OneShot Datasheet OneShot V 1.0

Copyright © 2005-2013 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/21/20xxx, CY8CLED02/04/08/16, CYWUSB69xx, CY8CLED0xD, CY8CLED0xG, CY8CPLC20, CY8CLED16P01						
8/16/24/32-bit	1	0	0	21	0	2

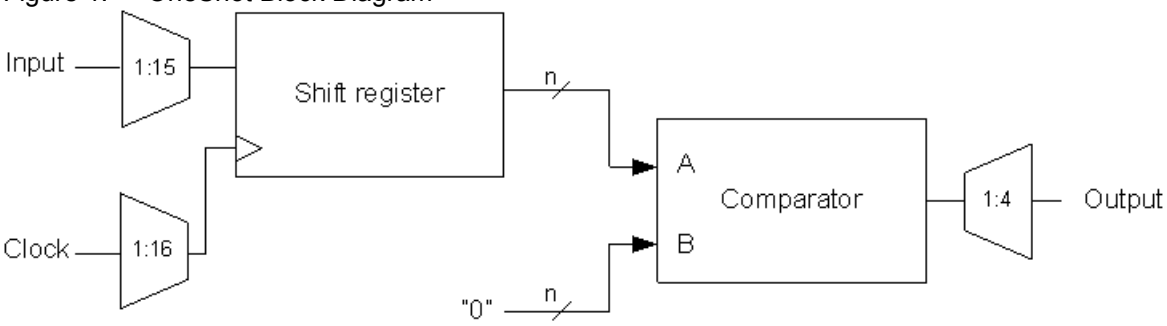
For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- Selectable input
- Selectable output (with row interconnect)
- Selectable clock up to 48 MHz
- Data input can be inverted
- Data output is active low
- Available 8, 16, 24, 32-bit Relax Time

The OneShot User Module is a device that produces a single pulse in response to an input signal. It can be used to reshape short input pulses and generate single pulses with a required duration.

Figure 1. OneShot Block Diagram



Functional Description

The OneShot User Module compares the output of a CRCPRS hardware block to a row interconnect. The Relax Time is selectable and can be 8, 16, 24, or 32 clock pulses, using one, two, three, or four PSoC digital blocks, respectively. The OneShot User Module can generate interrupts, which can be enabled or disabled using an API.

RelaxTime for 8-bit OneShot

Equation 1

$$RelaxTime = 8 \times \frac{1}{OneShotClock} \Leftrightarrow OneShotClock = 8 \times \frac{1}{RelaxTime}$$

RelaxTime for 16-bit OneShot

Equation 2

$$RelaxTime = 16 \times \frac{1}{OneShotClock} \Leftrightarrow OneShotClock = 16 \times \frac{1}{RelaxTime}$$

RelaxTime for 24-bit OneShot

Equation 3

$$RelaxTime = 24 \times \frac{1}{OneShotClock} \Leftrightarrow OneShotClock = 24 \times \frac{1}{RelaxTime}$$

RelaxTime for 32-bit OneShot

Equation 4

$$RelaxTime = 32 \times \frac{1}{OneShotClock} \Leftrightarrow OneShotClock = 32 \times \frac{1}{RelaxTime}$$

