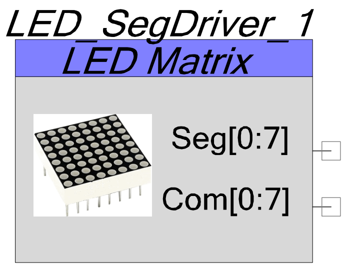
LED Matrix / 7 Segment Driver

1.00

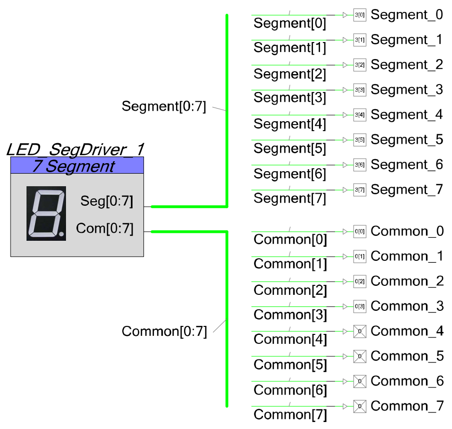


Features

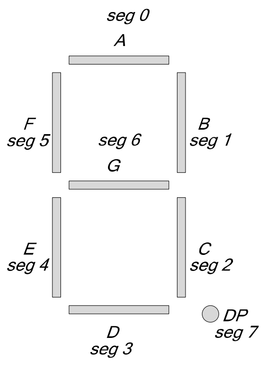
- 8 LEDs / Segments per common, up to 8 commons
  - 64 LED matrices
  - Up to 8, 7 segment displays with Decimal point
- Hardware driven, NO CPU or interrupts
- Easy to use API for controlling LED matrices and 7 segments displays
- Each common can have an independent brightness level
- Default refresh rate and PWM clock for ease of use, or external clock options for flexibility.
- Configurable segment and common drive levels for common cathode, common anode or other displays.

General Description

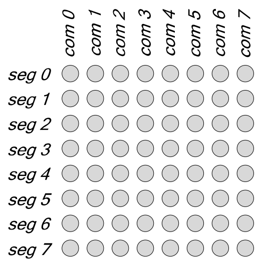
The LED_SegDriver component deals with multiplexing LED displays so you don't have to. All of the multiplexing is handled via DMA and logic so your CPU is free to deal with other tasks while you LED display is operating. Display RAM (1 byte per common) holds the data for each character display or row of LEDs, and to change the display, you need only change the display RAM. For ease of use, API are also included to display numbers and characters, as well as providing the ability to individually address row and column LEDs.



