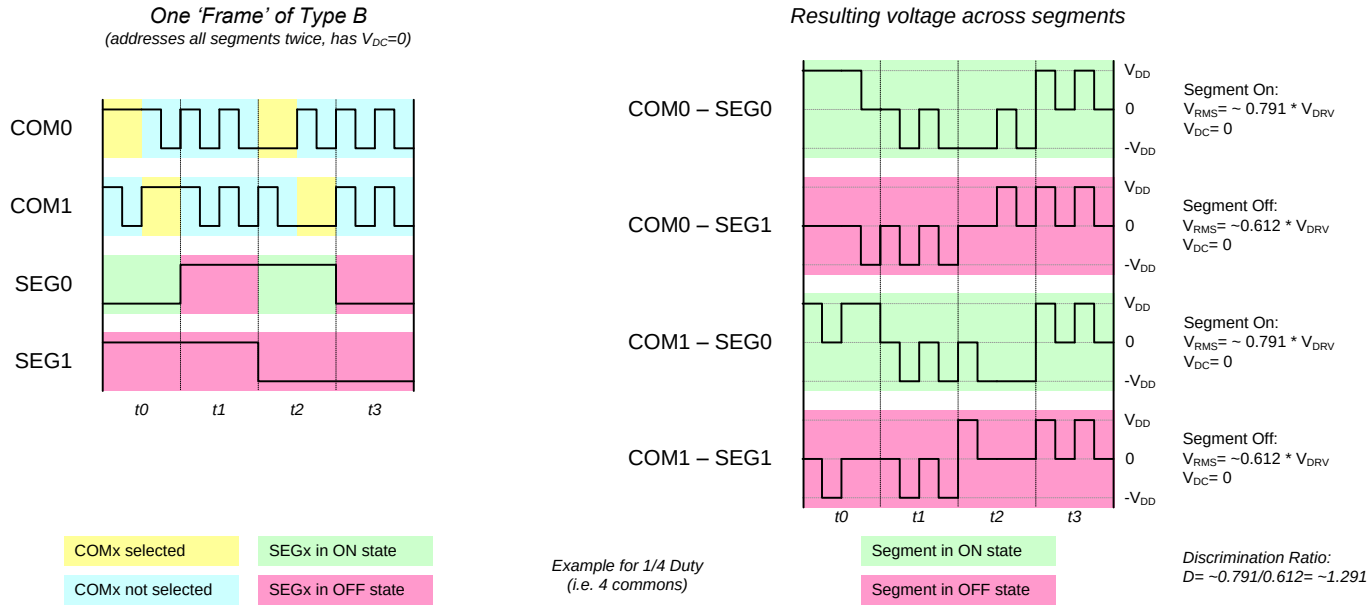


Figure 5. Digital Correlation Example - Type-B



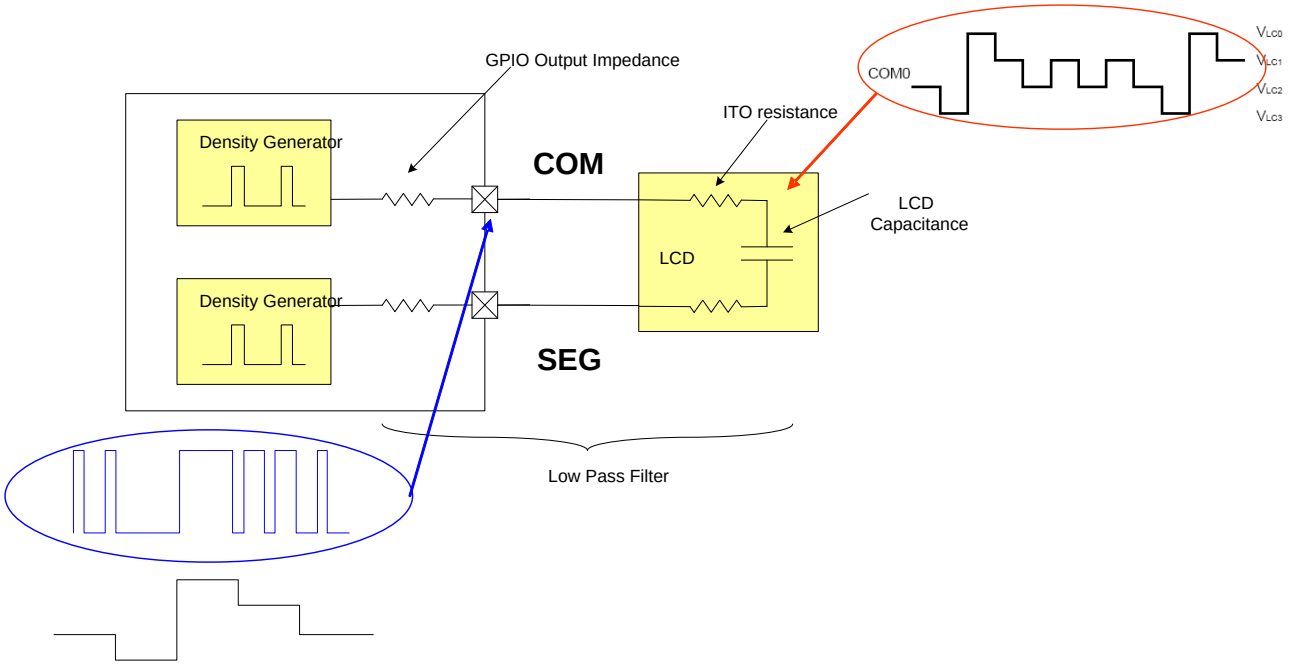
As illustrated, instead of generating bias voltages between the rails, this approach takes advantage of the characteristic of the LCD displays; the degree of on-ness and off-ness of LCD segments is determined by the RMS voltage across the segments. In this approach, the correlation coefficient between any given pair of COM and SEG signals determines whether the corresponding LCD segment is On or Off.

Thus, by doubling the base drive frequency of the COM signals in their inactive sub-frame intervals, the phase relationship of the COM and SEG drive signals can be varied to turn segments on and off – rather than varying the DC levels of the signals as is used in the conventional approaches.

