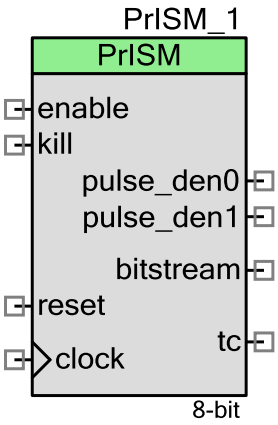
Precision Illumination Signal Modulation (PrISM)

1.50

Features



- Programmable flicker-free dimming resolution from 2- to 32-bit
- Two pulse density outputs
- Programmable output signal density
- Serial output bit stream
- Continuous run mode
- User configurable sequence start value
- Standard or custom polynomials provided for all sequence lengths
- Kill input disables pulse density outputs and forces them low
- Enable input provides synchronized operation with other components
- Reset input allows restart at sequence start value for synchronization with other components
- Terminal Count Output for 8-, 16-, 24- and 32-bit sequence lengths

General Description

The Precision Illumination Signal Modulation (PrISM) component uses a linear feedback shift register (LFSR) to generate a pseudo random sequence. The sequence outputs a pseudo random bit stream, as well as up to two user-adjustable pseudo random pulse densities. The pulse densities may range from 0 to 100%.

The LFSR is of the Galois form (sometimes known as the modular form) and utilizes the provided maximal length codes. The PrISM component runs continuously after started and as long as the enable input is held high. The PrISM pseudo random number generator may be started with any valid seed value, excluding 0.

When to use a PrISM

The PrISM component provides modulation technology that significantly reduces low-frequency flicker and radiated electro-magnetic interference (EMI), which are common problems with high

brightness LED designs. The PrISM is also useful in other applications requiring this benefit, such as motor controls and power supplies.

Input/Output Connections

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

clock – Input

The clock input defines the signal to compute Pseudo Random Sequence.

reset – Input

The reset input resets the pseudo random number to the start value at high state. This input valid for started component only and provides synchronized operation with other components.

kill – Input

The active high kill input disables the PrISM pulse density outputs and sets them to 0 untill kill is released low.

enable – Input

The PrISM component runs after started and as long as the enable input is held high and reset input is low. This input provides synchronized operation with other components.

pulse_den0 / pulse_den1 – Outputs

Two pulse density outputs are available; both are derived from the same pseudo random sequence. Each output is generated by comparing the desired pulse density value with the current pseudo random number. If the pulse density type is configured as "Less Than Or Equal To," the output is high while the pseudo random number is less than or equal to the pulse density value. The second option is set to the pulse density type of "Greater Than Or Equal To" and the output is high while the pseudo random number is greater than or equal to the pulse density value.

bitstream – Output

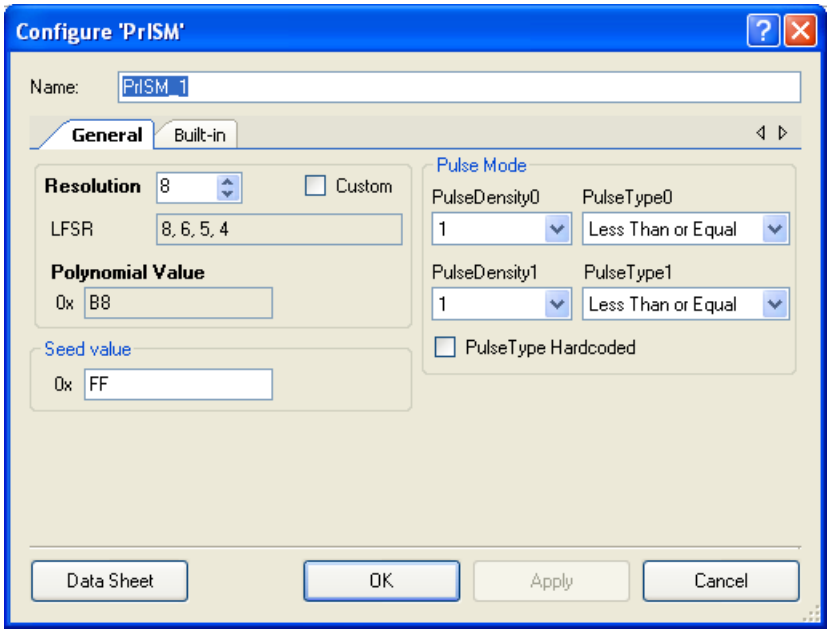
The Bitstream output continuously outputs the LSb of the LFSR.