7 segment display connections



8 x 8 LED Matrix connections

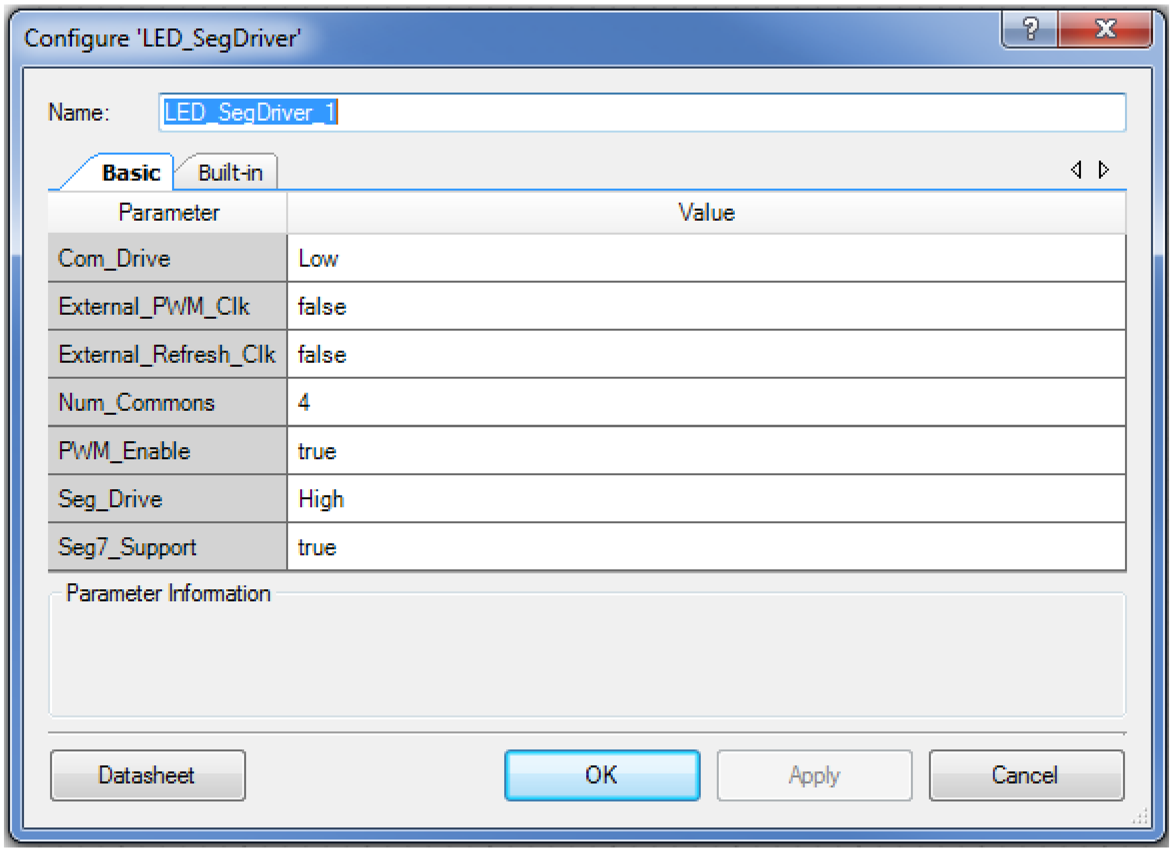


In addition, a PWM channel can also be enabled which will individually adjust the brightness of each common. The brightness for each common is stored in a separate brightness RAM which can be modified at any time.

Parameters and Setup

Drag an LED_SegDriver component onto your design and double-click it to open the Configure dialog.

Figure 1 Configure LED_SegDriver Dialog



Name: LED_SegDriver_1

Parameter	Value
Com_Drive	Low
External_PWM_Clk	false
External_Refresh_Clk	false
Num_Commons	4
PWM_Enable	true
Seg_Drive	High
Seg7_Support	true

The LED_SegDriver has the following parameters:

Com_Drive

This parameter sets the drive level to activate a common line. In the case of a common cathode display, this would be ‘low’ since the common cathode is active when driven low. For a common anode display, this would be set to ‘high’. You can also change this parameter to deal with logic inversions if you have an external transistor that will be responsible for carrying the current of the display common.

External_PWM_Clk

This parameter enables an optional terminal to set the frequency of the brightness PWM. This feature may only be enabled if the PWM_Enable parameter is set to ‘true’. When the PWM feature is enabled, an internal 250 Khz clock will drive the internal 8 bit PWM. If you wish to use a different frequency to drive the internal PWM, set this parameter to ‘true’ to enable an external clock terminal on the LED_SegDriver component. The clock you attach to this external PWM clock terminal will drive the 8 bit PWM responsible for controlling the brightness of each common. You may want to enable this option if you already have a clock in your system that would be suitable for driving the LED_SegDriver PWM and do not want to consume an extra clock unnecessarily in the LED_SegDriver component.

External_Refresh_Clk

This parameter enables an optional terminal to set the refresh rate of the LED display. If this parameter is set to ‘false’, an internal 500 Hz clock will be used to refresh the display. If it is set to ‘true’, any clock of a suitable frequency can be connected to this terminal to drive the refresh rate of the LED display. The refresh rate is the rate that the common is switched. For example, with a display of 8 commons and refresh rate of 500 Hz, each common is active for 2 ms, and the entire display is updated once every 16 ms.

Num_Commons

This parameter sets the number of commons you wish to drive with the LED_SegDriver component. Setting this parameter will also affect the maximum allowable bytes that can be used in the display RAM. More details will be provided later.

PWM_Enable

This parameter enables an optional PWM to independently control the brightness of each common by varying the duty cycle. A dedicated RAM, Num_Commons bytes in size, will be created that will contain the desired brightness value (from 0 to 255) for each common. An internal 250 Khz clock drives this PWM. You can select an external clock source for your PWM by setting the External_PWM_Clk parameter to ‘true’.

Seg_Drive

This parameter sets the drive level to activate a segment line. In the case of a common cathode display, this would be ‘high’ since the segment is active when driven high. For a common anode display, this would be set to ‘low’. You can also change this parameter to deal with logic inversions if you have an external transistor that will be responsible for carrying the current of the display segment.

Seg7_Support

Setting this parameter to true includes helper API for decoding numbers and characters and displaying them on a seven segment LED display. If you intend to use the LEG_SegDriver component only for matrix LED displays, setting this parameter to 'false' will save a small amount of RAM and FLASH space.

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 "LED_SegDriver_1" to the first instance of a component in a given design. You can rename it 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.