PWM Drive

This approach duplicates the multi-voltage drive signals of the conventional method with bias B using a PWM output signal together with the intrinsic resistance and capacitance of the LCD display to create a simple PWM DAC. This is illustrated in the following figure.

Figure 6. PWM Principles of Operation



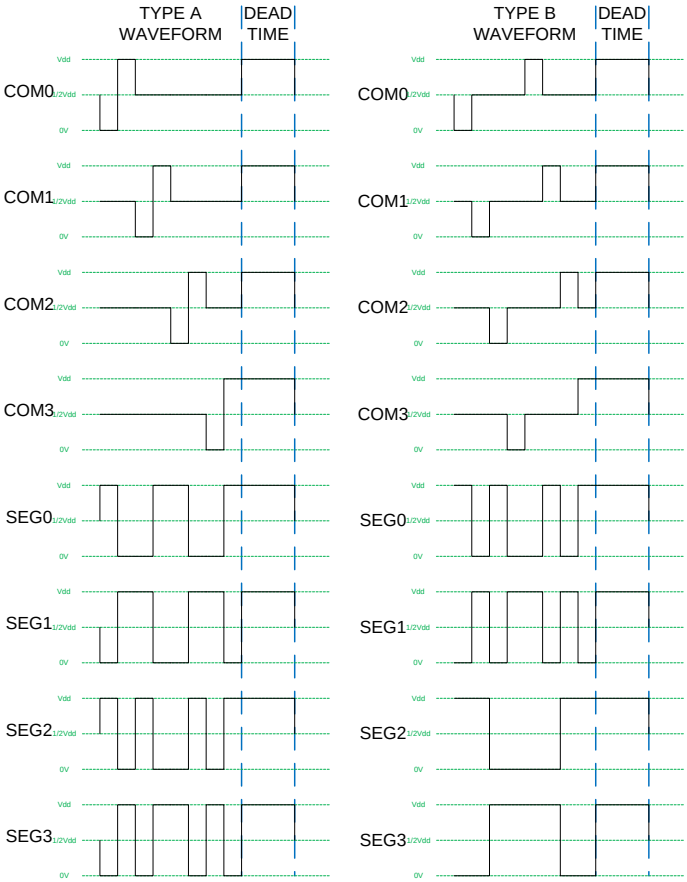
In order to drive a low capacitance display with acceptable ripple and rise/fall time using a 32 kHz PWM an additional external series resistance of 100 k - 1 M ohm is required. External Resistors are not required for PWM frequencies of greater than ~1 MHz. The exact frequency depends on the capacitance of the display, the internal ITO resistance of the ITO routing traces, and the drive impedance of the I/O pins.

