16-Bit Hardware PrISM Datasheet PrISM16HW V 1.0

Copyright © 2009-2014 Cypress Semiconductor Corporation. All Rights Reserved.

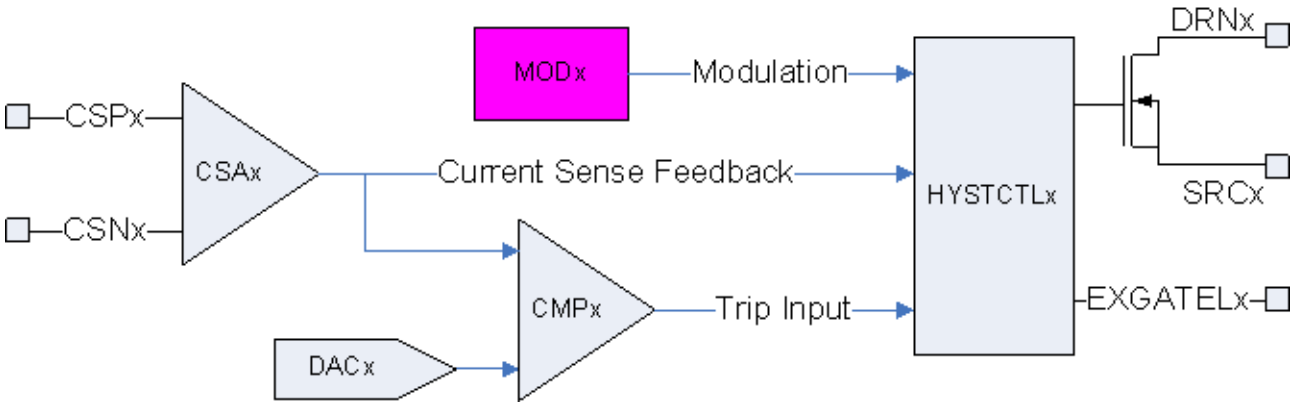
Resources	PSoC® Blocks	API Memory (Bytes)		Pins (per External I/O)
	Digital	Flash	RAM	
CY8CLED0xD, CY8CLED0xG	1	254	6	1

Features and Overview

- Programmable flicker-free dimming resolution from 2 to 16 bits
- Dedicated PrISM User Module enables you to use PSoC core digital blocks for other needs
- Programmable output signal density
- Selectable software output frequency compensation
- Reduced EMI

The 16-Bit Hardware Precise Intensity Signal Modulation (PrISM™) User Module 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 less than or equal to) the value in the Signal Density register.

Figure 1. PrISM16HW Block Diagram



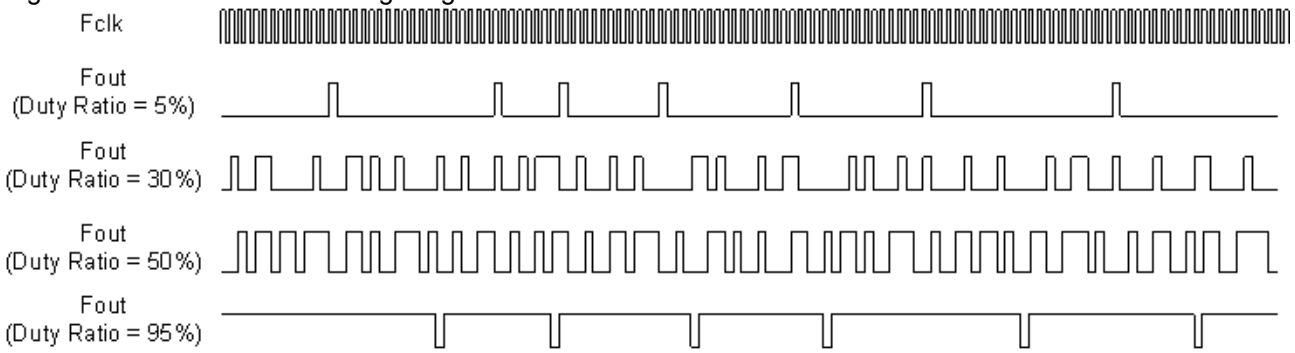
The application note AN47372 - PowerPSoC - PrISM(TM) Technology for LED Dimming addresses the challenges faced in implementing high resolution PrISM and recommends solutions to address these issues.