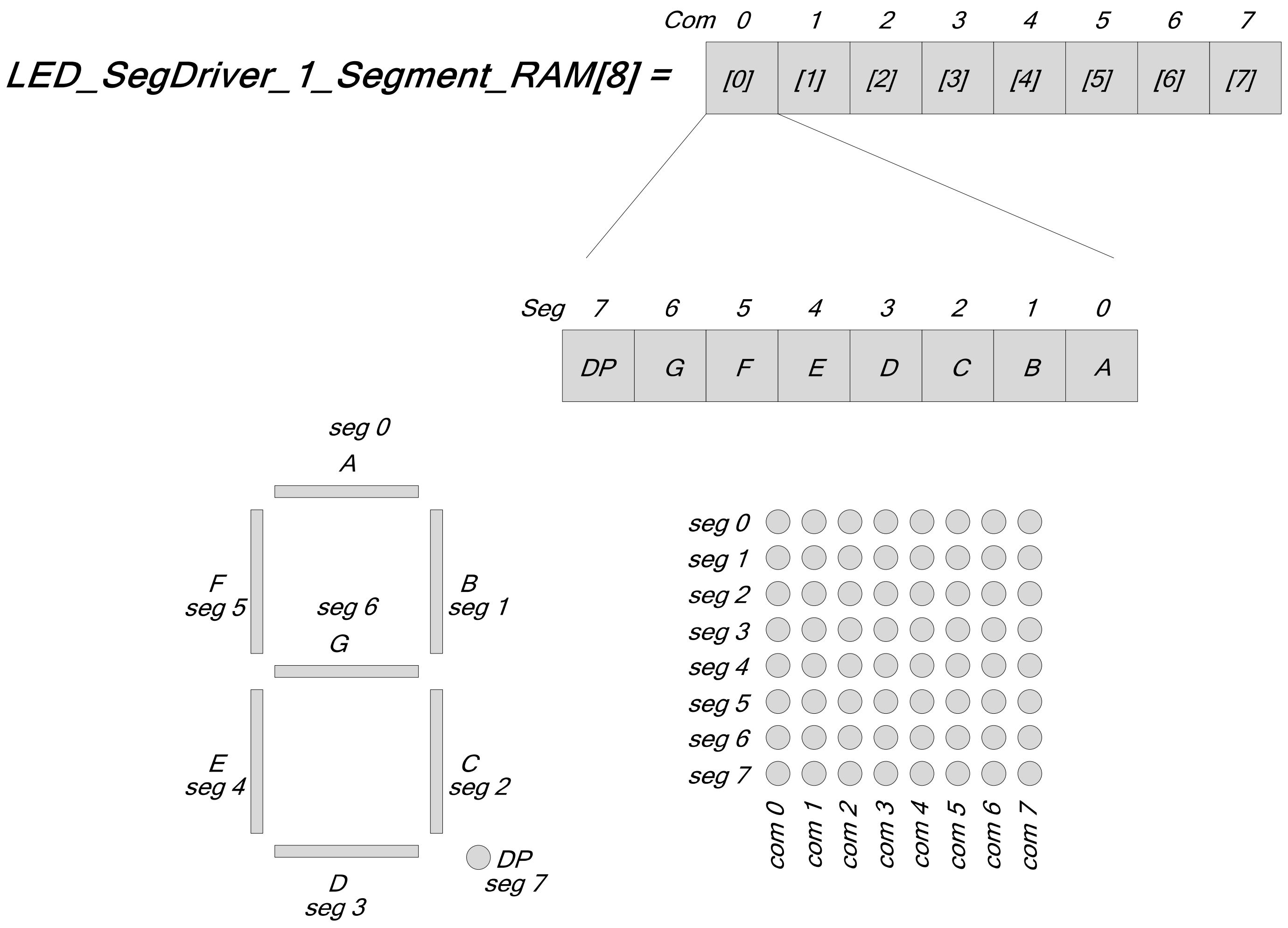
Function	Description
void LED_SegDriver_1_Start(void)	Configures the hardware (DMA and optional PWM) and enables the LED display.
void LED_SegDriver_1_Stop(void)	Clears the display RAM, disables all of the DMA channels and stops the PWM (if enabled).
void LED_SegDriver_1_SetDisplayRAM(uint8 value, uint8 display)	Writes 'value' directly into the display RAM for the display connected to common 'display'. I.e. to enable all of the LEDs connected to common 3, you would use: value = 0xFF and display = 3. (Com 0 = display 0, Com 1 = display 1 …)
uint8 LED_SegDriver_1_GetDisplayRAM(uint8 display)	Returns the value from the display RAM at location 'display'. To read the data from the RAM for all the LEDs connected to common 3, pass: display = 3. (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_SetRC(uint8 row, uint8 column)	Sets the bit in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.
void LED_SegDriver_1_ClearRC(uint8 row, uint8 column)	Clears the bit in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.
void LED_SegDriver_1_ToggleRC(uint8 row, uint8 column)	Toggles the bit in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.
uint8 LED_SegDriver_1_GetRC(uint8 row, uint8 column)	Returns the bit value in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.
void LED_SegDriver_1_ClearDisplay(uint8 display)	Clears the display (disables all the LEDs) for the associated common by setting the display RAM to zero. To clear the LEDs connects to common 1, pass: display = 1. (Com 0 = display 0, Com

Function	Description
	1 = display 1 …)
void LED_SegDriver_1_ClearDisplayAll(void)	Clears the entire display by writing zero to all the display RAM locations.

At all times, the following variable will be available. It can be written with firmware or DMA to directly control what is on the display. This is the display RAM.

uint8 LED_SegDriver_1_Segment_RAM[8]

This variable can be written directly in your firmware if you do not wish to use the provided API to manipulate the display data. While this array will always have 8 bytes associated with it, only the first bytes up to the number of commons you have will actually have an impact on the display. Common 0 data resides in byte 0, common 1 in byte 1 and so on. The bits in each byte correspond to the segments, with bit 7 corresponding to segment 7 …



Application Programming Interface if Seg7_Support is enabled

If The Seg7_Support parameter is set to true, the following API and variables will be available:

Function	Description
void LED_SegDriver_1_PutNumberInt(int32 value, uint8 display, uint8 digits)	Displays a signed integer up to 8 characters long, with the left most digit starting at “display” and extending for “digits” characters. The negative sign will take an extra digit if it is required. If you do not use the correct number of digits for your number, the least significant digits will be displayed. For example, if value is -1234, display is 0 and digits is 4, the result will be: -234. Unused characters are zero padded. For example if value is 12, display is 0 and digits is 4 the result will be: 0012 Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_PutNumberHex(uint32 value, uint8 display, uint8 digits)	Displays a hexadecimal number up to 8 characters long, with the left most digit starting at “display” and extending for “digits” characters. If you do not use the correct number of digits for your number, the least significant digits will be displayed. For example, if value is 0xDEADBEEF, display is 0 and digits is 4, the result will be: BEEF. Unused characters are zero padded. For example if value is 0xAF, display is 0 and digits is 4 the result will be: 00AF Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_PutString(char[] string, uint8 display)	Displays a string starting at “display” and ending at either the end of the string or the end of the display. The displayable characters are the same for the SegDriver_1_PutCharacter(…) function. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_PutChar (char value, uint8 display)	Displays a character at “display”. This function can display all alphanumeric characters, with special characters for ‘m’ and ‘n’. The function can also display ‘-‘, ‘.’, ‘_’, ‘ ‘, and ‘=’. All unknown characters are displayed as a space. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_SetNumberDec(uint8 value, uint8 display)	Displays a single digit. The number in ‘value’ (0 – 9) is placed on “display.” Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_SetNumberHex(uint8 value, uint8 display)	Displays a single digit. The number in ‘value’ (0 – F) is placed on “display.” Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
void LED_SegDriver_1_PutDecimalPoint(uint8 value, uint8 display)	Sets or clears the decimal point on “display” based on the least significant bit of “value”. If the LSb is set in value, the decimal point will be turned on. If the LSb is clear, the decimal point will be turned off. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Function	Description
uint8 LED_SegDriver_1_GetDecimalPoint(uint8 display)	Returns the status of the decimal point on “display”. If the LSb of the return value is set, then the decimal point is turned on. If the LSb is clear, the decimal point is off. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
uint8 LED_SegDriver_1_DecodeNumber(uint8 input)	Converts the lower 4 bits of the input into segment data that will display the number in hex on a display. The returned data can be written directly into the display RAM to display the desired number.
uint8 LED_SegDriver_1_DecodeAlpha(char input)	Converts the ascii encoded alphabet character input into the segment data that will display the character on a display. The returned data can be written directly into the display RAM to display the desired number.

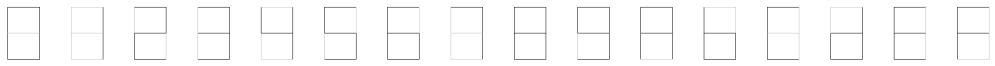
When Seg7_Support is enabled, the following decoder tables are available for use in your code if you do not wish to use the API to decode and display the information.

uint8 LED_SegDriver_1_seg7_Table[16] = Decoder table for converting numbers into segement values.

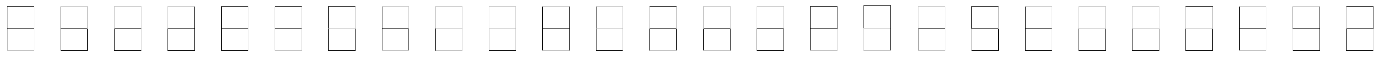
uint8 LED_SegDriver_1_seg7_Alpha_Table[26] = Decoder table for converting alphabet characters into segement values.

Defined character set:

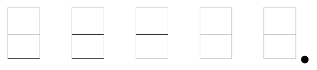
Numbers 0-9, 0-F:



Alphabet A-Z:



Special characters:



Application Programming Interface if PWM_Enable is set to ‘true’

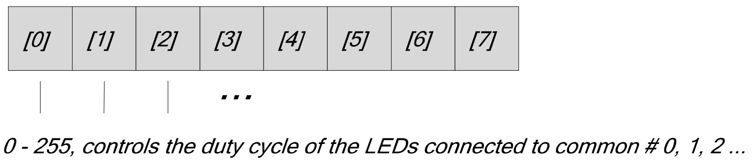
If The PWM_Enable parameter is set to true, the following API and variables will be available:

Function	Description
void LED_SegDriver_1_SetBrightnessRAM(uint8	Sets the desired brightness value (0 = display off – 255 = display at full brightness) for the chosen display by applying a PWM duty

Function	Description
value, uint8 display)	cycle to that displays common when the display is active. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)
uint8 LED_SegDriver_1_GetBrightnessRAM(uint8 display)	Returns the present brightness value for the chosen display. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

When PWM_Enable is set to true, the following variable will be available. It can be written with firmware or DMA to directly control the brightness for each common. This is the brightness RAM.

LED_SegDriver_1_Brightness_RAM[8] =


[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
…
0 - 255, controls the duty cycle of the LEDs connected to common # 0, 1, 2 …

void LED_SegDriver_1_Start (void)

Description: Configures the hardware (DMA and optional PWM) and enables the LED display. Whatever is currently in the display RAM will be displayed on the LED display. The display RAM is an array of 8 bytes called SegDriver_1_Segment_Ram[].

Parameters: None

Return Value: None

Side Effects: None

void LED_SegDriver_1_Stop (void)

Description: Clears the display RAM, disables all of the DMA channels and stops the PWM (if enabled in the customizer).

Parameters: None

Return Value: None

Side Effects: None

void LED_SegDriver_1_SetDisplayRAM (uint8 value, uint8 display)

Description: Writes 'value' directly into the display RAM for the display connected to common 'display'. I.e. to enable all of the LEDs connected to common 3, you would use: value = 0xFF and display = 3. (Com 0 = display 0, Com 1 = display 1 …). ***It is not necessary to use this API to manipulate the display RAM, you can also access the LED_SegDrive_1_Segment_RAM[8] directly.***

Parameters: uint8 value: Desired value to write into the display RAM. A logic '1' enables the segment, a logic '0' disables the segment.

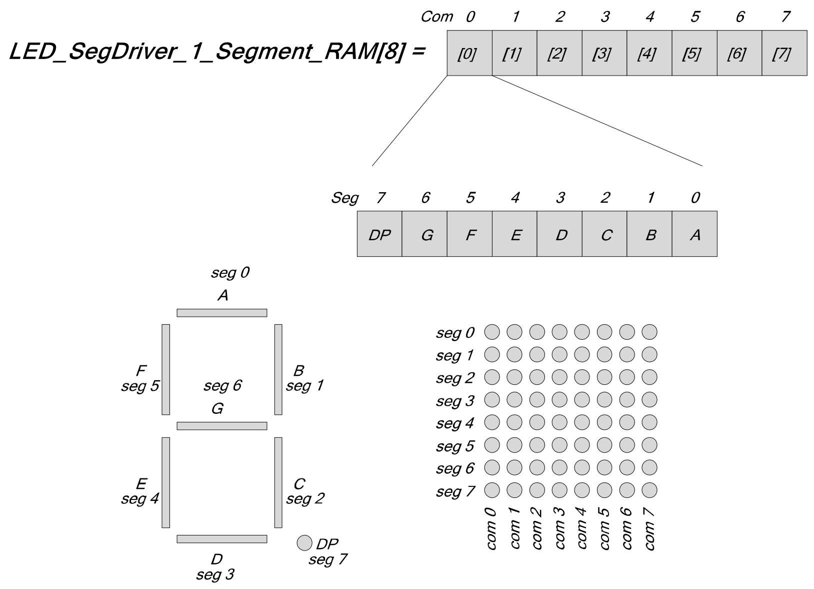
bit	Description
0	Controls segment 0 (A)
1	Controls segment 1 (B)
2	Controls segment 2 (C)
3	Controls segment 3 (D)
4	Controls segment 4 (E)
5	Controls segment 5 (F)
6	Controls segment 6 (G)
7	Controls segment 7 (DP)

uint8 display: number of the Display / Common RAM you wish to modify.

Display	Description
0	Modifies the display data for the LEDs connected to common 0
1	Modifies the display data for the LEDs connected to common 1
2	Modifies the display data for the LEDs connected to common 2
3	Modifies the display data for the LEDs connected to common 3
4	Modifies the display data for the LEDs connected to common 4
5	Modifies the display data for the LEDs connected to common 5
6	Modifies the display data for the LEDs connected to common 6
7	Modifies the display data for the LEDs connected to common 7

Return Value: None

Side Effects: None



uint8 LED_SegDriver_1_GetDisplayRAM (uint8 display)

Description: Returns the value from the display RAM at location 'display'. To read the data from the RAM for all the LEDs connected to common 3, pass: display = 3. (Com 0 = display 0, Com 1 = display 1 …). ***It is not necessary to use this API to read the display RAM, you can also access the LED_SegDrive_1_Segment_RAM[8] directly.***

Parameters: uint8 display: number of the Display / Common RAM you wish to read.

Display	Description
0	Reads the display data for the LEDs connected to common 0
1	Reads the display data for the LEDs connected to common 1
2	Reads the display data for the LEDs connected to common 2
3	Reads the display data for the LEDs connected to common 3
4	Reads the display data for the LEDs connected to common 4
5	Reads the display data for the LEDs connected to common 5
6	Reads the display data for the LEDs connected to common 6
7	Reads the display data for the LEDs connected to common 7

Return Value: Uint8 value. The data stored in the specified display RAM location.

Side Effects: None

void LED_SegDriver_1_SetRC(uint8 row, uint8 column)

Description: Sets the bit in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.

Parameters: Uint8 row: 0 – 7, represents the segment line
Uint8 column: 0 – 7, represents the common line

Return Value: None

Side Effects: None

void LED_SegDriver_1_ClearRC(uint8 row, uint8 column)

Description: Clears the bit in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.

Parameters: Uint8 row: 0 – 7, represents the segment line
Uint8 column: 0 – 7, represents the common line

Return Value: None

Side Effects: None

void LED_SegDriver_1_ToggleRC(uint8 row, uint8 column)

Description: Toggles the bit in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.

Parameters: Uint8 row: 0 – 7, represents the segment line
Uint8 column: 0 – 7, represents the common line

Return Value: None

Side Effects: None

void LED_SegDriver_1_GetRC(uint8 row, uint8 column)

Description: Returns the bit value in the display RAM corresponding to the LED in the designated row and column. Rows are the segments and columns are the commons.

Parameters: Uint8 row: 0 – 7, represents the segment line
Uint8 column: 0 – 7, represents the common line

Return Value: None

Side Effects: None

void LED_SegDriver_1_ClearDisplay(uint8 display)

Description: Clears the display (disables all the LEDs) for the associated common by setting the display RAM to zero. To clear the LEDs connects to common 1, pass: display = 1. (Com 0 = display 0, Com 1 = display 1 …)

Parameters: uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Clears the display data for the LEDs connected to common 0
1	Clears the display data for the LEDs connected to common 1
2	Clears the display data for the LEDs connected to common 2
3	Clears the display data for the LEDs connected to common 3
4	Clears the display data for the LEDs connected to common 4
5	Clears the display data for the LEDs connected to common 5
6	Clears the display data for the LEDs connected to common 6
7	Clears the display data for the LEDs connected to common 7

Return Value: None

Side Effects: None

void LED_SegDriver_1_ClearDisplayAll(void)

Description: Clears the entire display by writing zero to all the display RAM locations.

Parameters: None

Return Value: None

Side Effects: None

void LED_SegDriver_1_PutNumberInt(int32 value, uint8 display, uint8 digits)

Description: Displays a signed integer up to 8 characters long, with the left most digit starting at "display" and extending for "digits" characters. The negative sign will take an extra digit if it is required. If you do not use the correct number of digits for your number, the least significant digits will be displayed. For example, if value is -1234, display is 0 and digits is 4, the result will be: -234. Unused characters are zero padded. For example if value is 12, display is 0 and digits is 4 the result will be: 0012
Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: Int32 value: a signed integer number to display. It is the responsibility of the user to ensure that the display has enough digits to accurately represent the number passed to this function. If not, the least significant digits will be displayed. Also note that a negative number will require 1 more digit than the equivalent positive number to display the negative sign.

uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Put the left most digit on the display connected to common 0
1	Put the left most digit on the display connected to common 1
2	Put the left most digit on the display connected to common 2
3	Put the left most digit on the display connected to common 3
4	Put the left most digit on the display connected to common 4
5	Put the left most digit on the display connected to common 5
6	Put the left most digit on the display connected to common 6
7	Put the left most digit on the display connected to common 7

Uint8 digits: the number of digits you wish to use to display the value passed into the function. The negative sign will take an extra digit if it is required. If you do not use the correct number of digits for your number, the least significant digits will be displayed. For example, if value is -1234, display is 0 and digits is 4, the result will be: -234. Unused characters are zero padded. For example if value is 12, display is 0 and digits is 4 the result will be: 0012

Return Value: None

Side Effects: None

void LED_SegDriver_1_PutNumberHex(uint32 value, uint8 display, uint8 digits)

Description: Displays a hexadecimal number up to 8 characters long, with the left most digit starting at "display" and extending for "digits" characters. If you do not use the correct number of digits for your number, the least significant digits will be displayed. For example, if value is 0xDEADBEEF, display is 0 and digits is 4, the result will be: BEEF. Unused characters are zero padded. For example if value is 0xAF, display is 0 and digits is 4 the result will be: 00AF

Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: uInt32 value: a hexadecimal number to display. It is the responsibility of the user to ensure that the display has enough digits to accurately represent the number passed to this function. If not, the least significant digits will be displayed.

uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Put the left most digit on the display connected to common 0
1	Put the left most digit on the display connected to common 1
2	Put the left most digit on the display connected to common 2
3	Put the left most digit on the display connected to common 3
4	Put the left most digit on the display connected to common 4
5	Put the left most digit on the display connected to common 5
6	Put the left most digit on the display connected to common 6
7	Put the left most digit on the display connected to common 7

Uint8 digits: the number of digits you wish to use to display the value passed into the function. If you do not use the correct number of digits for your number, the least significant digits will be displayed. For example, if value is 0xDEADBEEF, display is 0 and digits is 4, the result will be: BEEF. Unused characters are zero padded. For example if value is 0xAF, display is 0 and digits is 4 the result will be: 00AF

Return Value: None

Side Effects: None

void LED_SegDriver_1_PutString(char * string, uint8 display)

Description: Displays a null terminated string starting at “display” and ending at either the end of the string or the end of the display. The displayable characters are the same for the SegDriver_1_PutCharacter(…) function. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: Char * string: the null terminated string to be displayed

uint8 display: number of the Display / Common RAM you wish to clear.