DC and AC Electrical Characteristics

Timing

Figure 2. OneShot Timing Diagram

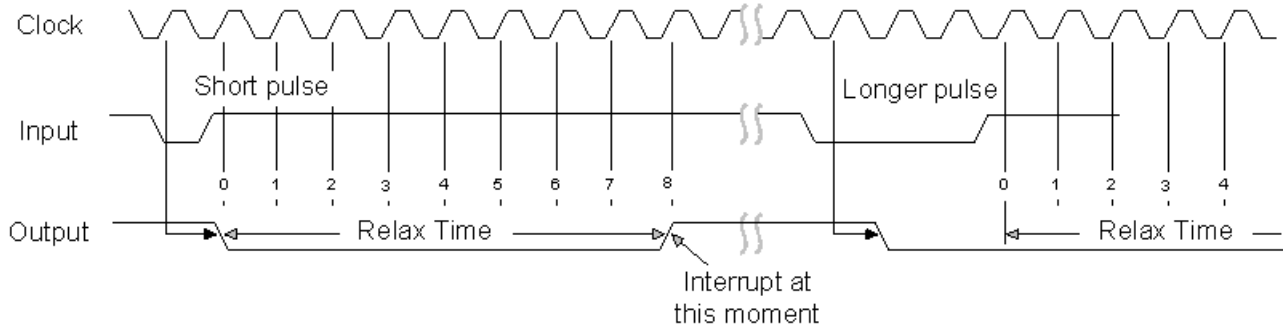
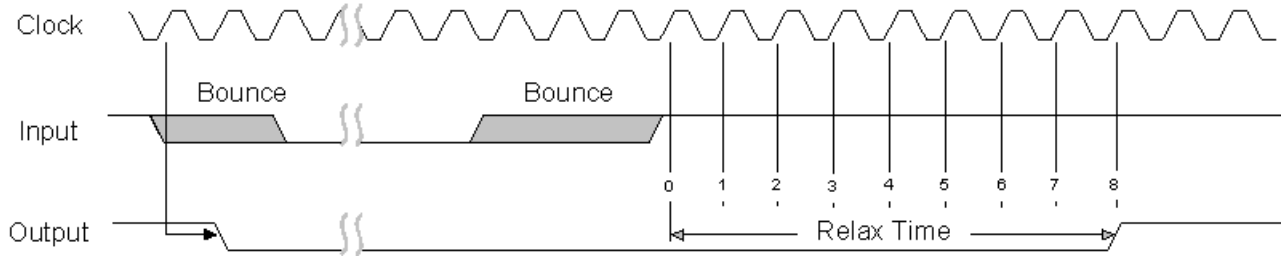


Figure 3. Using OneShot for Debouncing



Placement

The OneShot User Module consumes one digital PSoC block per 8 clock pulses of output signal length. When more than one block is allocated, all blocks are placed consecutively by the Device Editor in order of increasing block number from the least-significant byte (LSB) to the most significant (MSB). Each block is given a symbolic name displayed by the device editor during and after placement. The API qualifies all register names with user assigned instance name and block name to provide direct access to the OneShot registers through the API include files. The block names used by the various widths are given in the following table.

PSoC Blocks	8-Bit	16-Bit	24-Bit	32-Bit
1	OneShot8	OneShot16_LSB	OneShot24_LSB	OneShot32_LSB
2	--	OneShot16_MSB	OneShot24_ISB	OneShot32_ISB1
3	--	--	OneShot24_MSB	OneShot32_ISB2
4	--	--	--	OneShot32_MSB

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), VC1, VC2 and VC3, other PSoC blocks, and external inputs routed through global inputs and outputs. When using an external digital clock for a 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.

ClockSync

In the 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 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), 32 KHz, 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 this setting 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, above. 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 this setting when the 48 MHz (SysClk*2) input is selected. Also use this setting when unsynchronized inputs are desired. In general, you should do this only when interrupt generation is the sole application of the OneShot User Module.

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 ONESHOT_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 ONESHOT 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 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.

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

OneShot_Start

Description:

Enables the OneShot module.

C Prototype:

```
void  OneShot_Start(void)
```

Assembly:

```
lcall  OneShot_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.

OneShot_Stop

Description:

Stops the OneShot module.

C Prototype:

```
void  OneShot_Stop(void)
```

Assembly:

```
lcall  OneShot_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.

OneShot_EnableInt

Description:

Enables the interrupt operation mode.

C Prototype:

```
void  OneShot_EnableInt(void)
```

Assembly:

```
lcall  OneShot_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.

OneShot_DisableInt

Description:

Disables the interrupt mode operation.

C Prototype:

```
void OneShot_DisableInt(void)
```

Assembler:

```
lcall OneShot_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.

OneShot_ClearInt

Description:

Clears the pending interrupt from this user module.

C Prototype:

```
void OneShot_ClearInt(void)
```

Assembler:

```
lcall OneShot_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 following is a C example for communicating with a One Shot device.

```
//
//  This sample shows how to generate single pulses with a 5uS delayed in response to
//  an input signal.
//
// OVERVIEW:
//
//  The OneShot output can be routed to any pin.
//  The OneShot input can be routed to any pin.
//  In this example the OneShot output is routed to P0[0] and the OneShot input is
//  routed to P0[1].
//
//The following changes need to be made to the default settings in the Device Editor:
//
//   1. Select OneShot user module.
```

```
//   2. Select the 8 bit OneShot configuration in MUM
//   3. The User Module will occupy the space in dedicated system resources.
//   4. Rename User Module's instance name to OneShot.
//   5. Set OneShot's Clock Parameter to VC1.
//   6. Set OneShot's Input Parameter to Row_0_Input_1.
//   7. Set OneShot's Output Parameter to Row_0_Output_0.
//   8. Set OneShot's ClockSync Parameter to SyncSysClk.
//   9. Click on Row_0_Output_0 and connect Row_0_Output_0 to GlobalOutEven_0.
//   10.Click on Port_0_1 and set the Select parameter to GlobalInEven_1.
//   11.Select GlobalOutEven_0 for P0[0] in the Pinout.
//
// CONFIGURATION DETAILS:
//
//  1. The UM's instance name must be shortened to OneShot.
//
// PROJECT SETTINGS:
//
//     IMO setting (SysClk)  = 24MHz     System clock is set to 24MHz
//     VC1=SysClk/1  = 16 (default)
//
// USER MODULE PARAMETER SETTINGS:
//
//  ------------------------------------------------------------------------------
//  UM         Parameter         Value                 Comments
//  ------------------------------------------------------------------------------
//  OneShot    Name              OneShot               UM's instance name
//             Clock             VC1
//             Input             Row_0_Input_1
//             Output            Row_0_Output_0
//             ClockSync         SyncSysClk
//             InvertInput       Normal
//
//  ------------------------------------------------------------------------------

/* Code begins here */

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

void main(void)
{
    // M8C_EnableGInt ; // Uncomment this line to enable Global Interrupts

    OneShot_Start();   // Start the OneShot!

    while(1)
    {
        // Insert your main routine code here.
    }
}
```

The following is the code in assembly.

```
;
; This sample shows how to generate single pulses with a 5uS delayed in response to an
; input signal.
```

```
;
;OVERVIEW:
;
; The OneShot output can be routed to any pin.
; The OneShot input can be routed to any pin.
; In this example the OneShot output is routed to P0[0] and the OneShot input is routed
; to P0[1].
;
;The following changes need to be made to the default settings in the Device Editor:
;
;   1. Select OneShot user module.
;   2. Select the 8 bit OneShot configuration in MUM
;   3. The User Module will occupy the space in dedicated system resources.
;   4. Rename User Module's instance name to OneShot.
;   5. Set OneShot's Clock Parameter to VC1.
;   6. Set OneShot's Input Parameter to Row_0_Input_1.
;   7. Set OneShot's Output Parameter to Row_0_Output_0.
;   8. Set OneShot's ClockSync Parameter to SyncSysClk.
;   9. Click on Row_0_Output_0 and connect Row_0_Output_0 to GlobalOutEven_0.
;   10.Click on Port_0_1 and set the Select parameter to GlobalInEven_1.
;   11.Select GlobalOutEven_0 for P0[0] in the Pinout.
;
; CONFIGURATION DETAILS:
;
;  1. The UM's instance name must be shortened to OneShot.
;
; PROJECT SETTINGS:
;
;     IMO setting (SysClk)  = 24MHz     System clock is set to 24MHz
;     VC1=SysClk/1  = 16 (default)
;
; USER MODULE PARAMETER SETTINGS:
;
; ------------------------------------------------------------------------------
;  UM         Parameter         Value                Comments
; ------------------------------------------------------------------------------
;  OneShot    Name              OneShot              UM's instance name
;             Clock             VC1
;             Input             Row_0_Input_1
;             Output            Row_0_Output_0
;             ClockSync         SyncSysClk
;             InvertInput       Normal
;
;  ------------------------------------------------------------------------------
;
; Code begins here

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:
```

```
    ; M8C_EnableGInt    ; Uncomment this line to enable Global Interrupts
    lcall OneShot_Start ; