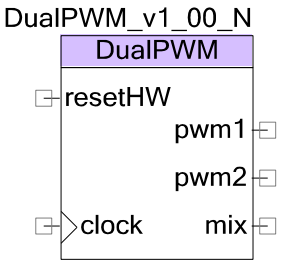
Dual Channel PWM with Independent Outputs

DualPWM v1.00

Features



- Two 16-bit PWM outputs
- Independently configurable period, compare, counter value
- Mix output (XOR of the two PWMs) provided
- Consumes 2 datapaths

General Description

The Component implements two independent 16-bit PWMs within 2 UDBs. Each PWM has its own period, compare and counter registers. See the Functional Description section for a Timing Diagram.

Input/Output Connections

Pin	Type	Function
resetHW	Input	reset
clock	Input	twice the operating frequency of each of the PWMs
pwm1	Output	output of the first pwm
pwm2	Output	output of the second pwm
mix	Output	pwm1 XOR pwm2

Component Parameters

Parameter Name	Default Value	Description
Compare1_init	0	16-bit initial compare value for PWM1
Compare2_init	0	16-bit initial compare value for PWM2
Counter1_init	0	16-bit initial counter value for PWM1
Counter2_init	0	16-bit initial counter value for PWM2

Debug	False	Enables the state machine pins to assist in debugging
EnableReset	False	Enables an optional reset terminal for resetting both PWM1 and PWM2
FIFO_0_AlternateLevelReport	False	*Must call the Stat() API for this to work properly* When set to 'false' The FIFO BUS and BLK status will mean the following: **FIFO in output mode** BUS: FIFO is not empty BLK: FIFO is full **FIFO in input mode** BUS: FIFO is not full BLK: FIFO is empty When set to 'true' The FIFO BUS and BLK status will mean the following: **FIFO in output mode** BUS: FIFO is at least half full BLK: FIFO is full **FIFO in input mode** BUS: FIFO is at least half empty BLK: FIFO is empty
FIFO_0_SingleBufferMode	True	* Must be TRUE for the DualPWM to work correctly* *Must call the Start() API for this to work properly* When set to "false" the FIFO acts as a four byte deep FIFO. When set to "true" the FIFO acts as a single register. This mode is useful when you need to use a FIFO as an extra datapath register. In this mode with the FIFO set to output, the ALU can be written into the FIFO with the Fx_Load signal and on the next clock edge, the data can be read out of the FIFO into the A register or into the Dx - register with a Dx_Load signal. This is the only FIFO mode where this is possible.
FIFO_1_AlternateLevelReport	False	See description for FIFO_0_AlternateLevelReport
FIFO_1_SingleBufferMode	True	See description for FIFO_0_SingleBufferMode

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 “DualPWM_1” to the first instance of this component in a given design. You can the 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 “DualPWM.”

Function	Description
DualPWM_Start()	Enable Component and set up initial configuration
DualPWM_Enable()	Enable the clock to the Component
DualPWM_Disable()	Disable the clock to the Component
DualPWM_ReadPeriod1()	Read period value for PWM1
DualPWM_ReadPeriod2()	Read period value for PWM2
DualPWM_WritePeriod1()	Write period value for PWM1
DualPWM_WritePeriod1()	Write period value for PWM2
DualPWM_ReadCompare1()	Read compare value for PWM1
DualPWM_ReadCompare2()	Read compare value for PWM2
DualPWM_WriteCompare1()	Write compare value for PWM1
DualPWM_WriteCompare2()	Write compare value for PWM2
DualPWM_ReadCounter1()	Read current counter value for PWM1
DualPWM_ReadCounter2()	Read current counter value for PWM2
DualPWM_WriteCounter1()	Write current counter value for PWM1
DualPWM_WriteCounter2()	Write current counter value for PWM2

void DualPWM_Start(void)

Description: Configures/Initializes the PWM, and enables the clock to it

Parameters: None

Return Value: None

Side Effects: None

void DualPWM_Enable(void)

Description: Enables the clock to the DualPWM

Parameters: None

Return Value: None

Side Effects: None

void DualPWM_Disable(void)

Description: Disables the clock to the DualPWM

Parameters: None

Return Value: None

Side Effects: None

All the rest of the ‘functions’ are implemented as macros.

uint16 DualPWM_ReadPeriodx(void)

Description: Reads the period value of counter x (x = 1 or 2)

Parameters: None

Return Value: uint16 period value

Side Effects: None

void DualPWM_WritePeriodx(uint16 period)

Description: Writes the period value of counter x (x = 1 or 2)

Parameters: uint16 period value

Return Value: None

Side Effects: None

uint16 DualPWM_ReadComparex(void)

