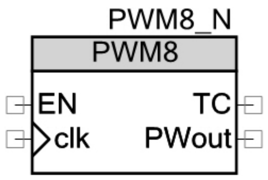
Pulse Width Modulator with Enable

PWM8 v1.0

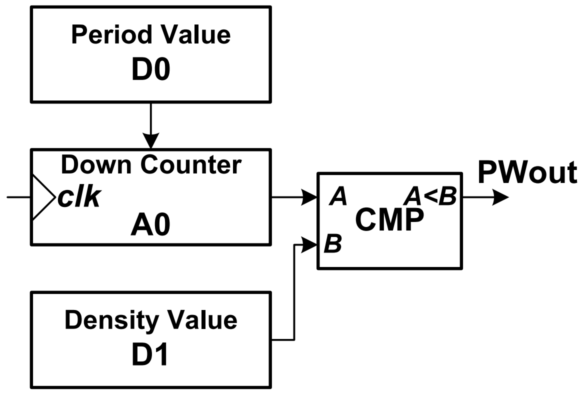


Features

- 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

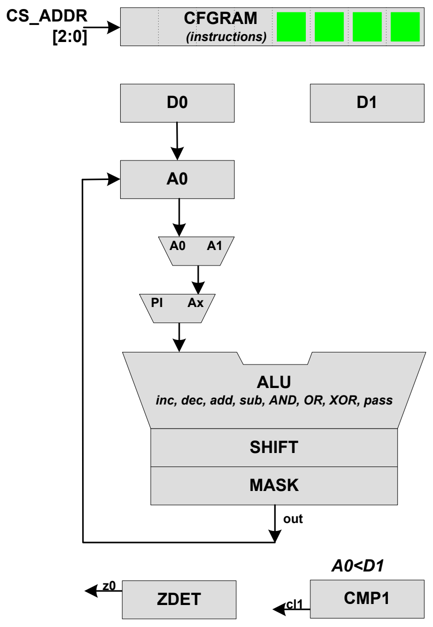
Pin	Type	Function
EN	Input	enable input
clk	input	clock input
TC	output	terminal count output
PWMout	output	pulse width modulated output

Function Table

	Input		Register	Output	
Operating Modes	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.

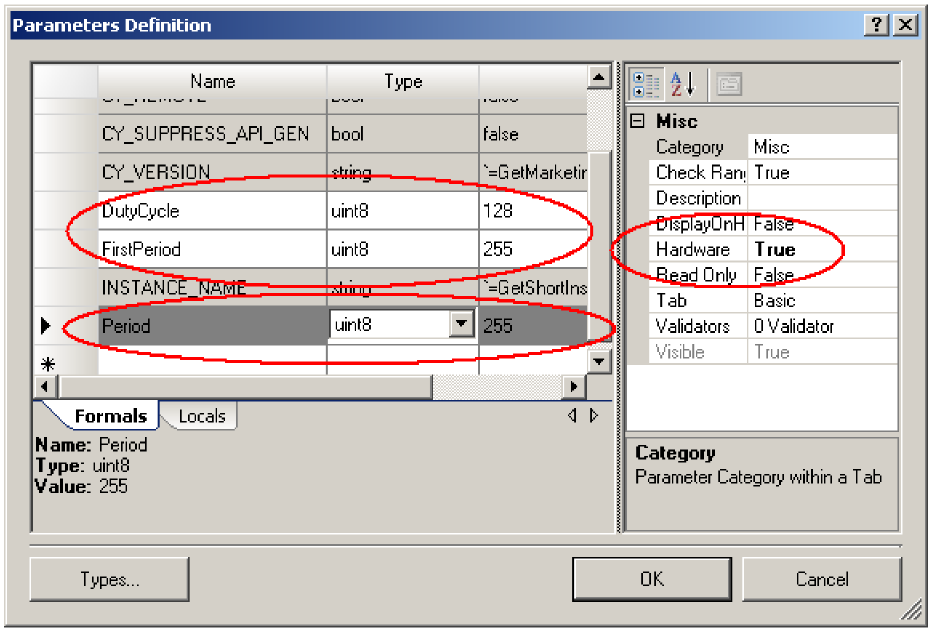
Using PSoC Creator, open the PWM8 example project to see the project schematic (*TopDesign.cysch*). It has a PWM8 component connected to an input switch, output LEDs and a clock.

