LED Datasheet LED V 2.00

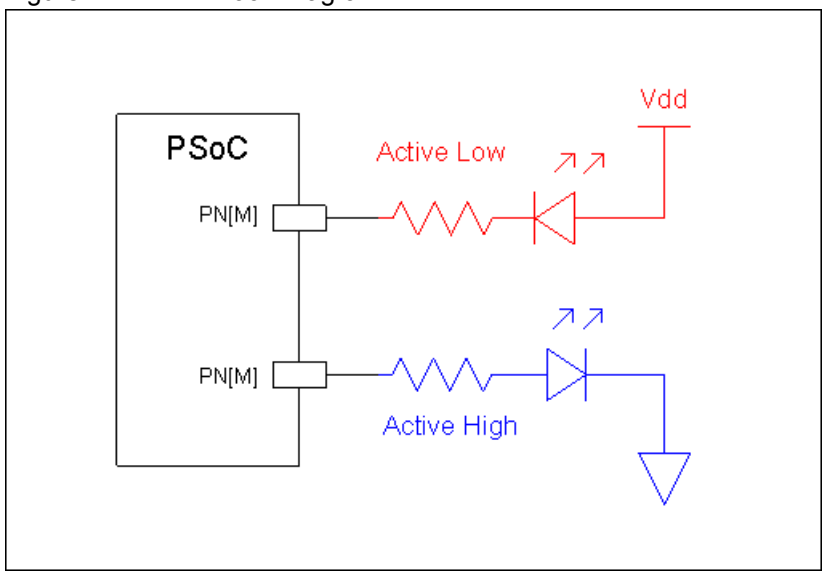
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
All PSoC Devices	0	0	0	40	1	1

Features and Overview

- Support for both Active High and Active Low circuits
- Works with system shadow registers
- Functions (Switch, Invert, and GetState)

The LED User Module is just a few functions to control an LED or any simple device that is controlled by On and Off.

Figure 1. LED Block Diagram



Functional Description

The LED User Module is a convenient way to turn a pin on and off without having to think about shadow registers.

The project using the LED User Module which manipulates pins on a port shared with an instance of the LED User Module must avoid direct PRTxDR writes. The Shadow Registers should be used for such manipulation to prevent incorrect LED User Module operation.

Placement

The LED can be placed at any I/O pin. The Drive Mode of selected pin is automatically set to the Strong value.

Parameters and Resources

Port

Select the port where the LED will be connected.

Pin

Select pin where the LED will be connected.

Drive

The LED User Module may be configured to drive either an “Active High" or “Active Low" configuration. In an Active High configuration, the LED’s anode is connected to the PSoC pin through a resistor and the LED’s cathode is connected to Vss. In an Active Low configuration, the LED’s cathode is connected to the PSoC pin through a resistor and the LED’s anode is connected to Vcc.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the “include" files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This “registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they may do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Here are the API programming routines provided for the LED User Module:

LED_Start, LED_Stop

Description:

Both these functions do the same thing; turn off the LED.

C Prototype:

```
void  LED_Start(void)
void  LED_Stop(void)
```

Assembler:

```
lcall  LED_Start
lcall  LED_Stop
```

Return Value:

None

Side Effects:

None

LED_Switch

Description:

Turns LED on or off.

C Prototype:

```
void  LED_Switch(BYTE bOnOff)
```

Assembler:

```
Mov  a,0x01    ; Turn on LED
lcall   LED_Switch
```

Parameters:

bOnOff: 0 = Off, Non-Zero = On

Return Values:

None

Side Effects:

None

LED_On

Description:

Turns LED on.

C Prototype:

```
void  LED_On(void)
```

Assembler:

```
lcall   LED_On
```

Parameters:

None

Return Values:

None

Side Effects:

None

LED_Off

Description:

Turns LED off.

C Prototype:

```
void  LED_Off(void)
```

Assembler:

```
lcall   LED_Off
```

Parameters:

None

Return Values:

None

Side Effects:

None

LED_Invert

Description:

Inverts the state of the LED. If the LED was on, it will be turned off; if it was off, it will be turned on.

C Prototype:

```
void  LED_Invert(void)
```

Assembler:

```
lcall  LED_Invert
```

Parameters:

None

Return Values:

None

Side Effects:

None

LED_GetState

Description:

Returns state of LED.

C Prototype:

```
BYTE  LED_GetState(void)
```

Assembler:

```
lcall  LED_GetState
mov   [myLED_State],A  ; Place result in location myLED_State
```

Parameters:

None

Return Value:

Returns state of LED. A zero is returned if LED if off. A 1 is returned if the LED is on. Below are some symbolic names available in both C and ASM.

Symbolic Name	Value
LED_ON	1
LED_OFF	0

Side Effects:

None

Sample Firmware Source Code

Here is a sample project written in assembly code:

```
;;; Sample ASM Code for the LED User Module
;;;
;;; Turn off LED at start up then turn it on, WOW!
;;;

include "m8c.inc"       ; part specific constants and macros
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

area text(ROM,REL)
export _main

_main:

   mov  a,0x00          ; Turn Off LED
   call  LED_Switch
                        ; Do user stuff
   mov  A,0x01          ; Turn On LED
   call  LED_Switch
```

Here is a sample project written in C:

```
//-------------------------------------------------------
//  Sample C Code for the LED
//  Turn LED off then, then stay in a loop and invert
//  its state.
//
//
//-------------------------------------------------------------------------
#include <m8c.h>          // part specific constants and macros
#include "PSoCAPI.h"      // PSoC API definitions for all User

void main(void)
{
   LED_Start();

   LED_Switch(1);         // Turn on LED

   while(1) {
      LED_Invert();       // Flash LED
}
}
```

Configuration Registers

None

Version History

Version	Originator	Description
1.2	DHA	Added Version History.
1.3	DHA	Enhanced pin selection to exclude used pins.
1.40	DHA	Added DRC warning message. This message appears when several LED User Modules use the same pin.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.