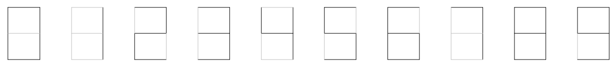
Display	Description
0	Put the left most character on the display connected to common 0
1	Put the left most character on the display connected to common 1
2	Put the left most character on the display connected to common 2
3	Put the left most character on the display connected to common 3
4	Put the left most character on the display connected to common 4
5	Put the left most character on the display connected to common 5
6	Put the left most character on the display connected to common 6
7	Put the left most character on the display connected to common 7

Return Value: None

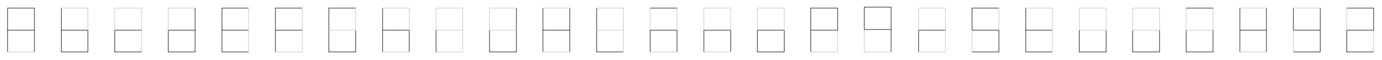
Side Effects: None

Defined character set:

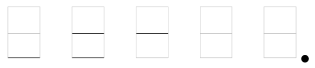
Numbers 0-9:



Alphabet A-Z:



Special characters:



void LED_SegDriver_1_Char(char value, uint8 display)

Description: Displays an ascii encoded character at "display". This function can display all alphanumeric characters, with special characters for 'm' and 'n'. The function can also display '-', '.', '_', ' ', and '='. All unknown characters are displayed as a space. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: Char value: ascii character

uint8 display: number of the Display / Common RAM you wish to clear.

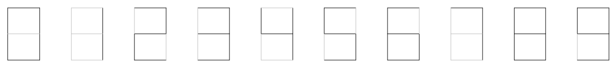
Display	Description
0	Put the character on the display connected to common 0
1	Put the character on the display connected to common 1
2	Put the character on the display connected to common 2
3	Put the character on the display connected to common 3
4	Put the character on the display connected to common 4
5	Put the character on the display connected to common 5
6	Put the character on the display connected to common 6
7	Put the character on the display connected to common 7

Return Value: None

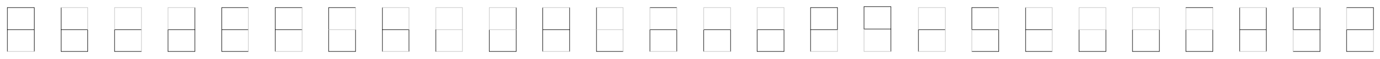
Side Effects: None

Defined character set:

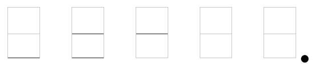
Numbers 0-9:



Alphabet A-Z:



Special characters:



void LED_SegDriver_1_SetNumberDec(uint8 value, uint8 display)

Description: Displays a single digit on the specified display. The number in 'value' (0 – 9) is placed on "display." Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: Uint8 value: a number between 0 and 9 to display.

uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Put the digit on the display connected to common 0
1	Put the digit on the display connected to common 1
2	Put the digit on the display connected to common 2
3	Put the digit on the display connected to common 3
4	Put the digit on the display connected to common 4
5	Put the digit on the display connected to common 5
6	Put the digit on the display connected to common 6
7	Put the digit on the display connected to common 7

Return Value: None

Side Effects: None

void LED_SegDriver_1_SetNumberHex(uint8 value, uint8 display)

Description: Displays a single digit on the specified display. The number in 'value' (0 – F) is placed on "display." Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: Uint8 value: a number between 0x0 and 0xF to display.

uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Put the digit on the display connected to common 0
1	Put the digit on the display connected to common 1
2	Put the digit on the display connected to common 2
3	Put the digit on the display connected to common 3
4	Put the digit on the display connected to common 4
5	Put the digit on the display connected to common 5
6	Put the digit on the display connected to common 6
7	Put the digit on the display connected to common 7

Return Value: None

Side Effects: None

void LED_SegDriver_1_PutDecimalPoint(uint8 value, uint8 display)

Description: Sets or clears the decimal point on “display” based on the least significant bit of “value”. If the LSb is set in value, the decimal point will be turned on. If the LSb is clear, the decimal point will be turned off. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: Uint8 value: only the least significant bit of value matters. If the LSb is set, the decimal point will be enabled. If the LSb is cleared, the decimal point will be cleared.

uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Sets the decimal point for the display connected to common 0
1	Sets the decimal point for the display connected to common 1
2	Sets the decimal point for the display connected to common 2
3	Sets the decimal point for the display connected to common 3
4	Sets the decimal point for the display connected to common 4
5	Sets the decimal point for the display connected to common 5
6	Sets the decimal point for the display connected to common 6
7	Sets the decimal point for the display connected to common 7

Return Value: None

Side Effects: None

Uint8 LED_SegDriver_1_GetDecimalPoint(uint8 display)

Description: Returns the status of the decimal point on “display”. If the LSb of the return value is set, then the decimal point is turned on. If the LSb is clear, the decimal point is off. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Returns the decimal point for the display connected to common 0
1	Returns the decimal point for the display connected to common 1
2	Returns the decimal point for the display connected to common 2
3	Returns the decimal point for the display connected to common 3
4	Returns the decimal point for the display connected to common 4
5	Returns the decimal point for the display connected to common 5
6	Returns the decimal point for the display connected to common 6
7	Returns the decimal point for the display connected to common 7

Return Value: Uint8 value: the LSb will be set or cleared depending on the status of the decimal point for the specified display.

