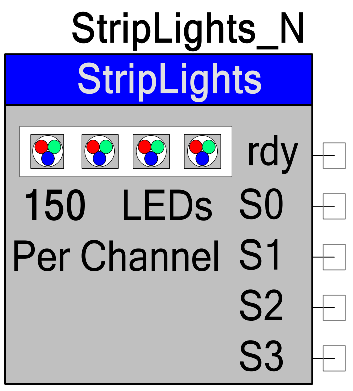
WS281xLib

3.0

Features

- Drives popular WS2811/12 LEDs or NeoPixels
- Hardware driven interface (No bit-banging)
- Supports 4-LED SK2812-RGBW modules
- Drives 1 to 16 strings of LEDs
- Supports 24-bit RGB and Color Lookup Table
- Supports 5x7 character display
- LEDs arranged as NxM graphics array
- Pixel, line, rectangle, circle primitives supported
- Triggered refresh occurs in ISR
- Supports Standard, Spiral, or 16x16 Grid Configuration

  Note: Requires PSoC Creator 4.1 or higher.



General Description

The StripLights component will control up to 3000 RGB LEDs with a single PSoC 4. It drives the popular WS2812, WS2811, and SK2812-RGBW LED modules. Internal CPU SRAM is used as display memory which is written to the LED modules when triggered by firmware. This component uses PSoC UDBs to perform the serial shifting of data to the WS2812 strings. The UDB FIFOs are used to hold the 3 or 4 byte packets for each LED module so that an entire packet can be loaded at once. Only about 10 or 12% of the CPU is required during display update, compared to 100% CPU that is required for many common bit-banging solutions.

The LEDs are configured in an NxM array to make programming easier. The X and Y coordinates can be inverted or swapped to simplify display orientation and to allow similar designs to be used across many LED module configurations.

The component can be configured in a number of ways to adapt to almost any user configuration. Simple drawing primitives used to draw pixels, lines, rectangles and circles, are supported. Printing of characters and strings with a 5x7 font is also supported.

To assist in using colors that will be consistent between the WS2811 and WS2812, there are several pre-defined color constants at the end of the StripLights.h file.

Note: Must use PSoC Creator 4.1 or higher.

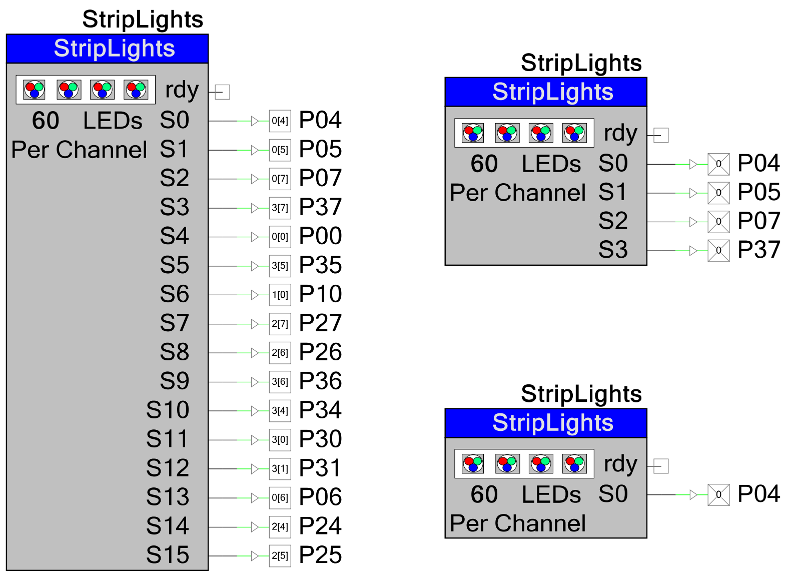


Figure 1 Examples project schematic

Input/Output Connections

This section describes the various input and output connections for the StripLights 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.

rdy – Digital Output

The "rdy" signal goes high when the transfer to a single string is complete.

S0-15 – Digital Output

There may be between 1 and 16 outputs depending on the configuration of the component. For best results, place the LED strips in sequential order from S0 to Sn. The component will only have as many outputs as required.

Below is an example of how six rows of LED modules may be connected and the default coordinate system that results. The coordinate system may be modified with the parameters in the interface to make your code compatible across multiple hardware configurations.

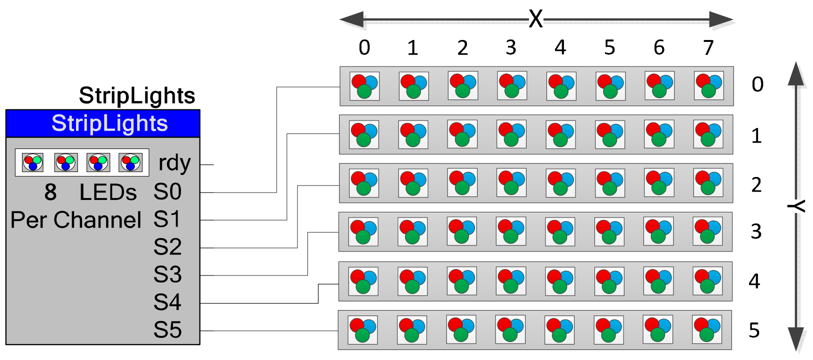
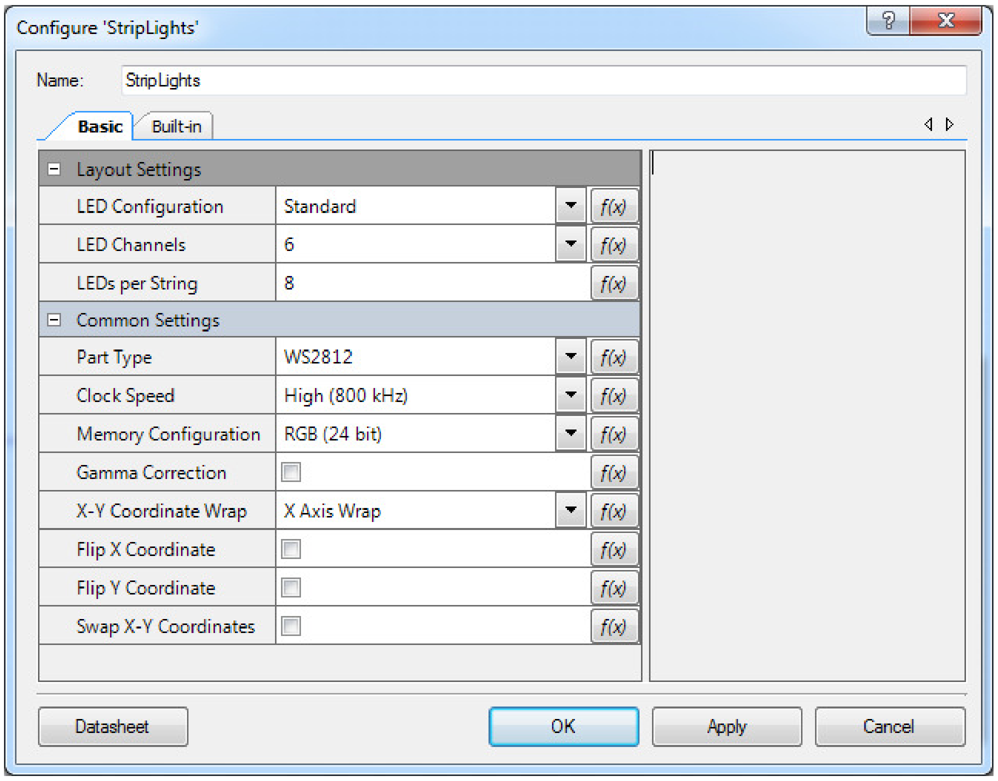


Figure 2 Example 6 Row by 8 Column configuration

Parameters and Setup

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



Configure 'StripLights'

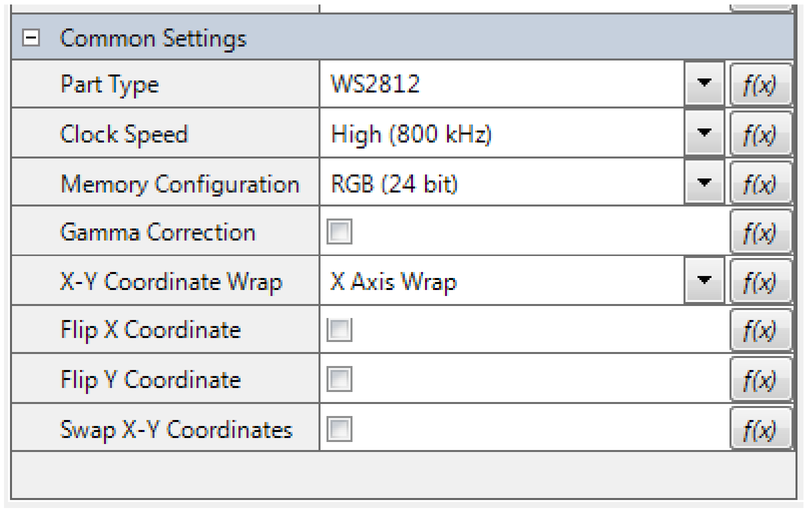
Name: StripLights

Layout Settings	
LED Configuration	Standard
LED Channels	6
LEDs per String	8
Common Settings	
Part Type	WS2812
Clock Speed	High (800 kHz)
Memory Configuration	RGB (24 bit)
Gamma Correction	☐
X-Y Coordinate Wrap	X Axis Wrap
Flip X Coordinate	☐
Flip Y Coordinate	☐
Swap X-Y Coordinates	☐

Figure 3 Configuration Interface

Common Settings

These are the parameters that are common between all LED Configurations.



Common Settings	
Part Type	WS2812
Clock Speed	High (800 kHz)
Memory Configuration	RGB (24 bit)
Gamma Correction	☐
X-Y Coordinate Wrap	X Axis Wrap
Flip X Coordinate	☐
Flip Y Coordinate	☐
Swap X-Y Coordinates	☐

Figure 4 Common Settings for all layouts.

Part Type

This parameter selects which type of LED module that you are using. Although the WS2811 and WS2812 appear to be the same, the order that the color data is transmitted is different. If you have selected either the WS2811 or WS2812 and the colors are incorrect, switch to the other module. The WS2812B works the same as the WS2812, but the module has only 4 pins instead of 6 pins. The SK2812 RGBW module has a fourth white LED.

- WS2811
- **WS2812**
- SK2812 RGBW

Clock Speed

This parameter selects the clock speed of the interface to the LED modules. This WS2811/12 parts operate at one of two frequencies, 400 kHz or 800 kHz, select the speed that is specified by the manufacturer. 800 kHz should work for most modules.

- Low (400 kHz)
- **High (800 kHz)**

Memory Configuration

This parameter determines if the color information is stored as a 32-bit RGB value or an 8-bit index to a color lookup table. The RGB (24-bit) method generates faster code, but requires 32-bits (4 Bytes) for each LED module. The 8-Bit mode only requires a single byte in memory for each LED module. This configuration requires less memory, but requires more CPU power during the interrupt to make use of a color lookup table. Also, color blending is not possible with the use of the lookup table.

- LUT (8-bit)
- **RGB (24-bit)**

Gamma_Correction

Setting this parameter to true will adjust the linear numerical value to a setting whose intensity appears to be linear to the human eye.   This is experimental at this time.

Coord_Wrap

When this parameter is set, the X and/or Y coordinates use the modulo function to wrap around from the maximum value back to zero.

Options:

No Coord Wrap: Normal Coordinate System

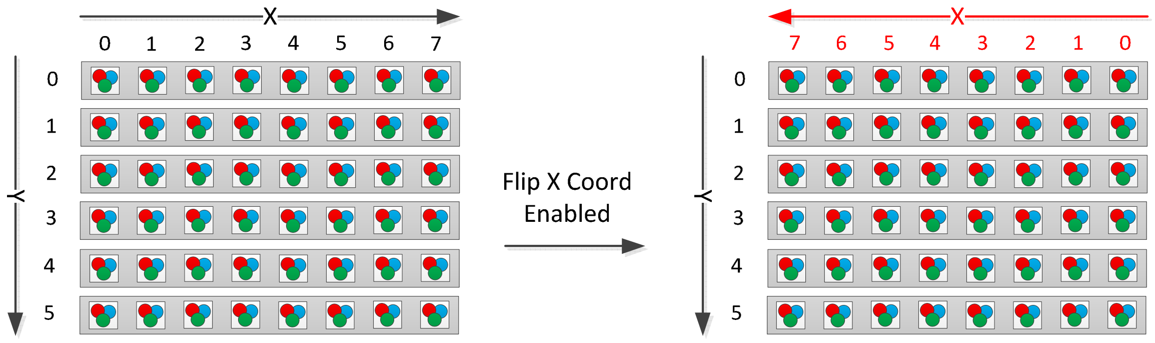
X Axis Wrap: Performs a module function on the X axis value.

Y Axis Wrap: Performs a module function on the Y axis value.

XY Axis Wrap: Module function performed on both X and Y axis values.

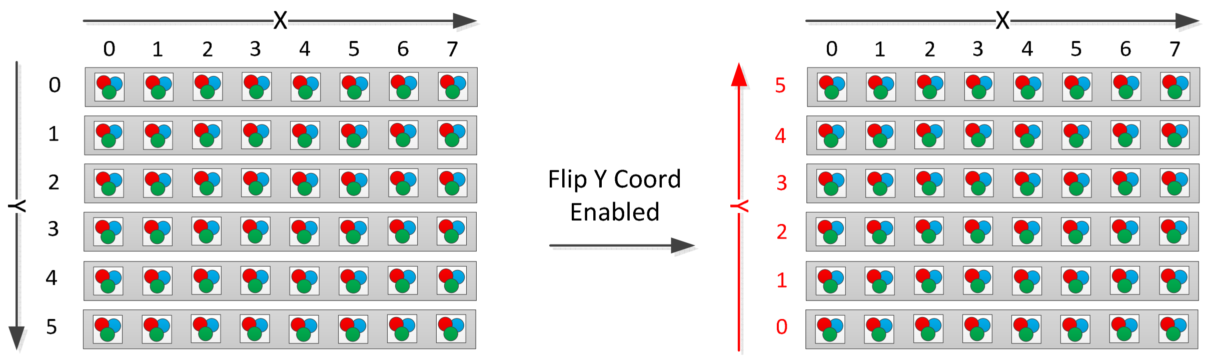
Flip X Coord

This parameter reverses the direction of the X coordinates system as shown below.



