Pulse Width Discriminator Datasheet PWD V 1.0

Copyright © 2008-2014 Cypress Semiconductor Corporation. All Rights Reserved.

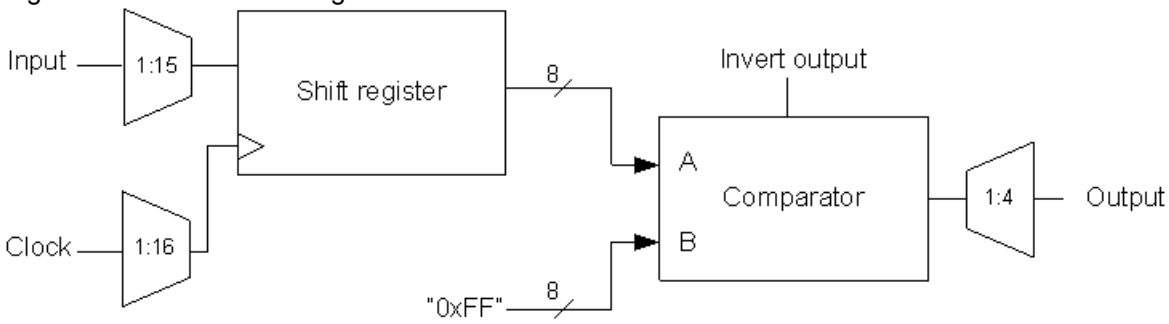
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O and Clock)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/21xxx, CY8CLED02/04/08/16, CY8CLED0xD, CY8CLED0xG, CYWUSB69xx, CY8C21x45, CY8C22x45, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx						
	1	0	0	21	0	1-2
	1	0	0	21	0	1-2
	1	0	0	21	0	1-2
	1	0	0	21	0	1-2

Features and Overview

- Selectable input
- Selectable output (with row interconnect)
- Selectable clock, also it can run up to 48 MHz
- Input signal is active low
- Output signal is active low
- Data input can be inverted
- Data output can be inverted

The Pulse Width Discriminator (PWD) User Module produces an output pulse and interrupt in response to an input pulse of certain duration. Input pulses less than specified are ignored. The main purpose of PWD is detecting signal preamble in the different communication protocols. Also it can be used as hardware signal debouncer.

Figure 1. PWD Block Diagram



Functional Description

PWD User Module provides the compare output of a one CRCPRS hardware block to a row interconnect. PWD User Module has ability to generate interrupts, which can be enabled or disabled using API. PWD occupies one digital block. PWD firmware is used to start and stop device, also for enable and disable interrupt. The pulse length should be at least 8 PWD clock pulses length to be detected. Note that interrupt activates at the positive edge of output signal not depending of InvertOutput parameter state.

Equation 1

$$PulseWidth \geq 8 \times \frac{1}{PWDClock} ; \ldots \Leftrightarrow \ldots PWDClock \leq 8 \times \frac{1}{PulseWidth}$$

Timing

Figure 2. PWD Timing Diagram, both InvertInput and InvertOutput parameters are set to Normal

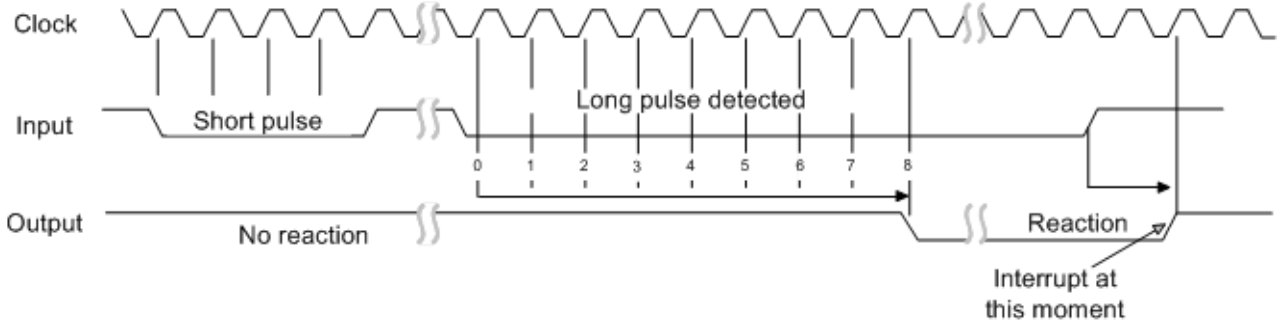


Figure 3. PWD Timing Diagram, both InvertInput and InvertOutput parameters are set to Invert

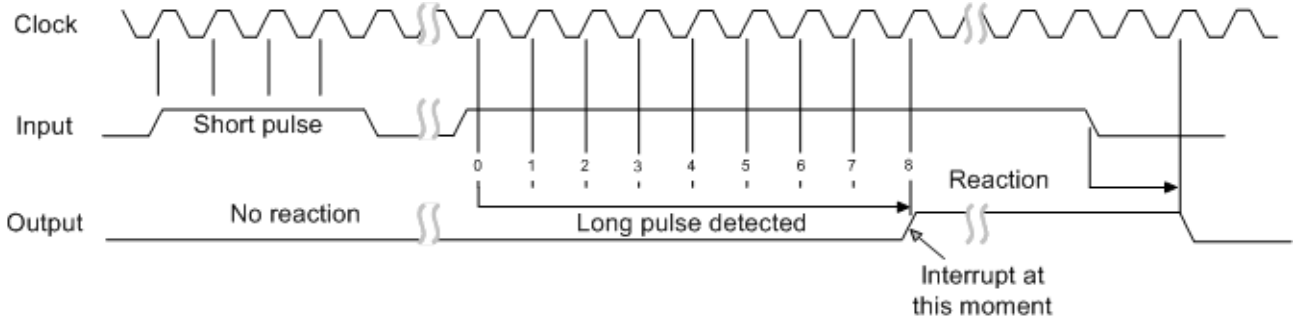
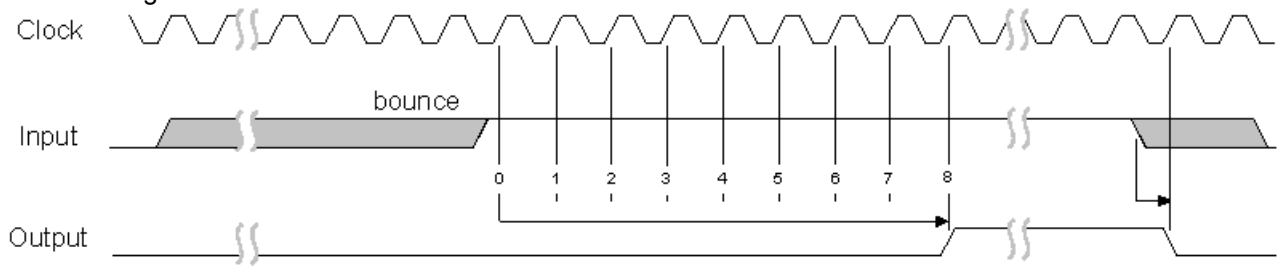


Figure 4. Using a PWD for Debounce



Placement

The PWD maps onto a single PSoC block. The block name is PWD in the PSoC Designer Device Editor. It may be placed in any of the digital blocks.

Parameters and Resources

Clock

The Clock parameter is selected from one of the available sources. These sources include the 48 MHz oscillator (5.0V operation only), system clock, 32 kHz, VC1, VC2 and VC3, other PSoC blocks, and external inputs routed through global inputs and outputs. When using an external digital clock for the block, the row input synchronization should be turned off for best accuracy, and sleep operation.

Input

The Input parameter is selected from one of the available sources.

Output

The output may be disabled (without interfering with interrupt operations) or connected to any of the row output buses.

InvertOutput

This parameter determines the sense of the output signal. When **Normal** is selected, the output is active-high. Selecting **Invert** causes the sense to be interpreted as active-low.

ClockSync

In PSoC devices digital blocks may provide clock sources in addition to the system clocks. Digital clock sources may even be chained in ripple fashion. This introduces skew with respect to the system clocks. These skews are more critical in the CY8C29/27/24/21xxx and CY8CLED02/04/08/16 PSoC device families because of various data-path optimizations, particularly those applied to the system buses. This parameter may be used to control clock skew and ensure proper operation when reading and writing PSoC block register values. Appropriate values for this parameter should be determined from the following table.

ClockSync Value	Use
Sync to SysClk	Use this setting for any 24 MHz (SysClk) derived clock source that is divided by two or more. Examples include VC1, VC2, VC3 (when VC3 is driven by SysClk), 32KHz, and digital PSoC blocks with SysClk-based sources. Externally generated clock sources should also use this value to ensure that proper synchronization occurs.
Sync to SysClk*2	Use this setting for any 48 MHz (SysClk*2) based clock unless the resulting frequency is 48 MHz (in other words, when the product of all divisors is 1).
Use SysClk Direct	Use when a 24 MHz (SysClk/1) clock is desired. This does not actually perform synchronization but provides low-skew access to the system clock itself. If selected, this option overrides the setting of the Clock parameter. It should always be used instead of VC1, VC2, VC3 or digital blocks where the net result of all dividers in combination produces a 24 MHz output.
Unsynchronized	Use when the 48 MHz (SysClk*2) input is selected. Use when unsynchronized inputs are desired. In general this use is advisable only when interrupt generation is the sole application of the Counter.