Side Effects: None

Uint8 LED_SegDriver_1_DecodeNumber(uint8 input)

Description:	Converts the lower 4 bits of the input into segment data that will display the number in hex on a display. The returned data can be written directly into the display RAM to display the desired number. It is not necessary to use this function since higher level API are provided to both decode the value and write it to the display RAM. Also, the decoder table LED_SegDriver_1_seg7_Table[16] is available to use directly.
Parameters:	Uint8 number: A number between 0x0 and 0xF to be converted into segment data.
Return Value:	Uint8 segment_data: the value to be written into the display RAM for displaying the specified number.
Side Effects:	None

Uint8 LED_SegDriver_1_DecodeAlpha(char input)

Description:	Converts the ascii encoded alphabet character input into the segment data that will display the alphabet character on a display. The returned data can be written directly into the display RAM to display the desired number. It is not necessary to use this function since higher level API are provided to both decode the value and write it to the display RAM. Also, the decoder table LED_SegDriver_1_seg7_Alpha_Table[26] is available to use directly.
Parameters:	Char input: An ascii alphabet character to be converted into segment data.
Return Value:	Uint8 segment_data: the value to be written into the display RAM for displaying the specified character.
Side Effects:	None

void LED_SegDriver_1_SetBrightnessRAM(uint8 value, uint8 display)

Description: Sets the desired brightness value (0 = display off – 255 = display at full brightness) for the chosen display by applying a PWM duty cycle to that displays common when the display is active. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: None

Return Value: Uint8 value: 0 – 255, sets the 8 bit duty cycle of the PWM connected to the specified common. 255 is full on, 0 is full off.

uint8 display: number of the Display / Common RAM you wish to clear.

Display	Description
0	Sets the brightness for the LEDs connected to common 0
1	Sets the brightness for the LEDs connected to common 1
2	Sets the brightness for the LEDs connected to common 2
3	Sets the brightness for the LEDs connected to common 3
4	Sets the brightness for the LEDs connected to common 4
5	Sets the brightness for the LEDs connected to common 5
6	Sets the brightness for the LEDs connected to common 6
7	Sets the brightness for the LEDs connected to common 7

Side Effects: None

Uint8 LED_SegDriver_1_GetBrightnessRAM(uint8 display)

Description: Returns the present brightness value for the chosen display. Display is the desired display number (Com 0 = display 0, Com 1 = display 1 …)

Parameters: uint8 display: number of the Display / Common RAM you wish to clear.

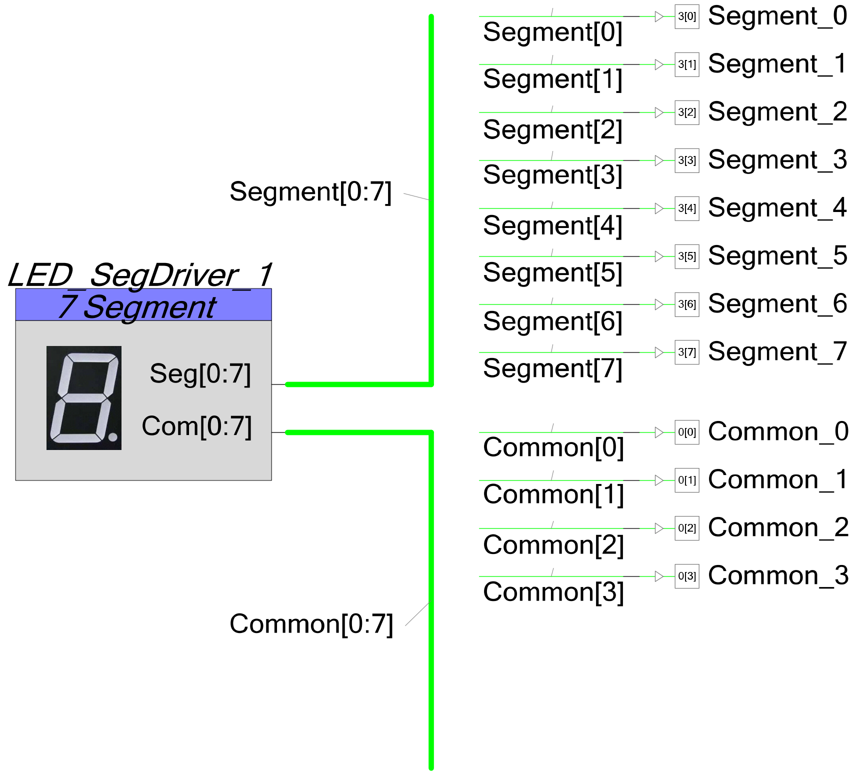
Display	Description
0	Returns the brightness for the LEDs connected to common 0
1	Returns the brightness for the LEDs connected to common 1
2	Returns the brightness for the LEDs connected to common 2
3	Returns the brightness for the LEDs connected to common 3
4	Returns the brightness for the LEDs connected to common 4
5	Returns the brightness for the LEDs connected to common 5
6	Returns the brightness for the LEDs connected to common 6
7	Returns the brightness for the LEDs connected to common 7

Return Value: Uint8 brightness: returns the current brightness setting for the specified display. 255 is full brightness and 0 is off.

Side Effects: None

Sample Firmware Source Code

The following example is all that is required to display data on the LCD. No need to call ADC start or LCD start. Everything is handled by the component.



```
void main()
{
      int8 j = 0;

      LED_SegDriver_1_Start();

      while(1)
      {
            LED_SegDriver_1_PutNumberInt(j, 0, 4);
            j++;
            CyDelay(150);
      }

}
```

© Cypress Semiconductor Corporation, 2009-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

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

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

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

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