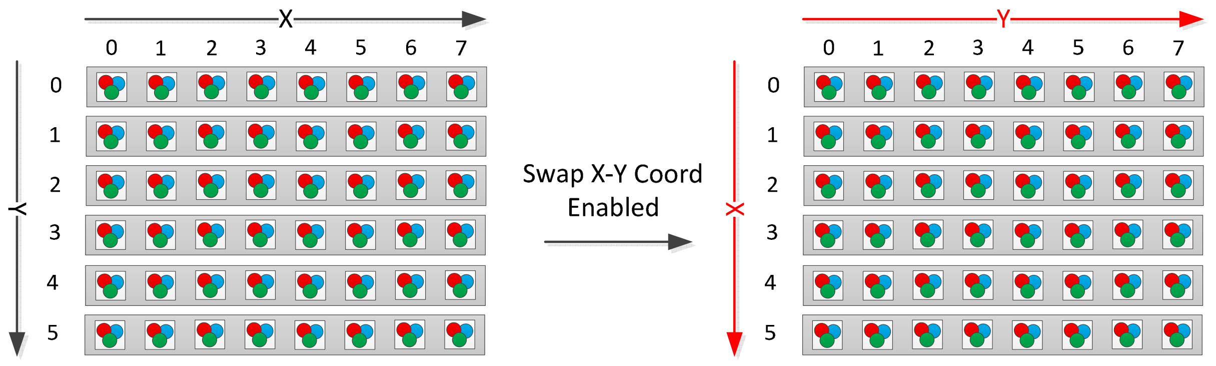
Flip Y Coord

This parameter reverses the direction of the Y coordinates system as shown below.



Swap X-Y Coordinates

This parameter swaps the X and Y coordinate system.



Layout Settings

The layout parameters are dependent on the LED Configuration parameter. It determines how your LED modules are arranged in hardware. Each of the LED Configurations has a different set of associated parameters. The three different LED Configurations are as shown.

- **Standard**
- Spiral
- Grid_16x16

Standard Mode

This is the default mode that consists of one or more strips of LEDs that are configured as a grid. This method is convenient when creating a large LED array used for text and drawing.

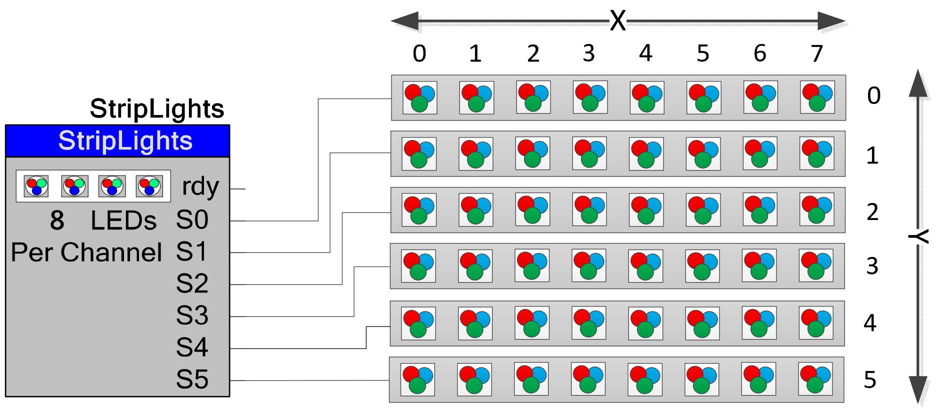
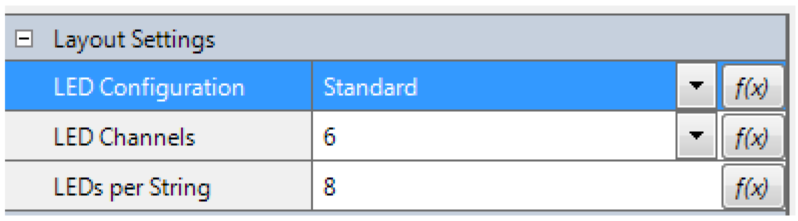


Figure 5 Example LED Configuration.



Layout Settings	
LED Configuration	Standard
LED Channels	6
LEDs per String	8

Figure 6 Standard Mode Parameters

LED Channels:

This determines how many rows of LED strips are connected. This value can be between 1 to 16.

LEDs per String:

The amount of LEDs on each strip. Each strip should have the same amount of LED modules.

Spiral Mode

The Spiral mode uses a single LED string wrapped around a tube in a spiral manor. By providing the number of LEDs in a single rotation around the tube, the number of rows and columns are calculated. It is important to use a tube diameter such that when the LED string is wrapped around the tube, the LEDs line up vertically. See figure below.

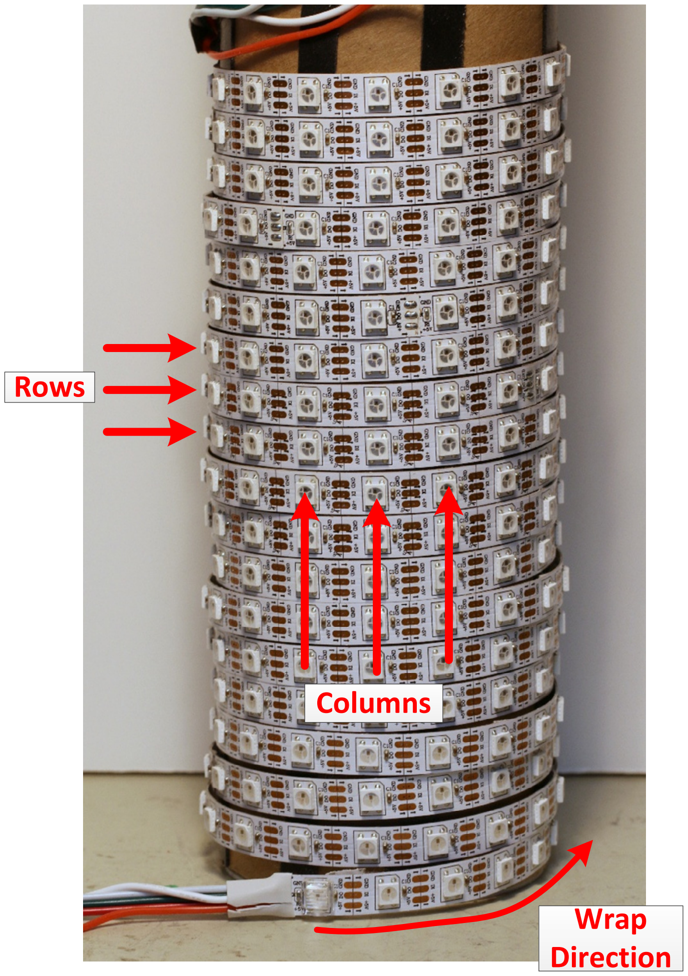
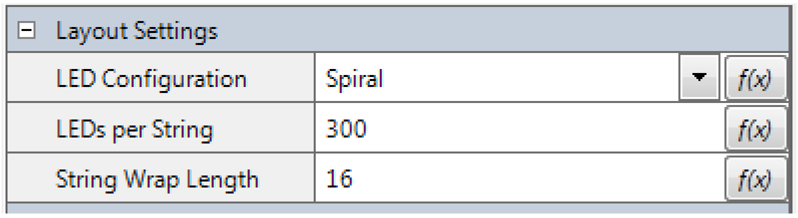


Figure 7 Example of how LED strip is wrapped around a tube.



Layout Settings	
LED Configuration	Spiral
LEDs per String	300
String Wrap Length	16

Figure 8 Spiral Mode Parameters.

LEDs per String

This is the total LED modules on the string.

String Wrap Length

The is the amount of LEDs in a single wrap around the tube. This number translates into the number of columns on the tube. The rows is the total number divided by the LEDs per String divided by the String Wrap Length.

Grid_16x16 Mode

This mode uses LED panels that are configured as a 16x16 Grid (256 LEDs total). These grids are configured in a zig-zag pattern but the firmware translates it as a simple X-Y grid. Also the firmware can handle up to 64 of these 16x16 panels, for a total of 65,536 LEDs (16x16x256) providing there is sufficient memory.