tc – Output *

The terminal count output is available for 8, 16, 24 and 32-bit length PrISM components. The terminal count output goes high for 1 clock period each time the pseudo random number equals 0xFF (8-bit), 0xFFFF (16-bit), 0xFFFFFF (24-bit) or 0xFFFFFFFF (32-bit) which occurs once each cycle of the pseudo random number generator.

Parameters and Setup

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



The PrISM component contains the following parameters:

Resolution

This parameter defines the PrISM maximal code length (period). The maximal code length is ($2^{Resolution} – 1$). Possible values include 2 - 32 bits. The maximal length code sets the length of the pseudo random number generator and therefore the length of the sequence to be generated. Longer sequences increase the pulse density resolution and lower the radiated EMI. The maximal length codes listed in the following table are provided in the Galois form and require no conversion prior to use in the PSoC 3 UDB ALU.

Resolution	LFSR
2	2, 1

Resolution	LFSR
13	13, 12, 10, 9

Resolution	LFSR
24	24, 23, 21, 20

3	3, 2	14	14, 13, 11, 9	25	25, 24, 23, 22
4	4, 3	15	15, 14, 13, 11	26	26, 25, 24, 20
5	5, 4, 3, 2	16	16, 14, 13, 11	27	27, 26, 25, 22
6	6, 5, 3, 2	17	17, 16, 15, 14	28	28, 27, 24, 22
7	7, 6, 5, 4	18	18, 17, 16, 13	29	29, 28, 27, 25
8	8, 6, 5, 4	19	19, 18, 17, 14	30	30, 29, 26, 24
9	9, 8, 6, 5	20	20, 19, 16, 14	31	31, 30, 29, 28
10	10, 9, 7, 6	21	21, 20, 19, 16	32	32, 30, 26, 25
11	11, 10, 9, 7	22	22, 19, 18, 17		
12	12, 11, 8, 6	23	23, 22, 20, 18		

Polynomial Value

This parameter is represented in the hexadecimal form. The correct polynomial is chosen based on the **Resolution** selected. You may optionally specify a custom polynomial.

Seed Value

This parameter by default is set to the maximum possible value ($2^{Resolution} – 1$). This value can be changed to any value except 0. The **Seed value** is represented in the hexadecimal form.

Warning Changing the **Resolution** causes the **Seed value** to be set to the default value.

Pulse Mode

These parameter values are chosen from combo boxes. Available values are from 1 to $2^{Resolution}$ - 1 with a step $2^{Resolution}$. Pulse compare type can be selected "Less Than Or Equal To" or "Greater Than Or Equal To".

PulseType Hardcoded

The **PulseType Hardcoded** parameter saves recourses (control register) when enabled, but prohibits the possibility to change the Pulse Type using the PrISM_SetPulse0Mode() or PrISM_SetPulse1Mode() APIs.

The PrISM_Stop() function also is not available if this parameter is enabled. To stop the PrISM in this case, use the "enable" input.

Local Parameters (For API usage)

These parameters are used in the API and not shown in the Configure dialog; however, these are used in the APIs.

- **PolyValue (uint32) –** Contains polynomial value in hexadecimal form. The default is 0xB8h (LFSR= [8,6,5,4]).
- **Density0(uint32) –** Contains density0 value in hexadecimal form.
- **Density1(uint32) –** Contains density1 value in hexadecimal form.
- **CompareType0(CompareType) –** Contains Pulse Type for Density0 which may be "Less Than or Equal" or "Greater Than or Equal".
- **CompareType1 (CompareType) –** Contains Pulse Type for Density1 which may be "Less Than or Equal" or "Greater Than or Equal".

Clock Selection

There is no internal clock in this component. You must attach a clock source. The maximum frequency input is 67 MHz.

Placement

The PrISM is placed throughout the UDB array and all placement information is provided to the API through the *cyfitter.h* file.

Resources

Resolution	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter 7	Flash	RAM	
8-Bits	1	4	0	1	0	TBD	TBD	TBD
8-Bits *	1	4	0	0	0	TBD	TBD	TBD
16-Bits	2	4	0	1	0	TBD	TBD	TBD
24-Bits	3	4	0	1	0	TBD	TBD	TBD
32-Bits	4	4	0	1	0	TBD	TBD	TBD

* Parameter **PulseType Hardcoded** enabled.

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 "PrISM_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 "PrISM".

Function	Description
void PrISM_Init(void)	Initializes default configuration provided with customizer.
void PrISM_Enable(void)	Enables the PrISM block operation.
void PrISM_Start(void)	The start function sets Polynomial, Seed and Pulse Density registers provided by customizer.
void PrISM_Stop(void)	Stops PrISM computation.
void PrISM_SetPulse0Mode (unit8 Pulse0Type)	Sets the pulse density type for Density0.

Function	Description
void PrISM_SetPulse1Mode (unit8 Pulse1Type)	Sets the pulse density type for Density1.
void PrISM_ReadSeed(void)	Reads the PrISM Seed register.
void PrISM_WriteSeed(unit8/16/32 Seed)	Writes the PrISM Seed register with the start value.
void PrISM_ReadPolynomial(void)	Reads the PrISM Polynomial register.
void PrISM_WritePolynomial(unit8/16/32 Polynomial)	Writes the PrISM Polynomial register with the start value.
unit8/16/32 PrISM_ReadPulse0(void)	Reads the PrISM Pulse Density0 value register.
void PrISM_WritePulse0(unit8/16/32 PulseDensity0)	Writes the PrISM Pulse Density0 value register with the new Pulse Density value.
unit8/16/32 PrISM_ReadPulse1(void)	Reads the PrISM Pulse Density1 value register.
void PrISM_WritePulse1(unit8/16/32 PulseDensity1)	Writes the PrISM Pulse Density1 value register with the new Pulse Density value.
void PrISM_Sleep(void)	Stops and saves the user configuration.
void PrISM_Wakeup(void)	Restores and enables the user configuration
void PrISM_SaveConfig(void)	Saves the current user configuration.
void PrISM_RestoreConfig(void)	Restores the current user configuration.

Global Variables

Variable	Description
PrISM_initVar	Indicates whether the PrISM has been initialized. The variable is initialized to 0 and set to 1 the first time PrISM_Start() is called. This allows the component to restart without reinitialization after the first call to the PrISM_Start() routine. If reinitialization of the component is required, then the PrISM_Init() function can be called before the PrISM_Start() or PrISM_Enable() functions.

void PrISM_Init(void)

Description: Initializes component's parameters and variables to the parameters set by user in the customizer of the component placed onto schematic. Usually called in PrISM_Start().

Parameters: None

Return Value: None

Side Effects: None

void PrISM_Enable(void)

Description: Enables the PrISM block operation.

Parameters: None

Return Value: None

Side Effects: None

void PrISM_Start(void)

Description: The start function sets polynomial, seed and pulse density registers provided by customizer. PrISM computation starts on rising edge of input clock.

Parameters: None

Return Value: None

Side Effects: None

void PrISM_Stop(void)

Description: Stops PrISM computation. Outputs remain constant.

Parameters: None

Return Value: None

Side Effects: Valid only if PulseType Hardcoded parameter disabled.

void PrISM_SetPulse0Mode(unit8 pulse0Type)

Description: Sets the pulse density type for Density0. Less than or Equal(<=) or Greater that or Equal(>=).

Parameters: unit8 pulse0Type: Selected pulse density type.

Parameters Value	Description
PrISM_1_LESSTHAN_OR_EQUAL	The pulse_den0 output is high when the pseudo random number is less than or equal to the PulseDensity0 register value
PrISM_1_GREATERTHAN_OR_EQUAL	The pulse_den0 output is high when the pseudo random number is greater than or equal to the PulseDensity0 register value

Return Value: None

Side Effects: Valid only if PulseType Hardcoded parameter disabled.

void PrISM_SetPulse1Mode(unit8 pulse1Type)

Description: Sets the pulse density type for Density1. Less than or Equal(<=) or Greater that or Equal(>=).