Description: Reads the compare value of counter x (x = 1 or 2)

Parameters: None

Return Value: uint16 compare value

Side Effects: None

void DualPWM_WriteComparex(uint16 compare)

Description: Writes the compare value of counter x (x = 1 or 2)

Parameters: uint16 compare value to be written

Return Value: None

Side Effects: None

uint16 DualPWM_ReadCounterx(void)

Description: Reads the counter value of counter x (x = 1 or 2)

Parameters: None

Return Value: uint16 counter value

Side Effects: None

void DualPWM_WriteCounterx(uint16 value)

Description: Writes the counter value of counter x (x = 1 or 2)

Parameters: uint16 counter value to be written

Return Value: None

Side Effects: None

Functional Description

The DualPWM implements two 16-bit PWMs with independent period counters, periods and compares. Figure 1 and Figure 2 contrast the behavior of a regular UDB-based PWM Component and the DualPWM.

Figure 1 shows the behavior of the UDB-based PWM Component with a period of 300. PWM1 has a compare value of 150, while PWM2 has a compare value of 75.

Figure 1. Cypress Component Catalog UDB-based PWM Behavior

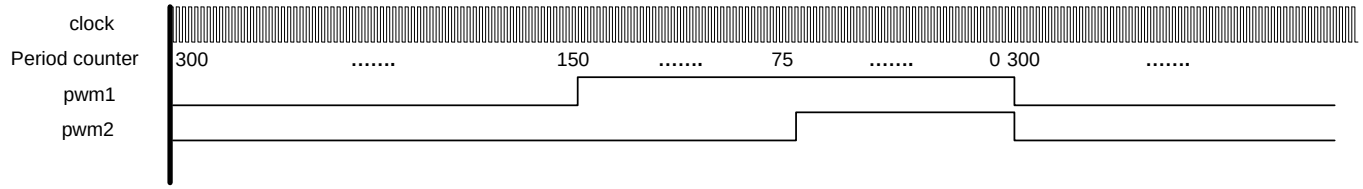
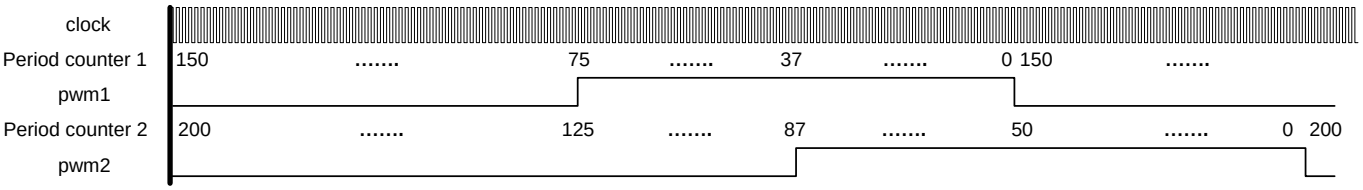


Figure 2 shows the behavior of the DualPWM Component with PWM1 configured with a period of 150 and compare of 75, and PWM2 configured with a period of 200 and a compare of 87.

Figure 2. Custom DualPWM Component Behavior



Notable Facts

- Both the regular and custom PWM are implemented as down-counters with a less-than compare function (i.e. only when the period counter is less than the compare value is the PWM output asserted)
- Each of the DualPWM period counters update at half the frequency of the ‘clock’ input.
- If pwm1 and pwm2 are at the same frequency and in phase, the mix output will be at 0.

Registers

Control

The DualPWM uses one instance of a control register to provide firmware enable and reset functions. Table 1 shows how to use the control register to enable and reset the Component (named DualPWM for the purpose of the example).

Table 1. Usage of DualPWM Control Register

```
DualPWM_ControlReg |= DualPWM_ENABLE;           /* enable the dualPWM */
DualPWM_ControlReg &= DualPWM_ENABLE_CLR;       /* disable the dualPWM */
DualPWM_ControlReg |= DualPWM_RESET;            /* hold dualPWM in reset */
DualPWM_ControlReg &= DualPWM_RESET_CLR;        /* release dualPWM from reset */
```

DualPWM_ENABLE

Bit 0 of the control register. Enables (‘1’) or disables (‘0’)the clock to the DualPWM using a UDB clock enable instantiated in hardware.

DualPWM_RESET

Bit 1 of the control register. Holds the Component in reset (‘1’) or releases it from reset (‘0’)

Resources

Resource Type			
Datapath Cells	**Macrocells**	**Status Cells**	**Control Cells**
2	5	0	1

API Memory Usage

Minimal. TBD

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