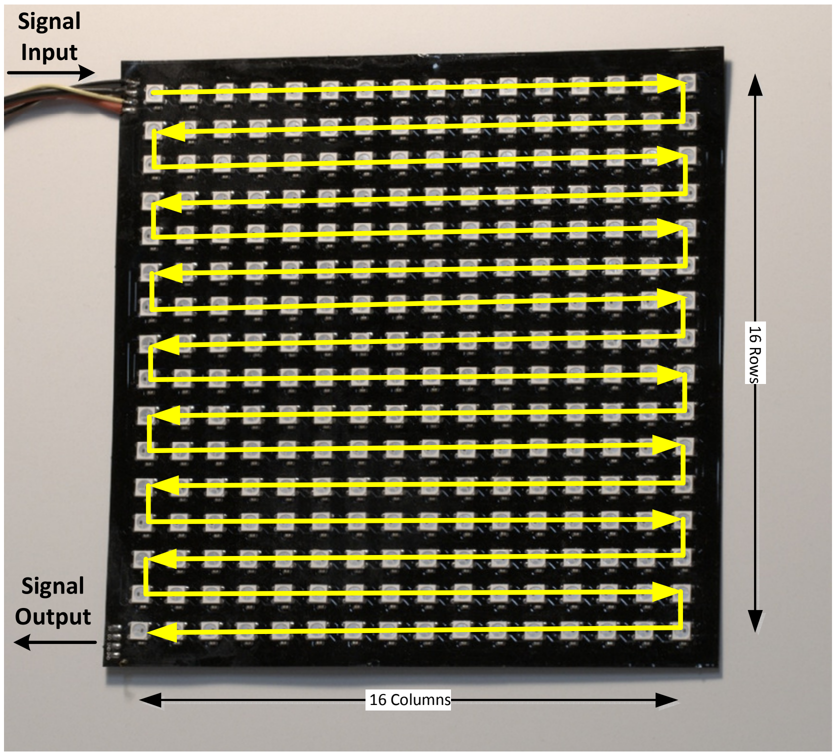


Figure 9 Example 16x16 Grid Panel

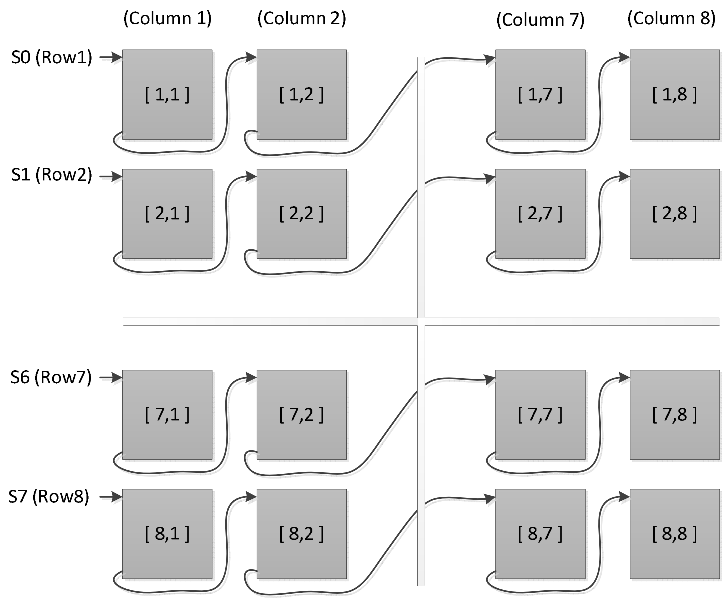
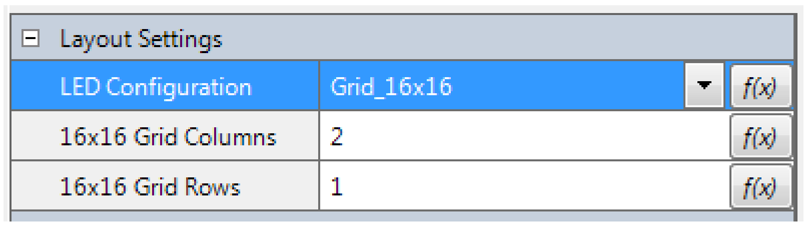


Figure 10 Layout of 64 (x8x) LED Grids



Layout Settings	
LED Configuration	Grid_16x16
16x16 Grid Columns	2
16x16 Grid Rows	1

Figure 11 Grid_16x16 LED Configuration

16x16 Grid Columns

The number of 16x16 panels in the X direction.

16x16 Grid Rows

The number of 16x16 panels in the Y direction.

Resources

The StripLights component requires between 2 to 4 UDBs depending on the channel count.

Flash: TBD

SRAM: One byte per RGB LED in LUT(8-bit) mode and four bytes per RGB LED in RGB (24-bit) mode plus stack overhead.

Application Programming Interface

Application Programming Interface (API) routines allow you to control and 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 "StripLights_1" to the first instance of a component in a given design. You can rename the instance to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "StripLights".