In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComponentLibrary 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**.


Parameters Definition

Name	Type	
CY_SUPPRESS_API_GEN	bool	false
CY_VERSION	string	`=GetMarketir
DutyCycle	uint8	128
FirstPeriod	uint8	255
INSTANCE_NAME	string	`=GetShortIns
Period	uint8	255

Formals | Locals

Name: Period
Type: uint8
Value: 255

Misc

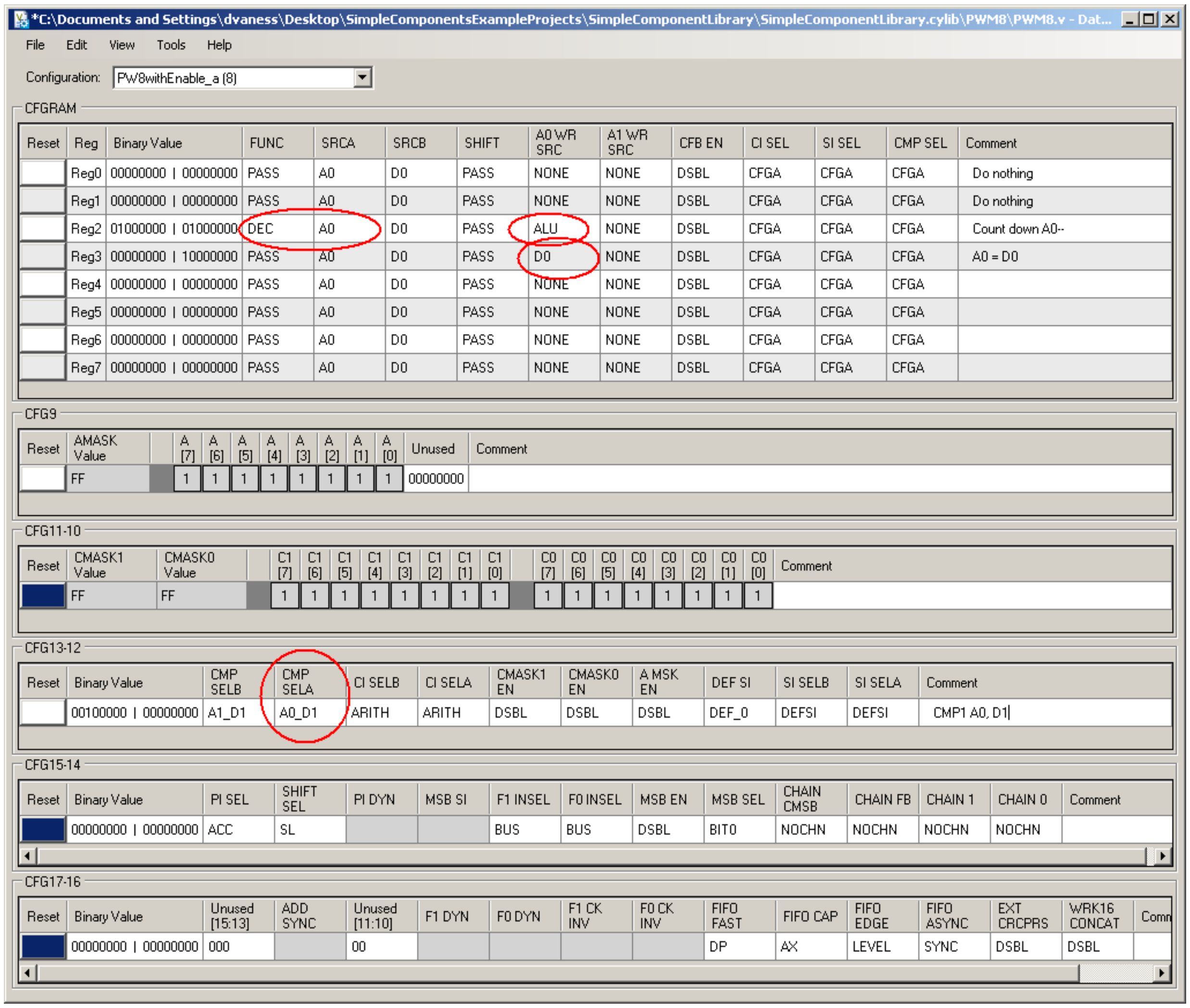
Category	Misc
Check Ran	True
Description	
DisplayOnH	False
Hardware	**True**
Read Only	False
Tab	Basic
Validators	0 Validator
Visible	True

Category
Parameter Category within a Tab

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.