The PWM method works for any bias value (B). Note that as B gets higher, a higher PWM step frequency is required to maintain the same PWM output frequency (the RC response of the LCD depends on the PWM output frequency, not the step frequency).

The PWM approach may also be used to drive a ½-bias display. This has the advantage that PWM is only required on the COM signals, as the SEG signals of a half bias display use only logic levels. Therefore, PWM half-bias can be supported at 32 kHz using just four external resistors.

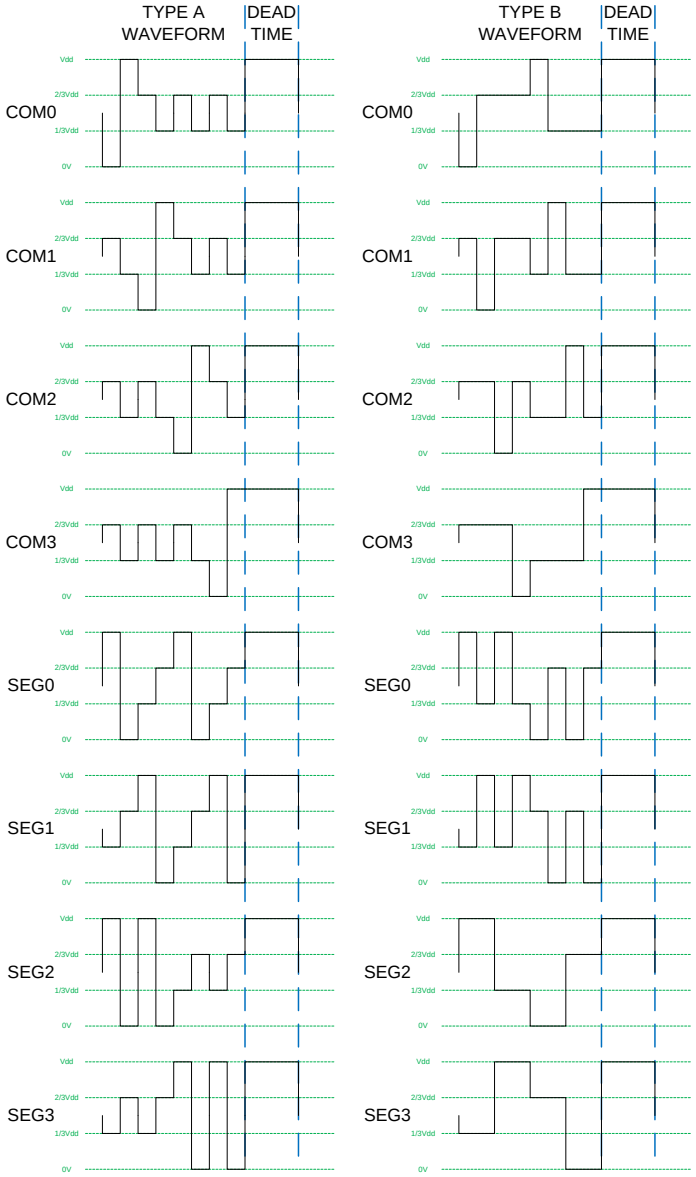
The power consumption of the approach (even for ½ bias) is substantially higher than other methods. Therefore, it is recommended that power-sensitive customers use Digital Correlation drive in Deep Sleep Mode, and change to PWM mode to gain the advantage of better contrast/viewing angle on TN displays when in Active or Sleep Mode.

