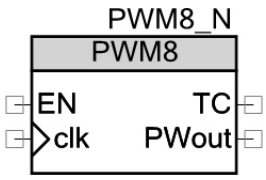
Pulse Width Modulator with Enable

PWM8 v1.0

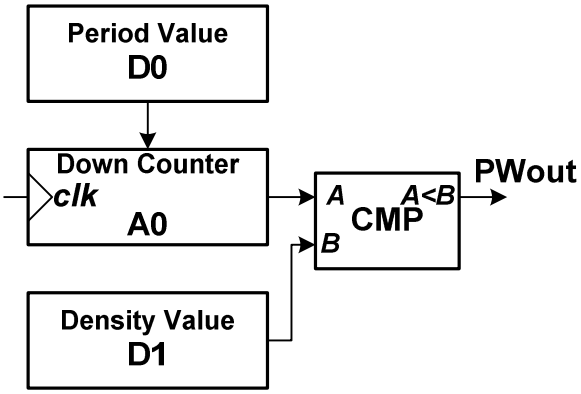


Features

- Synchronous operation
- Positive-edge triggered clock
- Parameterized period and duty cycle values
- Parameterized first period value for phase control
- Simple to deconstruct

General Description

A pulse width modulator produces the lowest possible output frequency for a given density. This implementation is a down counter and an 8 bit wide digital comparator. The output is HIGH whenever the counter value (A0) is less than the density value (D1).



The output will be HIGH for a continuous number of cycles equal to the density value.

This component was built as a teaching tool. This classic component's operation is well understood and this datasheet's function is to help understand how it was built through its deconstruction. It is recommended that you review the DSMod component before attempting this one.

Pin Description

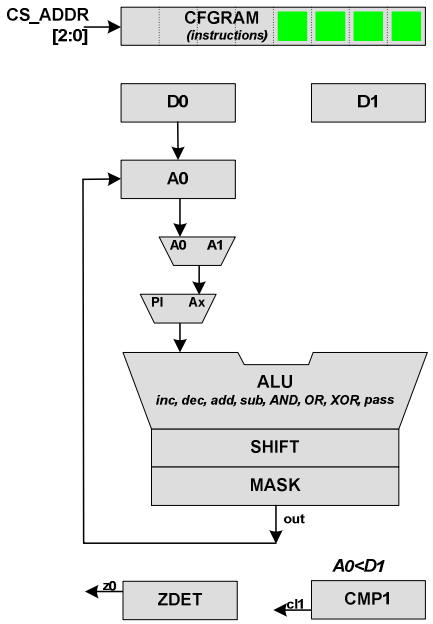
Pin	Type	Function
EN	Input	enable input
clk	input	clock input
TC	output	terminal count output
PWMout	output	pulse width modulated output

Function Table

Operating Modes	Input		Register	Output	
	EN	clk	A0	PWout	TC
Disabled	0	↑	A0	A0 < D1	A0 == 0x00
Decrement (A0 != 0)	1	↑	A0-1	A0<D1	A0 == 0x00
Load (A0 == 0)	1	↑	D0	A0<D1	A0 == 0x00

Deconstructing the Component

This component's purpose is to show how to build a four instruction design and also use the "all zeros" comparator. For more information about datapaths, first review the DSMod component. This component uses the following pieces of a datapath.



The single operation to be performed is that on each clock cycle, A0 is fed to the ALU were it is decremented and fed back to A0. CMP1 is used to compare A0 and D0. The comparator's output is HIGH whenever A0 is less than D0. This flag (cl1) is brought out.