Function	Description
void StripLights_Start()	Configures and starts component.
void StripLights_Stop()	Disables interrupt.
void StripLights_DisplayClear()	Clears the screen with a given color.
void StripLights_MemClear()	Clear display memory, but don’t update display.
void StripLights_Trigger()	Update entire display with memory data.
Void StripLights_TriggerWait()	Update entire display with memory data, but do not return until entire display is updated.
void StripLights_Pixel()	Set the LED at a given (X,Y) location with the specified color
void StripLights_DrawLine()	Draw line of specified color from (x1,y1) to (x2,y2)
void StripLights_DrawRect()	Draw rectangle from upper corner (x1,y1) to lower corner (x2,y2)
void StripLights_DrawCircle()	Draw circle from point (x,y) with specified radius and color.
void StripLights_ColorInc()	Return next color in color wheel, with specified increment value.
void StripLights_Dim()	Reduce (Dim) intensity next triggered update by 1, ½, ¼, 1/8, or 1/16
void StripLights_Ready()	Check if display update is complete
void StripLights_SetFont()	Set font type (Not implemented at this time)
void StripLights_PutChar()	Place character at (x,y) with specified foreground and background color
void StripLights_PrintString()	Print string at (x,y) with specified foreground and background color
void StripLights_Bplot32()	Print a rectangular bit map to the display.

void StripLights_Start(void)

Description:	Initialize hardware, clear display memory, and enable interrupts.
Parameters:	None
Return Value:	None
Side Effects:	None

void StripLights_Stop(void)

Description:	Disable StripLights component.
Parameters:	None
Return Value:	None
Side Effects:	None

void StripLights_DisplayClear(uint32 color)

Description:	Clears the display memory with the given color, then updates the LEDs.
Parameters:	uint32 color: Color in which to clear the display.
Return Value:	None
Side Effects:	All display data will be overwritten by the given color and LEDs are updated.

void StripLights_MemClear(uint32 color)

Description:	Clears the display memory with the given color, but does not update the LEDs.
Parameters:	uint32 color: Color in which to clear the display.
Return Value:	None
Side Effects:	All display data will be overwritten by the given color.

void StripLights_Trigger(uint32 rst)

Description:	Starts the transfer of the display RAM to the LEDs. Use the StripLights_Ready() function to determine when entire transfer is complete.
Parameters:	rst: Resets the interface to start at the first LED module. (Usually set to 1)
Return Value:	None
Side Effects:	None

void StripLights_TriggerWait(uint32 rst)

Description: Starts the transfer of the display RAM to the LEDs. This function is blocking and will not return until the entire display has been updated.

Parameters: rst: Resets the interface to start at the first LED module.

Return Value: None

Side Effects: None

void StripLights_Pixel(uint32 x, uint32 y, uint32 color)

Description: Sets the pixel (LED module) at the specified X,Y location to the specified color.

Parameters: uint32 x: Horizontal position of the LED to modify
uint32 y: Vertical position of the LED to modify
uint32 color: Color to set the LED.

Return Value: None

Side Effects: None

void StripLights_DrawLine(int32 x0, int32 y0, int32 x1, int32 y1, uint32 color)

Description: Draws a line in the graphics memory from (x0,y0) to (x1,y1) with specified color.

Parameters: None

Return Value: int32 x0, y0: First endpoint of the line to be drawn.
int32 x1, y1: Second endpoint of the line to be drawn.
uint32 color: Color to set the LED.

Side Effects: None

void StripLights_DrawRect(int32 x0, int32 y0, int32 x1, int32 y1, int32 fill, uint32 color)

Description: Draws a rectangle in the graphics memory from upper left corner (x0,y0) to lower right corner (x1,y1) with specified color and fill option.

Parameters: None

Return Value: int32 x0, y0: First endpoint of the line to be drawn.
int32 x1, y1: Second endpoint of the line to be drawn.
int32 fill: Fill rectangle with given color, if fill is non-zero.
uint32 color: Color to set the LED.

Side Effects: None

void StripLights_DrawCircle(int32 x0, int32 y0, int32 radius, uint32 color)

Description: Draws a circle in the graphics memory with center at (x0,y0) with the given radius and color.

Parameters: int32 x0, y0: Location of center of circle.
uint32 radius: Radius of circle.
uint32 color: Color of circle.

Return Value: None.

Side Effects: None

void StripLights_ Dim (uint32 dimLevel)

Description: This function sets the entire display to the given dim level. This will not dim the display until the Trigger() function is called.

Parameters: uint32: Level to dim the LED display.

- 0: No dimming
- 1: Dim to 50%
- 2: Dim to 25%
- 3: Dim to 12.5%
- 4: Dim to 6.3%

Return Value: None

Side Effects: None

void StripLights_ PutChar (int32 xpos, int32 ypos, uint8 theChar, uint32 fgColor, uint32 bgColor)

Description: Prints one character into the display memory at position (xpos,ypos). Use Trigger() function to update the display.

Parameters: int32 xpos, ypos: Location of lower left corner of character.
uint8 theChar: Character to be printed.
uint32 fgColor: Color of character to be printed.
Uint32 bgColor: Background color of character to be printed.

Return Value: None

Side Effects: None

void StripLights_ PrintString (int32 xpos, int32 ypos, uint8 * theString, uint32 fgColor, uint32 bgColor)

Description: Prints a null terminated string to the display memory at the coordinates (xpos, ypos). Use Trigger() function to update the display.