Figure 7. PWM1/2 LCD drive waveform



COM / SEG	0	1	2	3
0	O	X	O	X
1	X	O	X	O
2	O	O	X	X
3	X	X	O	O

Figure 8. PWM1/3 LCD drive waveform

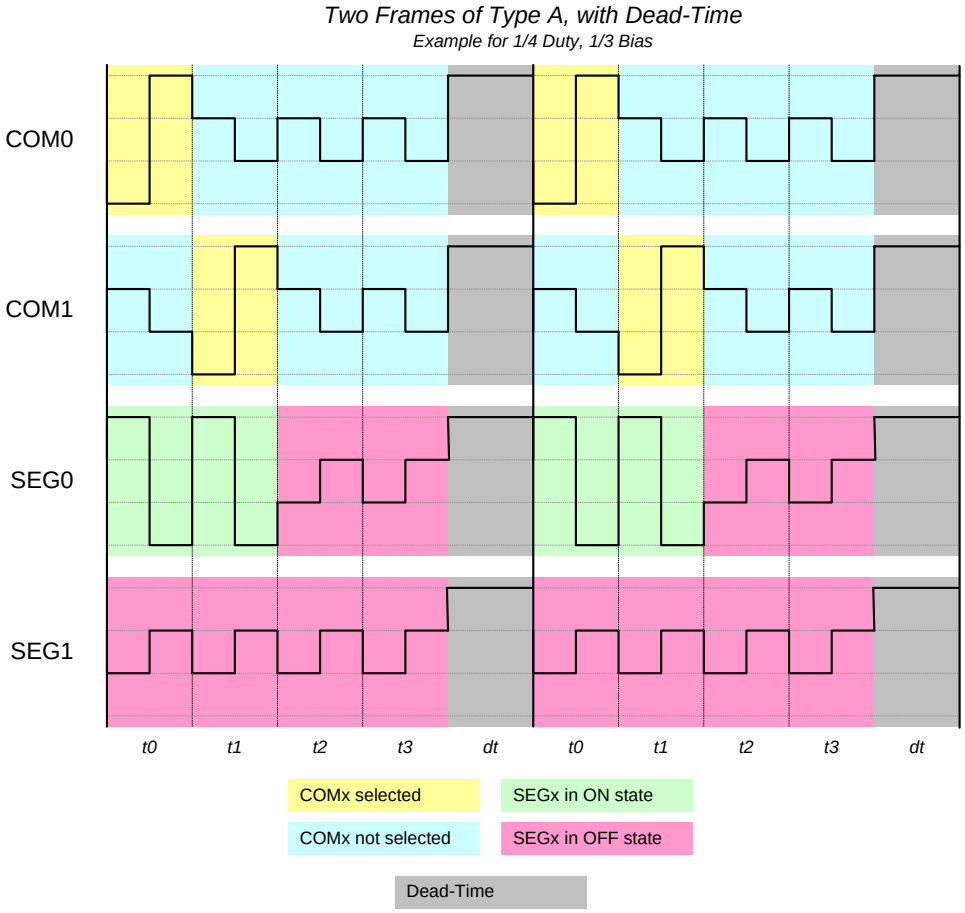


COM / SEG	0	1	2	3
0	O	X	O	X
1	X	O	X	O
2	O	O	X	X
3	X	X	O	O

Digital Contrast Control

In all modes, digital contrast control is available using the duty cycle/dead time method illustrated in the following figure.