Parameters: unit8 pulse1Type: Selected pulse density type.

Parameters Value	Description
PrISM_1_LESSTHAN_OR_EQUAL	The pulse_den1 output is high when the pseudo random number is less than or equal to the PulseDensity1 register value
PrISM_1_GREATERTHAN_OR_EQUAL	The pulse_den1 output is high when the pseudo random number is greater than or equal to the PulseDensity1 register value

Return Value: None

Side Effects: Valid only if PulseType Hardcoded parameter disabled.

(uint8/16/32) PrISM_ReadSeed (void)

Description: Reads the PrISM Seed register.

Parameters: None

Return Value: (uint8/16/32) Seed register value.

Side Effects: None

void PrISM_WriteSeed (uint8/16/32 seed)

Description: Writes the PrISM Seed register with the start value.

Parameters: (unit8/16/32) seed: seed register value.

Return Value: None

Side Effects: None

(uint8/16/32) PrISM_ReadPolynomial (void)

Description: Reads the PrISM polynomial.

Parameters: None

Return Value: (uint8/16/32) value of the polynomial.

Side Effects: None

void PrISM_WritePolynomial (uint8/16/32 polynomial)

Description: Writes the PrISM polynomial.

Parameters: (unit8/16/32) polynomial: polynomial register value.

Return Value: None

Side Effects: None

(uint8/16/32) PrISM_ReadPulse0 (void)

Description: Reads the PrISM Pulse Density0 value register.

Parameters: None

Return Value: (uint8/16/32) Pulse Density0 register value.

Side Effects: None

void PrISM_WritePulse0 (uint8/16/32 pulseDensity0)

Description: Writes the PrISM Pulse Density0 value register with the new Pulse Density value.

Parameters: (unit8/16/32) pulseDensity0: Pulse Density value.

Return Value: None

Side Effects: None

(uint8/16/32) PrISM_ReadPulse1 (void)

Description: Reads the PrISM Pulse Density1 value register.

Parameters: None

Return Value: (uint8/16/32) Pulse Density1 register value.

Side Effects: None

void PrISM_WritePulse1 (uint8/16/32 pulseDensity1)

Description: Writes the PrISM Pulse Density1 value register with the new Pulse Density value.

Parameters: (unit8/16/32) pulseDensity1: Pulse Density value.

Return Value: None

Side Effects: None

void PrISM_Sleep(void)

Description: Stops the PrISM operation and saves the user configuration. Should be called just prior to entering sleep.

Parameters: None

Return Value: None

Side Effects: None

void PrISM_Wakeup(void)

Description: Restores and enables the user configuration. Should be called just after awaking from sleep.

Parameters: None

Return Value: None

Side Effects: None

void PrISM_SaveConfig(void)

Description: Saves the current user configuration.

Parameters: None

Return Value: None

Side Effects: None

void PrISM_RestoreConfig(void)

Description: Restores the current user configuration.

Parameters: None

Return Value: None

Side Effects: None

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the PrISM component. This example assumes the component has been placed in a design with the default name PrISM_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

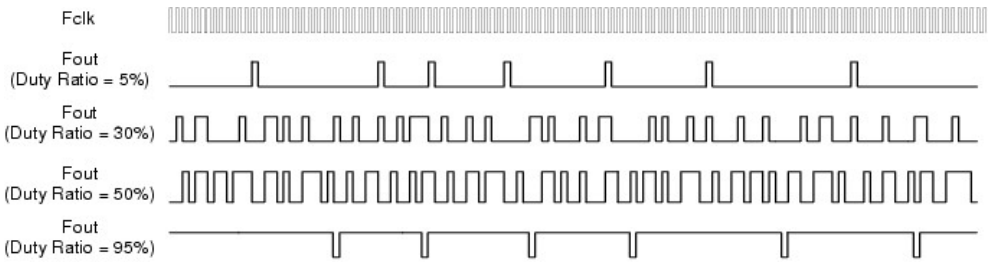
```
#include <device.h>
void main()
{
    PrISM_1_Start();
    PrISM_1_SetPulse0Mode(PrISM_1_LESSTHAN_OR_EQUAL);
    PrISM_1_SetPulse1Mode(PrISM_1_GREATERTHAN_OR_EQUAL);
    PrISM_1_WriteSeed(0xFF);
    PrISM_1_WritePulse0(0x80);
    PrISM_1_WritePulse1(0x80);
}
```

