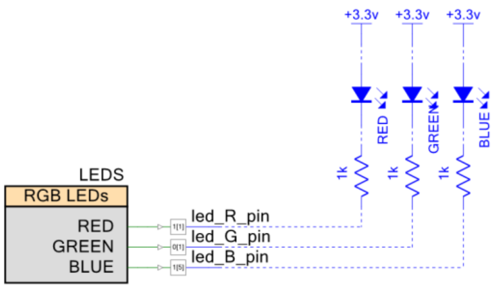
RGB LEDS

MICA

Controlling Leds

Features

- Control RGB LEDs through pin-by-pin basis
- Use software API for controlling colors



General Description

Users should use the RGB LEDs component when there is a Red, Green, and Blue LED connected to the PSoC. This API does not support hardware PWM blocks for driving the LEDs. There are two versions of the component: active high and active low. Active high should be used when the pin is connected to the anode of the LEDs, and active low should be used when the pin is connected to the cathode of the LEDs. Mixed high/low support is currently not available.

Input/Output Connections

This section describes all of the virtual connections for the RGB LEDs component.

RED – Output

This should be connected to the virtual pin that represents the red LED. For active high configurations the pin should be strong drive, for active low configurations, the pin should be open drain, drives low.

GREEN– Output

This should be connected to the virtual pin that represents the green LED. For active high configurations the pin should be strong drive, for active low configurations, the pin should be open drain, drives low.

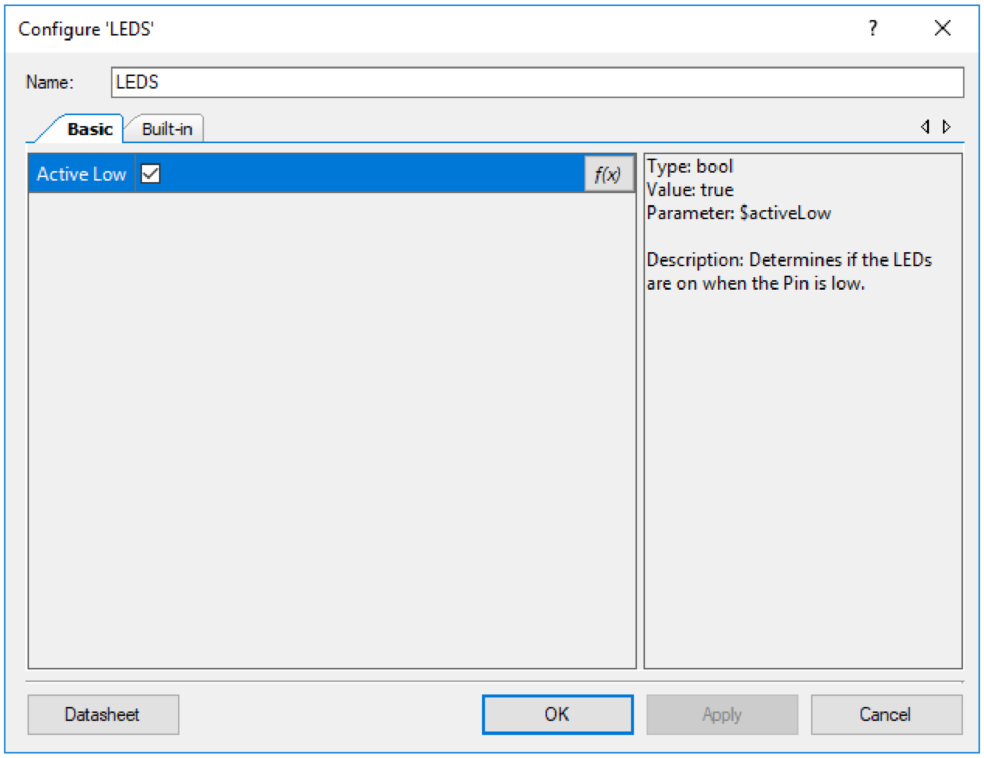
BLUE – Output

This should be connected to the virtual pin that represents the blue LED. For active high configurations the pin should be strong drive, for active low configurations, the pin should be open drain, drives low.

Component Parameters

Double click on the RGB LEDs component to open the Configure dialog.

Basic Tab



Type: bool
Value: true
Parameter: $activeLow

Description: Determines if the LEDs are on when the Pin is low.

The RGB LED Component has the following parameters.

Active Low

When selected, this indicates that the LEDs connected to the component will be on when a logical low is present on the hardware pin. It is suggested (but not required) that the virtual pin be designated Open Drain, Drives Low if this is selected.

Application Programming Interface (API)

API routines allows users to control the LEDs from software. The default name for the component is “LEDS”, which can be changed in the configure dialogue. All functions and constants are generated based on this name.

Functions

Function	Description
LEDS_Write()	Sets the state of the LED control register, setting the color of the LEDs on the board.
LEDS_R_Write()	Sets the state of the Red LED
LEDS_G_Write()	Sets the state of the Green LED
LEDS_B_Write()	Sets the state of the Blue LED
LEDS_R_Toggle()	Toggle the current state of the Red LED
LEDS_G_Toggle()	Toggle the current state of the Green LED
LEDS_B_Toggle()	Toggle the current state of the Blue LED
LEDS_Read()	Returns the current value of the LED control register
LEDS_R_Read()	Return the state of the Red LED
LEDS_G_Read()	Return the state of the Green LED
LEDS_B_Read()	Return the state of the Blue LED
LEDS_Sleep()	Prepares the component for sleep. Sleeps the control Register.
LEDS_Wakeup()	Wakes up the component from sleep. Wakes the control Register.
LEDS_Test()	Runs through the testing routine for the LEDs Expected Outcome: 0. All LEDs off 1. Red LED on 2. Green LED on 3. Blue LED on 4. Yellow (RB) 5. Cyan (BG) 6. Magenta (RG) 7. White (RGB) LEDs on

uint8 LEDS_Write(uint8 state)

Description:	Sets the state of the LED control register, setting the color of the LEDs on the board.
Parameters:	uint8: The new state mask to write to the control register. It is recommended that users only pass the color macros defined by the component.
Return Value:	uint8: Value of the control register after it was written.

uint8 LEDS_R_Write(bool state)

Description:	Sets the state of the Red LED to the state passed in, Regardless of the polarity of the LEDs. I.E. LEDS_R_Write(true) will always turn on the LED.
Parameters:	**State** - The new state to write.
Return Value:	uint8: Value of the LED control register after it was written.

uint8 LEDS_G_Write(bool state)

Description:	Sets the state of the Green LED to the state passed in, Regardless of the polarity of the LEDs. I.E. LEDS_G_Write(true) will always turn on the LED.
Parameters:	**State** - The new state to write.
Return Value:	uint8: Value of the LED control register after it was written.

uint8 LEDS_B_Write(bool state)

Description:	Sets the state of the Blue LED to the state passed in, Regardless of the polarity of the LEDs. I.E. LEDS_B_Write(true) will always turn on the LED.
Parameters:	**State** - The new state to write.
Return Value:	uint8: Value of the LED control register after it was written.

void LEDS_R_Toggle(void)

Description:	Toggles the state of the Red LED. Implemented with a Function-like Macro.

void LEDS_G_Toggle(void)

Description:	Toggles the state of the Green LED. Implemented with a Function-like Macro.

void LEDS_B_Toggle(void)

Description:	Toggles the state of the Blue LED. Implemented with a Function-like Macro.

uint8 LEDS_Read(void)

Description:	Returns the current value of the LED control register
Return Value:	uint8: Value held in the LED control register

bool LEDS_R_Read(void)

Description:	Return the current state of the Red LED
Return Value:	State of the Red LED. True if on, False if off, regardless of LED Polarity

bool LEDS_G_Read(void)

Description:	Return the current state of the Green LED
Return Value:	State of the Green LED. True if on, False if off, regardless of LED Polarity

bool LEDS_B_Read(void)

Description:	Return the current state of the Blue LED
Return Value:	State of the Blue LED. True if on, False if off, regardless of LED Polarity

void LEDS_Sleep(void)

Description:	Prepares the component for sleep. Sleeps the LED control Register
Side Effects:	Places the LED control registers in sleep, it does not however, change the status of the virtual pins. I.e. an active low virtual pin will remain open drain drives low, and not Hi-Z.

void LEDS_Wakeup(void)

Description:	Wakes up the component from sleep. Wakes the control Register.
Side Effects:	Sleep does not change the status of the virtual pins. I.e. a pin that has been placed in Hi-Z will not be set back to open drain drives low.

void LEDS_Test(uint8 runs)

Description: Runs through the board level testing routine for the LEDs.

Parameters: uint8: The number of times to iterate through the test. 0 means infinite.

Expected Outcome:

0. All LEDs off
1. Red LED on
2. Green LED on
3. Blue LED on
4. Yellow (RB)
5. Cyan (BG)
6. Magenta (RG)
7. White (RGB) LEDs on

Component Macros

The following is a list of macros/constants that a user may find useful for interacting with the component. A component may contain macros not listed here.

Macro Name	Description
LEDS_ON	Turns on an LED when passed to LEDS_X_Write(). Equates to true.
LEDS_OFF	Turns off an LED when passed to LEDS_X_Write(). Equates to false.
LEDS_ON_NONE	Value that corresponds to no LEDs being on when passed into LEDS_Write() .
LEDS_ON_RED	Value that corresponds to the red LED being on when passed into LEDS_Write() .
LEDS_ON_GREEN	Value that corresponds to the green LED being on when passed into LEDS_Write() .
LEDS_ON_BLUE	Value that corresponds to the blue LED being on when passed into LEDS_Write() .
LEDS_ON_YELLOW	Value that corresponds to a yellow (red & green) LED being on when passed into LEDS_Write() .
LEDS_ON_MAGENTA	Value that corresponds to a magenta (red & blue) LED being on when passed into LEDS_Write() .
LEDS_ON_CYAN	Value that corresponds to a cyan (blue & green) LED being on when passed into LEDS_Write() .
LEDS_ON_WHITE	Value that corresponds to a white (red, blue & green) LED being on when passed into LEDS_Write() .

Component Resources

The RGB LED block consumes one Control Register.

Change Log

This sections lists changes to the component from previous versions

Version	Revision	Description of Changes	Reason for Changes / Impact
v1.0	r3	Added individual read, write and toggle functions	Expand functionality.
	r2	Update datasheet, color assignments	Added Component Macros section, fixed mistake in magenta, cyan definitions.
	r1	Update datasheet	Updated MICA component template.
	r0	Initial implementation of the component and datasheet	