Functional Description

The PrISM16HW User Module produces a stochastic signal density modulation. It uses a pseudo-random counter that is implemented as a linear feedback shift register (LFSR). The counter operates in the range of [1..$2^{DimmingResolution}$-1]. Zero is the missing value. The pseudorandom count is compared with the Signal Density value. If the random value is less than (or less than or equal to) the signal density value, the comparator outputs a high level. This principle of operation is very similar to a standard Pulse Width Modulator (PWM). In both cases the counter generates a similar set of binary numbers from Period to zero (PWM) or from Period to one (PrISM). However, a PWM produces a linearly ordered series of numbers where PrISM produces the numbers in random order. That is why the PrISM16HW output signal looks like series of random pulses.

The following time diagram shows PrISM16HW dimming frequency output depending on the duty ratio.

Figure 2. PRISM16HW Timing Diagram



The following table lists modular irreducible polynomials for the pseudo-random counter. These polynomials define 2 to 16-bit pseudo-random number sequences (including 8, 16 bits). Note that the table is not complete. Only one of the many polynomials is specified. This table is for reference purposes. The polynomial is set automatically by the PrISM16HW User Module.

Table 1. Modular Irreducible Polynomials (Continued)

Number of Bits	Code Length	Modular Form Polynomial	Polynomial Value
2	3	[2,1]	0x03
3	7	[3,2]	0x06
4	15	[4,3]	0x0C
5	31	[5,4,3,2]	0x1E
6	63	[6,5,4,1]	0x39
7	127	[7,6,5,2]	0x72
8	255	[8,6,5,4]	0xB8
9	511	[9,6,5,3]	0x134
10	1023	[10,8,7,2]	0x2C2
11	2047	[11,9,6,3]	0x524
12	4095	[12,11,8,6]	0xCA0

Number of Bits	Code Length	Modular Form Polynomial	Polynomial Value
13	8191	[13,12,10,9]	0x1B00
14	16383	[14,13,12,2]	0x3802
15	32767	[15,13,10,8]	0x5280
16	65535	[16,15,13,4]	0xD008

Signal Density is a 16-bit value that defines the average output duty ratio:

Equation 1

$$\begin{cases} DutyRatio = \dfrac{SignalDensity - 1}{Period}, \text{for CompareType} = \text{Less Than} \\ DutyRatio = \dfrac{SignalDensity}{Period}, \text{for CompareType} = \text{Less Than or Equal To} \end{cases}$$

In this equation, Period depends on the Dimming Resolution (polynomial order):

Equation 2

$$Period = 2^{DimmingResolution} - 1$$

Frequency Compensation

Figure 2 shows that output the frequency depends on the Signal Density value (or to the duty ratio). If the duty ratio is near zero or one then the output frequency is the lowest. The middle of the duty ratio values range has the highest value.

In general, the output frequency can be calculated using the following formula:

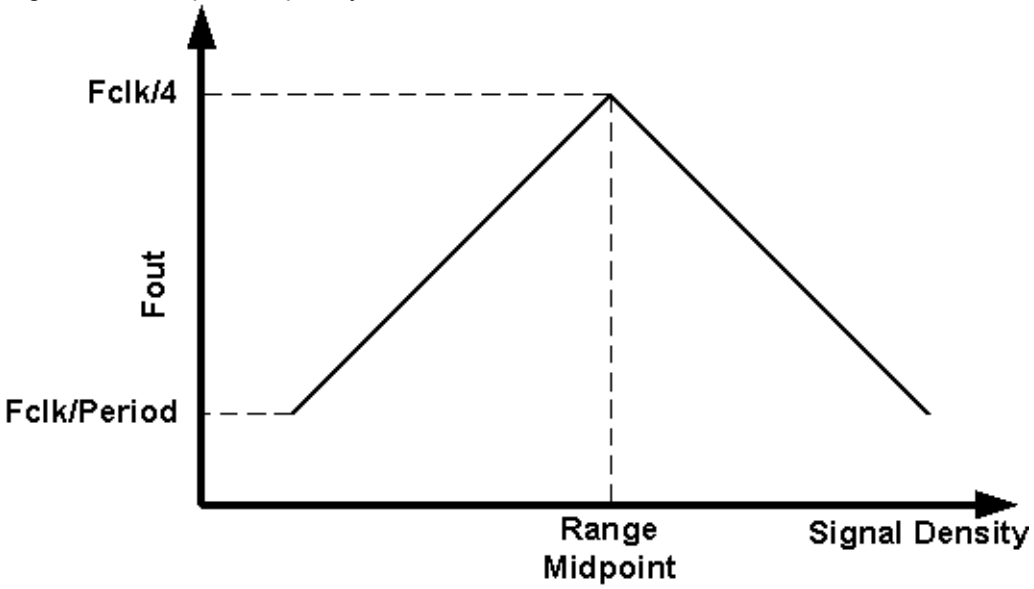
Equation 3

$$\begin{cases} Fout \approx 0.5 \times DutyRatio \times Fclk & for \quad 0 < DutyRatio \le 0.5 \\ Fout \approx 0.5 \times (1 - DutyRatio) \times Fclk & for \quad 0.5 < DutyRatio < 1 \end{cases}$$

In Equation 3, Fclk is the input frequency of the pseudo-random counter.

The lowest frequency value is Fclk/Period and the highest is Fclk/4:

Figure 3. Output Frequency



Taking into account that Fclk depends on the block input frequency of the ModClock (24 or 48 MHz) and the ClockScaler value, the minimum value of Fout is:

Equation 4

$$Fout = \frac{ModClock}{ClockScaler \ \ Period}$$

In an LED application, dimming frequencies that are too low cause visible flicker in the LEDs. 300Hz is the recommended minimum for Fout. Increasing the dimming frequency at middle of the Signal Density values may reduce efficiency due to the higher switching losses. The Software Frequency Compensation mechanism solves this problem by limiting the range of variability on the Fout signal to a single frequency value (or close). This is done by an API that changes the ClockScaler value in such a way to keep Fout at this desired single value.

The value of the ClockScaler that compensates Fout changes can be calculated as:

Equation 5

$$ClockScaler = \frac{1}{2} \cdot ClockScaler_{def} \cdot SignalDensity \qquad for \quad 0 < SignalDensity < \frac{Period}{2}$$

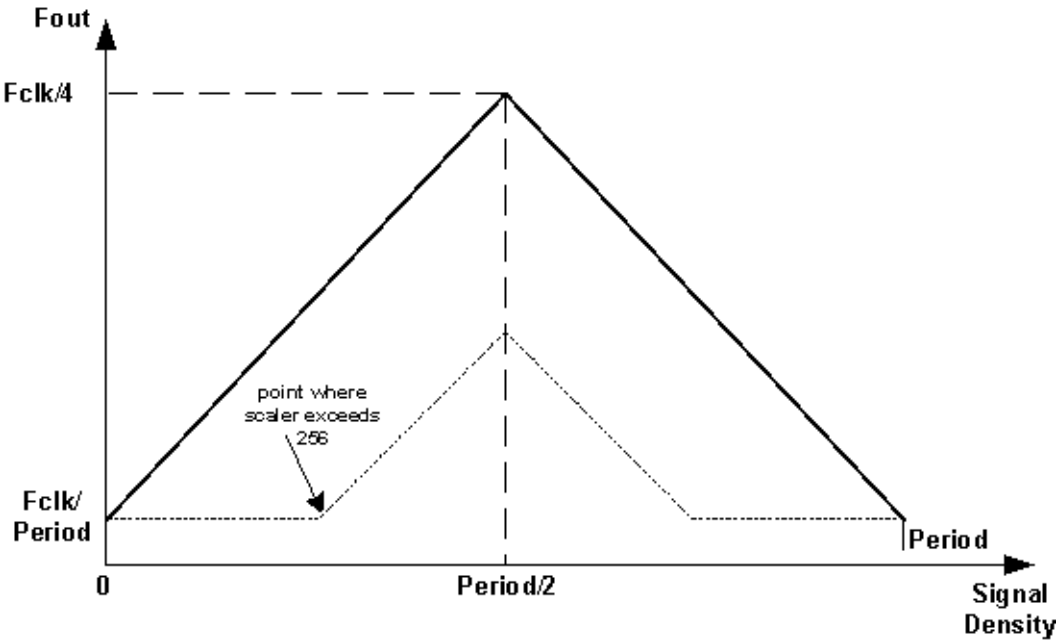
$$ckScaler = \frac{1}{2} \cdot ClockScaler_{def} \cdot (Period - SignalDensity) \qquad for \quad \frac{Period}{2} < SignalDensity < Per$$

The $ClockScaler_{def}$ is an initial user module parameter value.