*C:\Documents and Settings\dvaness\Desktop\SimpleComponentsExampleProjects\SimpleComponentLibrary\SimpleComponentLibrary.cylib\PWM8\PWM8.v - Dat...

Configuration: PW8withEnable_a (8)

CFGRAM

Reset	Reg	Binary Value	FUNC	SRCA	SRCB	SHIFT	A0 WR SRC	A1 WR SRC	CFB EN	CI SEL	SI SEL	CMP SEL	Comment
	Reg0	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	Do nothing
	Reg1	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	Do nothing
	Reg2	01000000 | 01000000	DEC	A0	D0	PASS	ALU	NONE	DSBL	CFGA	CFGA	CFGA	Count down A0--
	Reg3	00000000 | 10000000	PASS	A0	D0	PASS	D0	NONE	DSBL	CFGA	CFGA	CFGA	A0 = D0
	Reg4	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg5	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg6	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg7	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	

CFG9

Reset	AMASK Value		A [7]	A [6]	A [5]	A [4]	A [3]	A [2]	A [1]	A [0]	Unused	Comment
	FF		1	1	1	1	1	1	1	1	00000000	

CFG11-10

Reset	CMASK1 Value	CMASK0 Value		C1 [7]	C1 [6]	C1 [5]	C1 [4]	C1 [3]	C1 [2]	C1 [1]	C1 [0]		C0 [7]	C0 [6]	C0 [5]	C0 [4]	C0 [3]	C0 [2]	C0 [1]	C0 [0]	Comment
	FF	FF		1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	

CFG13-12

Reset	Binary Value	CMP SELB	CMP SELA	CI SELB	CI SELA	CMASK1 EN	CMASK0 EN	A MSK EN	DEF SI	SI SELB	SI SELA	Comment
	00100000 | 00000000	A1_D1	A0_D1	ARITH	ARITH	DSBL	DSBL	DSBL	DEF_0	DEFSI	DEFSI	CMP1 A0, D1

CFG15-14

Reset	Binary Value	PI SEL	SHIFT SEL	PI DYN	MSB SI	F1 INSEL	F0 INSEL	MSB EN	MSB SEL	CHAIN CMSB	CHAIN FB	CHAIN 1	CHAIN 0	Comment
	00000000 | 00000000	ACC	SL			BUS	BUS	DSBL	BIT0	NOCHN	NOCHN	NOCHN	NOCHN	

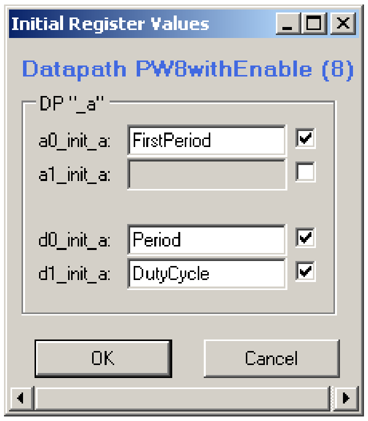
CFG17-16

Reset	Binary Value	Unused [15:13]	ADD SYNC	Unused [11:10]	F1 DYN	F0 DYN	F1 CK INV	F0 CK INV	FIFO FAST	FIFO CAP	FIFO EDGE	FIFO ASYNC	EXT CRCPRS	WRK16 CONCAT	Comm
	00000000 | 00000000	000		00					DP	AX	LEVEL	SYNC	DSBL	DSBL	

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.



a0_init_a: FirstPeriod [✓]
a1_init_a: [ ]
d0_init_a: Period [✓]
d1_init_a: DutyCycle [✓]

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.