InvertInput

This parameter determines the sense of the input signal. When **Normal** is selected, the input is active high. Selecting **Invert** causes the sense to be interpreted as active low.

Interrupt Generation Control

There are two additional parameters that become available when the **Enable interrupt generation control** check box in PSoC Designer is checked. This is available under **Project > Settings > Chip Editor**. Interrupt Generation Control is important when multiple overlays are used with interrupts shared by multiple user modules across overlays:

- Interrupt API
- IntDispatchMode

InterruptAPI

The InterruptAPI parameter allows conditional generation of a user module's interrupt service routine (ISR) and interrupt vector table entry. Select "Enable" to generate the ISR and interrupt vector table entry. Select "Disable" to bypass the generation of the ISR and interrupt vector table entry. Properly selecting this parameter is particularly important for projects that have multiple overlays where a single block resource is shared by the different overlays. By enabling Interrupt API generation only when it is necessary, the need to generate interrupt dispatch code is usually eliminated. This reduces code overhead.

IntDispatchMode

The IntDispatchMode parameter is used to specify how an interrupt request is handled for interrupts shared by multiple user modules existing in the same block but in different overlays. Selecting "ActiveStatus" causes firmware to test which overlay is active before servicing the shared interrupt request. This test occurs every time the shared interrupt is requested. This adds latency and also produces a nondeterministic procedure of servicing shared interrupt requests, but does not require any RAM. Selecting "OffsetPreCalc" causes firmware to calculate the source of a shared interrupt request only when an overlay is initially loaded. This calculation decreases interrupt latency and produces a deterministic procedure for servicing shared interrupt requests, but at the expense of a small amount of RAM.

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 PWD_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 PWD for simplicity.

Note

In this, as in all user module APIs, the values of the A and X register may be altered 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 reasons 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 function 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 perserve 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.

The API routines allow you to control the PWD User Module with software.

PWD_Start

Description:

Enables the PWD module.

C Prototype:

```
void  PWD_Start(void)
```

Assembly:

```
lcall  PWD_Start
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model. When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

PWD_Stop

Description:

Stops the PWD module.

C Prototype:

```
void  PWD_Stop(void)
```

Assembly:

```
lcall  PWD_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model. When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

PWD_EnableInt

Description:

Enables the interrupt mode operation.

C Prototype:

```
void  PWD_EnableInt(void)
```

Assembly:

```
lcall  PWD_EnableInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model. When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

PWD_DisableInt

Description:

Disables the interrupt mode operation.

C Prototype:

```
void PWD_DisableInt(void)
```

Assembler:

```
lcall PWD_DisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model. When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the IDX_PP page pointer register is modified.

PWD_ClearInt

Description:

Clears the pending interrupt from this user module.

C Prototype:

```
void PWD_ClearInt(void)
```

Assembler:

```
lcall PWD_ClearInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model. When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

Sample Firmware Source Code

The code sample in assembly language is as follows.

```
include "m8c.inc"       ; part specific constants and macros
include "PSoCAPI.inc"   ; PSoC API definitions for all user modules

export _main

_main:
   M8C_EnableGInt
   call PWD_Start
   call PWD_EnableInt
```

A same code written in C is as follows.

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all user modules

void main(void) {
  M8C_EnableGInt;
  PWD_Start();
  PWD_EnableInt();
}
```

Configuration Registers

The PWD uses a single digital PSoC block named PWD. Block is personalized and parameterized through 7 registers. The following tables give the “personality" values as constants and the parameters as named bit-fields with brief descriptions. Symbolic names for these registers are defined in the user module instance’s C and assembly language interface files (the *.h* and *.inc* files).

Table 1. Function Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	InvertInput	BCEN	1	InvertOutput		0	1	0

The InvertInput, set through a user module parameter displayed in the Device Editor, controls the sense of the input signal. BCEN gates the compare output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line. InvertOutput indicates whether the Invert output is set to “Normal"or “Inverted." The InvertOutput parameter is set in the Device Editor.

Table 2. Input Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	Input				Clock			

Input selects the PWD input source. Clock Source selects the clock to drive the receiver timing. Both parameters are set in the Device Editor.

Table 3. Output Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	ClockSync		Output			0	0	0

The user module "ClockSync" parameter in the Device Editor determines the value of the ClockSync bits. Output selects output from one of four global busses. This parameter is set in the Device Editor.

Table 4. Shift Register (DR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	Shift Register							

Shift Register is the PWD shift register.

Table 5. Polynomial Register (DR1), Bank 0

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

Table 6. Compare Register (DR2), Bank 0

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

Table 7. Control Register (CR0), Bank 0

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

Start indicates that the PWD is enabled when set. It is modified by using the PWD API Start/Stop routines.

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 © 2008-2014 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.