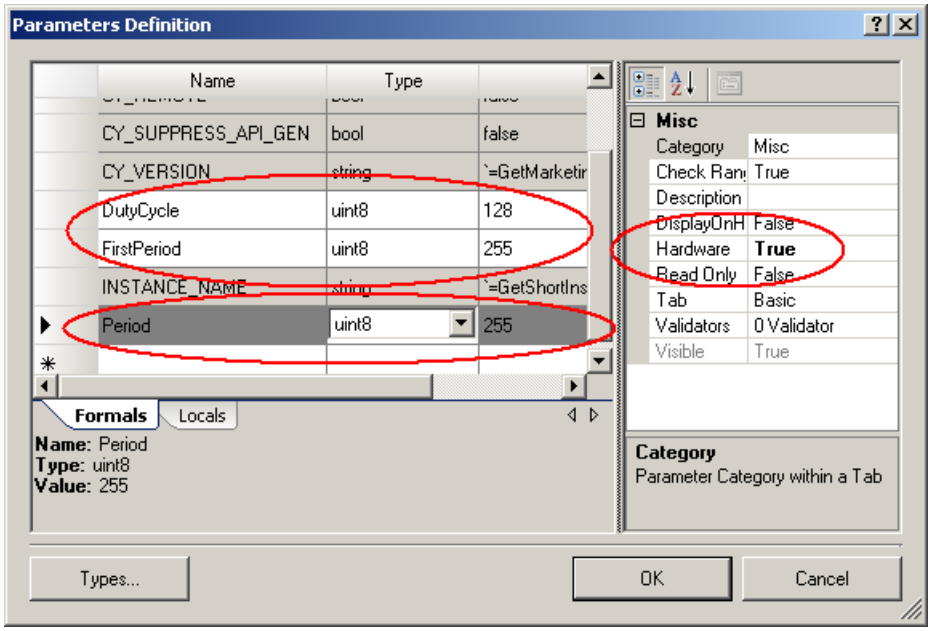
Using PSoC Creator, open the PWM8 example project to see the project schematic (*TopDesign.cysch*). It has a PWM8 component connected to an input switch, output LEDs and a clock.

In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComps library project is, and select the CYCC_PWM8_v1_0 component. Click **OK** and the following files are shown for the component:

- Symbol file (cysym)
- Datasheet (pdf)
- Verilog File (v)

Open the symbol file and you will find a symbol with two inputs and two outputs. It looks like the symbol shown on the first page.

While viewing the symbol file, right-click on the canvas and select **Symbol Parameters**.