Functional Description

The PrISM component runs continuously after started and as long as the "enable" input is held high. The PrISM pseudo random number generator may be started with any valid value excluding 0 allowing multiple PrISM components to run out of phase of each other to further reduce EMI. The "reset" Input, resets the pseudo random number to the start value. The active high "kill" input disables the PrISM pulse density outputs and sets them to 0 until kill is released low. The "bitstream" output continuously outputs the LSb of the LFSR.

Two Pulse Density outputs are available; both are derived from the same Pseudo Random Sequence. Each output is generating by comparing the desired pulse density value with the current pseudo random number.

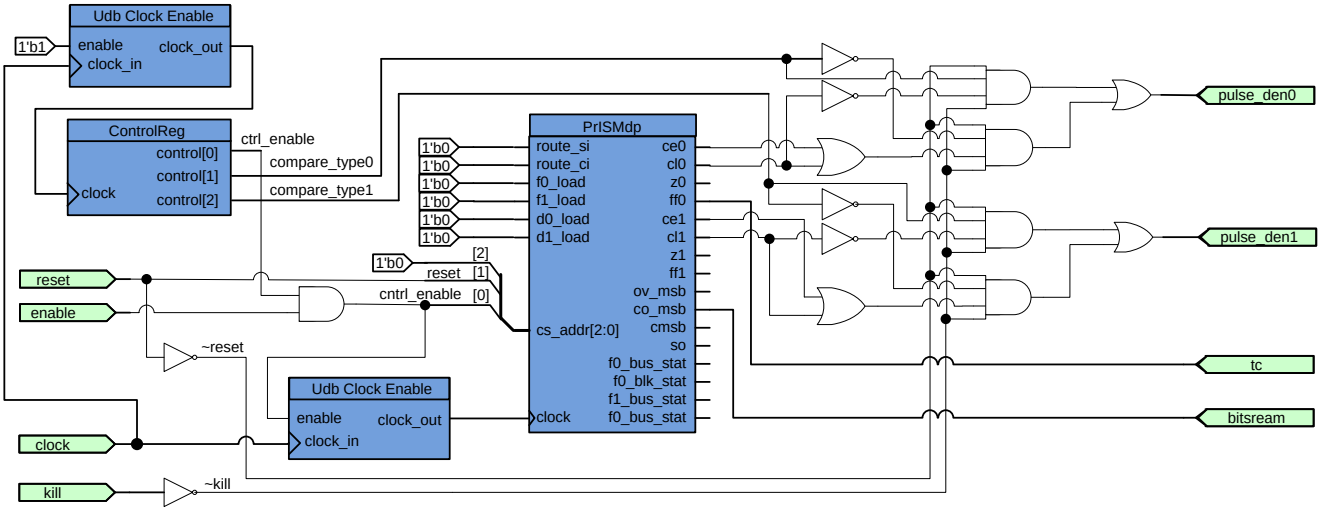
The following timing diagram shows the PrISM output based on several pulse density ratios.