As long as the ClockScaler is limited to a [1:256] range, not all signal density values can be compensated. This is especially true at resolutions higher than 9. At the point where the calculated scaler exceeds the maximum as shown, the compensation is not complete.

.

Figure 4. Primary and Compensated Output Frequency



DC and AC Electrical Characteristics

Table 2. AC Specifications

AC Parameter	Description	Specification		Units
		Min	Max	
Clock	Input frequency	24	48	MHz
ClockScaler	Input frequency scaler	1	256	
F_{out}	F_{out} is MOD block output frequency for PrISM mode	$\frac{Clock_{min}}{Period \cdot ClockScaler_{max}}$	$\frac{Clock_{max}}{4 \cdot ClockScaler_{min}}$	MHz

Placement

The PrISM16HW User Module can be placed in one of the available modulator blocks; MOD0, MOD1, MOD2, or MOD3.

Parameters and Resources

The following parameters configure the PrISM16HW User Module.

ClockScaler

The ClockScaler parameter allows the input clock (48 MHz or 24 MHz) to be scaled down by a factor of ClockScaler. Allowed values are between 1 and 256.

DimmingResolution

The DimmingResolution parameter contains a value from 2-16. This parameter defines the pseudo-random counter period.

Equation 6

$$Period = 2^{DimmingResolution} - 1$$

SignalDensity

The SignalDensity parameter contains values from 0 to 2^n-1 where n is the value selected in the Dimming Resolution parameter. The output average Duty Ratio is the SignalDensity divided by Period. For a CompareType of less than or equal to, use the following equation:

Equation 7

$$DutyRatio = \frac{SignalDensity}{Period}$$

For a CompareType of Less Than, as the following equation:

Equation 8

$$DutyRatio = \frac{SignalDensity - 1}{Period}$$

FrequencyCompensation

Enables or disables software frequency compensation.

CompareType

This parameter sets the compare function type "LessThan" or "LessThanEqual". This affects output signal duty ratio. A compare type must be chosen for the user module to function properly.

Parameter	Description
LessThan	PrISM16HW output goes high when Counter value is less than Signal Density value.
LessThanEqual	PrISM16HW output goes high when Counter value is less than or equal to Signal Density value.

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.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns the PrISM16HW_1 to the first instance of this user module in a given project. It can be changed to any unique value that follows the syntactic rules for identifiers. The assigned instance name becomes the prefix of every global function name, variable, and constant symbol. In the following descriptions, the instance name has been shortened to PrISM16HW for simplicity.

Note

** In this, as in all user module APIs, you can alter the values of the A and X register 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 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 functions 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.

PRISM16HW_Start

Description:

Starts the PrISM16HW operation. Until started output of PrISM16HW is asserted high.

C Prototype:

```
void PRISM16HW_Start(void);
```

Assembler:

```
lcall PRISM16HW_Start
```

Parameters:

None

Return Value:

None

Side Effects:

After this function is called ClockScaler is returned to value defined in the User Module ClockScaler parameter.

See Note ** at the beginning of the API section.

PRISM16HW_Stop

Description:

Stops the PrISM16HW operation. The output of the PrISM16HW is high when the user module is stopped.