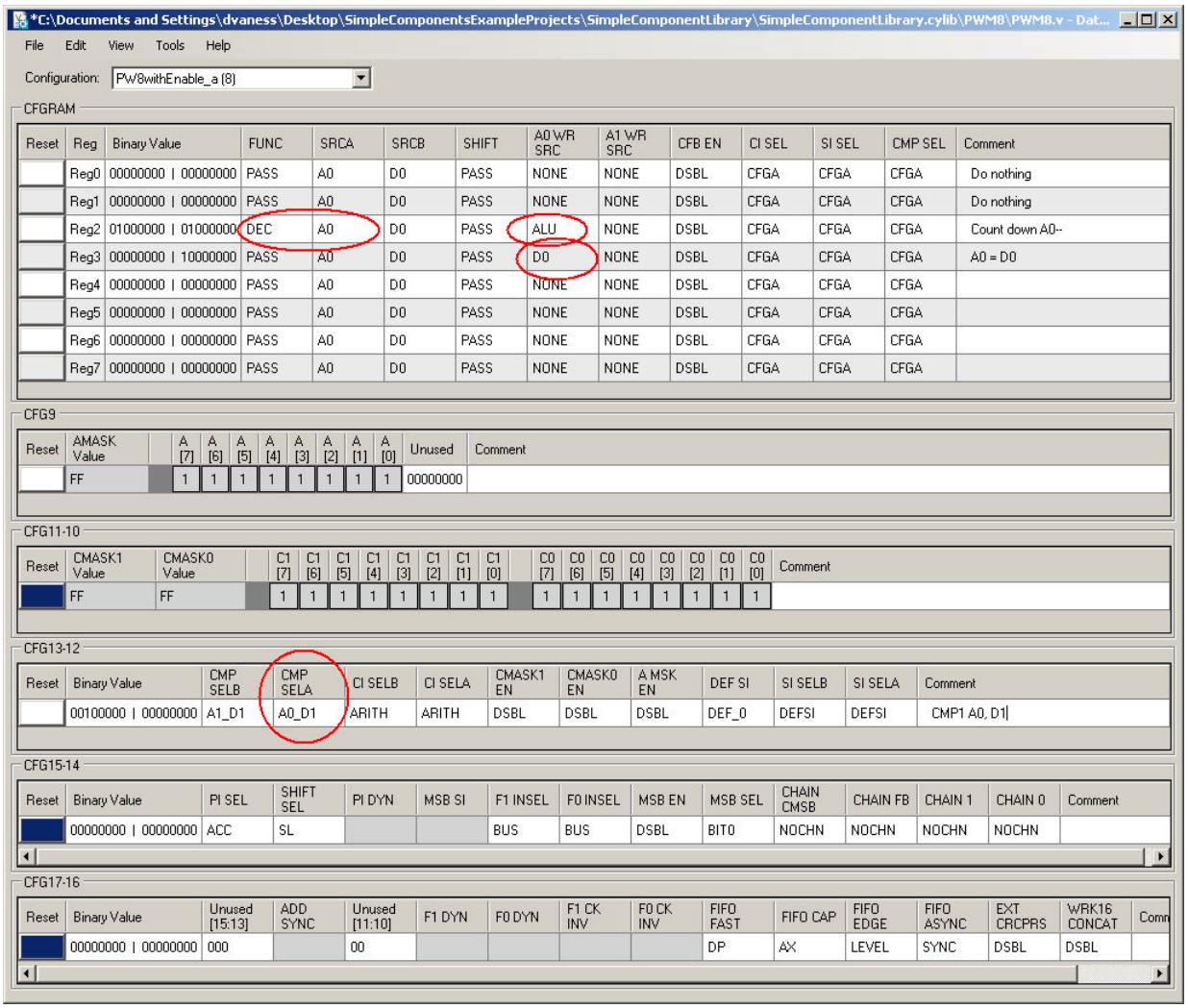


Note that **DutyCycle**, **Period**, and **FirstPeriod** are included as parameters. They are all unsigned 8-bit values and their hardware flags have been enabled.

Open the Verilog file and notice that at lines 26-29, these definitions were passed from the symbol to this Verilog file when it was created. Lines 31-33 are where the symbol parameter

values got passed. The first 20 lines of this header list register usage and the datapath instruction definitions.

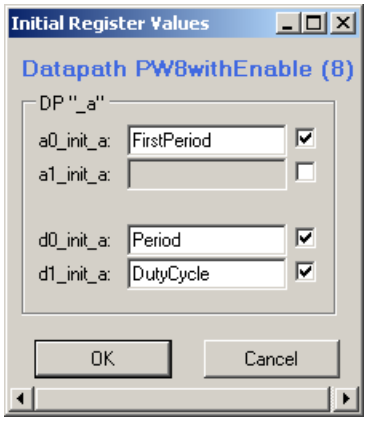
What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the Datapath Configuration Tool for this Verilog file results in the following.



Note that although only three operations are needed to be implemented, four of the datapath instructions are used. By judicious selection of their address position and duplication, the logic to control them has been greatly simplified.

- For the null instructions A0 is passed through the ALU goes nowhere. It does nothing.
- For the count instruction A0 is decremented in the ALU and fed back to A0.
- For the load instruction D0 is loaded directly into A0.

The three registers have been initialized as shown below.



When saved, this tool inserts the updated configuration data back into the Verilog file along with an instance of the datapath interface. Just pass the correct signals into and out of it and you are done.

This counter parameter list may look ominous but only a few parameters have to be entered as shown below.

```
/* input          */  .clk(clk),                 // Clock to datapath
/* input [02:00]  */  .cs_addr({1'b0, EN, TC}), // Control for four instructions
/* output         */  .z0(TC),                   // TC flag(A0 == 0x00)
/* output         */  .cl1(PWout),               // PWM output
```

These parameters do the following:

- Connect a clock signal to the datapath
- Control the four datapath instructions
- Bring out the “all zeros” flag
- Connect CMP1 to output

Support

PSoC Creator Community Components are developed and supported by the Cypress Developer Community. Go to www.cypress.com/CommunityComponents to discuss this and other Community Components.

© Cypress Semiconductor Corporation, 2013. 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® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This work is licensed under a Creative Commons Attribution 3.0 Unported License.

http://creativecommons.org/licenses/by/3.0/deed.en_US

You are free to:

- Share — to copy, distribute and transmit the work
- Remix — to adapt the work
- Make commercial use of the work