Parameters: int32 xpos, ypos: Location of lower left corner of character.
uint8 * theString: Null terminated string to be printed.
uint32 fgColor: Color of character to be printed.
Uint32 bgColor: Background color of character to be printed.

Return Value: None

Side Effects: None

void StripLights_Bplot32(uint32 x, uint32 y, uint32 * bitmap, uint32 update)

Description: Displays a bit-map on the display. The first two words in the bitmap array are the xSize and ySize size respectively. The next xSize words is the first row of the bitmap, and the next xSize words is the next row, and so on.

Parameters: uint32 x: X coordinate to start the bitmap
uint32 y: Y coordinate to start the bitmap
uint32 * bitMap: Pointer to array of bitmap.
uint32 update: Set to 1 if display should be updated.

Return Value: None

Side Effects: None

void StripLights_RgbBlend(uint32 fromColor, uint32 toColor, uint32 pct)

Description: Averages the two colors toColor and fromColor and returns the blended color. The returned color equals (pct * toColor) + ((100-pct) * fromColor).

Parameters: uint32 fromColor: Original color, return value will equal this color when pct = 0;
uint32 toColor: This color will be returned when pct = 100.
uint32 pct: This value should be between 0 and 100. As this value goes from 0 to 100, the output color changes from fromColor to toColor.

Return Value: uint32: The blended color.

Side Effects: None

External components in library

A couple of external components are included in the library to augment the user's schematic.

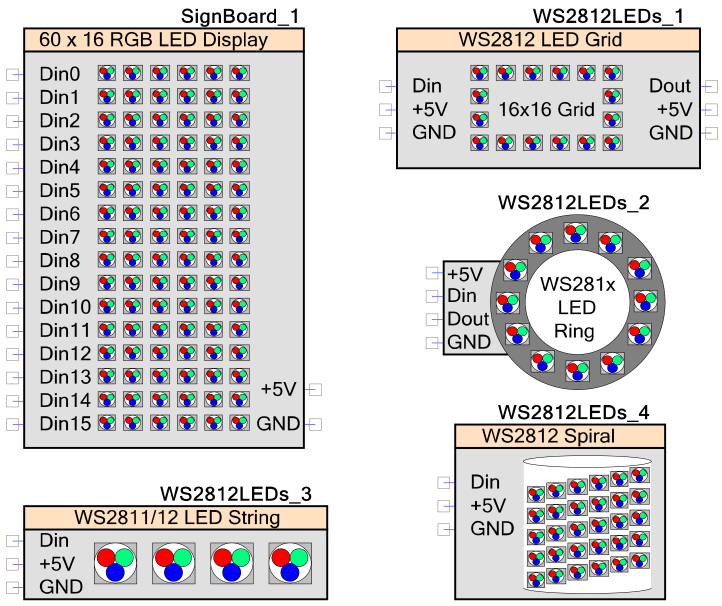


Figure 12 External Components Included

Table 1 Useful Constants

Color Constants	Description
StripLights_RED	Red
StripLights_GREEN	Green
StripLights_BLUE	Blue
StripLights_YELLOW	Yellow
StripLights_ORANGE	Orange
StripLights_LTBLUE	Light Blue
StripLights_MBLUE	Medium Blue
StripLights_LTGREEN	Light Green
StripLights_MGREEN	Medium Green
StripLights_LTRED	Light Red
StripLights_MGRED	Medium Red
StripLights_MAGENTA	Magenta
StripLights_TURQUOSE	Turquose
StripLights_CYAN	Cyan
StripLights_VIOLET	Violet
StripLights_DIM_WHITE	Dim White
StripLights_CWHEEL_SIZE	Size of Color wheel lookup table
StripLights_MIN_X	Minimum X coordinate (usually 0)
StripLights_MAX_X	Maximum X coordinate
StripLights_MIN_Y	Minimum Y coordinate (usually 0)
StripLights_MAX_Y	Maximum Y coordinate
StripLights_COLUMNS	Number of columns
StripLights_ROWS	Number of Rows
StripLights_TOTAL_LEDS	Total number of LEDs

DC and AC Electrical Characteristics

The component supports both 400 and 800 kHz interface. The device Vdd should always be 5V to interface properly to WS2811/12 devices. The System clock should be set to 24 MHz or highter.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.3		Initial document
2.8	- Added support for 4200M, 4200L, and BLE	
3.0	- Updated library to be compatible with PSoC Creator 4.1 SP1 - Updated to new default customizer. - Added support for single spiral string. - Added support for 16x16 LED panels. - Added X and Y coordinate flip - Added X-Y coordinate swap -Added Bplot() function to print bitmaps.	Must use PSoC Creator 4.1 More features == more fun

Component Author: Mark Hastings

This component is just funware. It was created independent of my employer Cypress Semiconductor and Cypress is not liable for any errors or defects that may be found in this component.