Figure 9. "Dead Time" Contrast Control



This illustration shows the principle for 1/3 bias and 1/4 duty implementation, but the general approach of reducing contrast by reducing the percentage of time the glass is driven can be generalized and applied to any drive method.

In all cases, during the dead time, all COM and SEG signals are driven to a logic "1" state.

Registers

See the chip *Technical Reference Manual (TRM)* for more information about registers.

Resources

The Segment LCD component uses the dedicated LCD segment drive controller IP in the silicon.

DC and AC Electrical Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted. Specifications are valid for 1.71 V to 5.5 V, except where noted.

Note Final characterization data for the PSoC Analog Coprocessor device is not available at this time. Once the data is available, the component datasheet will be updated on the Cypress web site.

DC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
I_{LCDLOW}	Operating current in low power mode	–	5	–	µA	16 × 4 small segment disp. at 50 Hz
I_{LCDLOW}	Operating current in low power mode for PSoC 4200 BLE family	–	17.5	–	µA	16 × 4 small segment disp. at 50 Hz
C_{LCDCAP}	LCD capacitance per segment/common driver	–	500	5000	pF	Guaranteed by Design
LCD_{OFFSET}	Long-term segment offset	–	20	–	mV	
I_{LCDOP1}	LCD system operating current. Vbias = 5 V	–	2	–	mA	32 × 4 segments. 50 Hz, 25 °C
I_{LCDOP1}	PSoC 4100M/PSoC 4200M/PSoC 4200L, PWM Mode current. 5-V bias. 24-MHz IMO	–	0.6	–	mA	32 × 4 segments. 50 Hz, 25 °C
I_{LCDOP2}	LCD system operating current Vbias = 3.3 V	–	2	–	mA	32 × 4 segments. 50 Hz, 25 °C
I_{LCDOP2}	PSoC 4100M/PSoC 4200M/PSoC 4200L, PWM Mode current. 3.3-V bias. 24-MHz IMO.	–	0.5	–	mA	32 × 4 segments. 50 Hz, 25 °C

AC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
F_{LCD}	LCD frame rate	10	50	150	Hz	

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.30.c	Edited datasheet.	Added final characterization data for PSoC 4100S device.
1.30.b	Edited datasheet.	Edited Input/Output Connections, LCD Mode, Contrast and Clock Selection sections with more detailed description. Added final characterization data for PSoC 4000S device.
1.30.a	Edited datasheet.	Final characterization data for PSoC 4000S, PSoC 4100S and PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the component datasheet will be updated on the Cypress web site.
1.30	Added PSoC 4200L device specific data.	PSoC 4200L device support.
1.20	Added 8 Commons support. Added PWM 1/4 and PWM 1/5 bias modes support. Added dependencies of LCD Speed mode and LCD Drive mode from Number of Commons.	PSoC 4100M/PSoC 4200M device support.
	Edited the datasheet.	Updated DC and AC Electrical Characteristics section with PSoC 4100M/PSoC 4200M data.
1.10.a	Added DC and AC Electrical Characteristics for Bluetooth Low Energy devices.	
1.10	Added support for Bluetooth Low Energy devices.	
	Corrected upper limit of range for hexadecimal digit in SegLCD_Write7SegDigit_n() function.	For proper displaying of hexadecimal digit in case ASCII code of a hex number (A-F) by SegLCD_Write7SegDigit_n() function.
	Changed defines generation for proper handling of LCD Commons and Segments mapping on GPIO ports.	To correct mapping of LCD Commons and Segments on GPIO ports for PSoC 4200-BL device.
	Added Macros parameters description.	
1.0.a	Fixed parameter type in LCD_Seg_WriteBargraph_n() function description.	Clarify the function description.
1.0	Initial release	

© Cypress Semiconductor Corporation, 2015-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