Block Diagram and Configuration

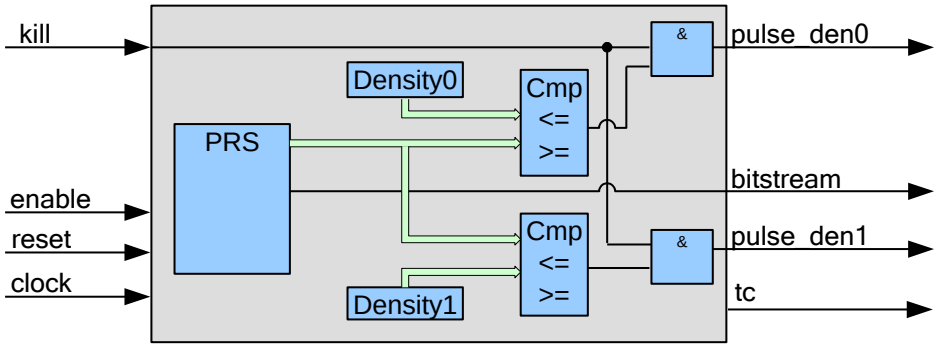
The PrISM is only available as a UDB configuration. The API is described above and the registers are described here to define the overall implementation of the PrISM.

The implementation is described in the following block diagram.



Top Level Architecture

The 2 to 32-Bit Hardware PrISM component compares the output of a pseudo-random counter with a signal density value. The comparator output asserts when the count value is less than (or greater than) and equal to the value in the Density value register.



Registers

PrISM_CONTROL

Bits	7	6	5	4	3	2	1	0
Value	reserved					compare type1	compare type0	ctrl enable

- ctrl enable: This bit enables generation of all internal signals described in the previous sections. The value can be changed by PrISM_Start() and PrISM_Stop() APIs.
- compare type0: This bit performs compare type for pulse_den0 output. The value of this bit is determined by the choice made for the pulse compare type parameter in the component Configure dialog. Also, the value can be changed by the PrISM_SetPulse0Mode(uint8 Pulse0Type) API.
- compare type1: This bit performs compare type for pulse_den1 output. The value of this bit is determined by the choice made for the pulse compare type parameter in the component Configure dialog. Also value can be changed by PrISM_SetPulse1Mode(uint8 Pulse1Type) API.

The control register is not used if the **PulseType Hardcoded** option is selected.

PrISM_SEED

Bits	7	6	5	4	3	2	1	0
Value	Seed							

- Seed: Contains the initial **Seed value** and PRS residual value at the end of the computation. The value of this register is determined by the **Seed value** parameter in the component Configure dialog. Also the value can be changed by the PrISM_WriteSeed(uint8/16/32) API and can be read by uint8/16/32 PrISM_ReadSeed() API.

PrISM_SEED_COPY

Bits	7	6	5	4	3	2	1	0
Value	Seed_Copy							

- Seed_Copy: Contains the start **Seed value** for automatically loading PrISM_SEED register when reset Input is active. The value of this register is determined by the **Seed value** parameter in the component Configure dialog and automatically updates if the PrISM_WriteSeed(uint8/16/32) API is used.

PrISM_POLYNOM

Bits	7	6	5	4	3	2	1	0
Value	Polynomial							

- Polynomial: The correct polynomial chosen based on Resolution selected. The value can be changed by the PrISM_WritePolynomial(uint8/16/32) API and can be read by the uint8/16/32 PrISM_ReadPolynomial() API.

PrISM_DENSITY0

Bits	7	6	5	4	3	2	1	0
Value	Pulse density0							

- Pulse density0 determines the value for the PrISM pulse_den0 output. The value of this register is determined by the **PulseDensity0** parameter in the Configure dialog. This value can be changed by the PrISM_WritePulse0 (uint8/16/32 PulseDensity0) API.

PrISM_DENSITY1

Bits	7	6	5	4	3	2	1	0
Value	Pulse density1							

- Pulse density1 determines the value for the PrISM pulse_den1 output. The value of this register is determined by the **PulseDensity1** parameter in the Configure dialog. This value can be changed by the PrISM_WritePulse1 (uint8/16/32 PulseDensity1) API.

References

Refer also to the PRS component data sheet.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Maximum Clock Rate	---		67	MHz	

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.50.b	Minor datasheet edit.	
1.50.a	Minor datasheet edit.	
1.50	Added Sleep/Wakeup and Init/Enable APIs.	To support low power modes, as well as to provide common interfaces to separate control of initialization and enabling of most components.
	TC output hides when resolution other than 8-, 16-, 24- or 32-bit selected.	TC output available for mentioned resolutions only.

© Cypress Semiconductor Corporation, 2009-2016. 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, 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.

PRELIMINARY