C Prototype:

```
void PRISM16HW_Stop(void);
```

Assembler:

```
lcall PRISM16HW_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

PRISM16HW_SetClockScaler

Description:

Writes the Clock Scaler with a value from 1 to 256. The input clock (48 MHz or 24 MHz) is scaled down by a factor of ClockScaler.

C Prototype:

```
void PRISM16HW_SetClockScaler(WORD wClockScaler);
```

Assembly:

```
mov X, [wClockScaler] ; MSB
mov A, [wClockScaler+1] ; LSB
lcall PRISM16HW_SetClockScaler
```

Parameters:

wClockScaler is a value from 1 to 256.

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PRISM16HW_SetSignalDensity

Description:

Sets the PrISM16HW Signal Density of the output of the modulator block. If the Frequency Compensation mode is enabled, this API also adjusts the Frequency Scaler block to keep a stable output frequency.

C Prototype:

```
void PRISM16HW_SetSignalDensity(WORD wSignalDensity);
```

Assembler:

```
mov X, [wSignalDensity]
mov A, [wSignalDensity+1]
lcall PRISM16HW_SetSignalDensity
```

Parameters:

wSignalDensity - 16-bit signal density value. MSB is passed in the X register and LSB is passed in the Accumulator.

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PRISM16HW_SetDimmingResolution

Description:

Initializes the pseudorandom counter Polynomial Register with a polynomial that matches the desired resolution.

C Prototype:

```
void  PRISM16HW_SetDimmingResolution(BYTE bResolution);
```

Assembler:

```
mov   A, bResolution
lcall  PRISM16HW_SetDimmingResolution
```

Parameters:

bResolution - the resolution parameter contains a value from 2 to 16 (which be presented by 1 byte). This parameter defines the pseudo-random counter period as 2^n-1 where n is resolution.

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section

PRISM16HW_EnableFrequencyComp

Description:

Enables the software frequency compensation.

C Prototype:

```
void PRISM16HW_EnableFrequencyComp(void);
```

Assembler:

```
lcall PRISM16HW_EnableFrequencyComp
```

Parameters:

None.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section

PRISM16HW_DisableFrequencyComp

Description:

Disables the software frequency compensation.

C Prototype:

```
void PRISM16HW_DisableFrequencyComp(void);
```

Assembler:

```
lcall PRISM16HW_DisableFrequencyComp
```

Parameters:

None.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section

Sample Firmware Source Code

The C code illustrated here shows you how to use the PrISM16HW User Module.

```
PRISM16HW_SetDimmingResolution(10);
PRISM16HW_SetSignalDensity(256);
PRISM16HW_Start();
```

The same code in assembly is as follows.

```
mov   A, 0Ah
call  PRISM16HW_SetDimmingResolution
mov   X, 01h
mov   A, 00h
call  PRISM16HW_SetSignalDensity
call  PRISM16HW_Start
```

Configuration Registers

Table 3. PRISM16HW_PCF_REG

Bit	7	6	5	4	3	2	1	0
Value	Programmable Clock Frequency Scaler							

Programmable Clock Frequency Scaler determines the clock scaler for the PrISM block. The value of this register is determined by the choice made for the ClockScaler parameter in the user module parameters of the Device Editor. Also value can be changed by PRISM16HW_SetClockScaler() API.

Table 4. PRISM16HW_PDH_REG (MSB), PRISM16HW_PDL_REG (LSB)

Bit	7	6	5	4	3	2	1	0
LSB	Polynomial Register (LSB)							
MSB	Polynomial Register (MSB)							

Polynomial determines the period value for the PrISM16HW block. The value of this register is determined by the choice made for the DimmingResolution parameter in the user module parameters of the Device Editor. Also value can be changed by PRISM16HW_SetDimmingResolution() API.

Table 5. PRISM16HW_PWH_REG (MSB), PRISM16HW_PWL_REG (LSB)

Bit	7	6	5	4	3	2	1	0
LSB	Signal Density Register (LSB)							
MSB	Signal Density Register (MSB)							

Signal Density determines the value for the PrISM16HW block output. The value of this register is determined by the choice made for the SignalDensity parameter in the user module parameters of the Device Editor. The value can be changed by PRISM16HW_SetSignalDensity() API.

Table 6. PRISM16HW_PCFG_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	CompType	0

CompType determines the PrISM16HW compare type. The value of this bit is determined by the choice made for the CompareType parameter in the user module parameter of the Device Editor.

Table 7. PRISM16HW_GCFG_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	1	Enable

Enable turns on the PrISM16HW block. The value of this bit can be changed by PRISM16HW_Start() and PRISM16HW_Stop() API.

Version History

Version	Originator	Description
1.0	DHA	Initial version

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.

Copyright © 2009-2014 Cypress Semiconductor Corporation. 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 Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark 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.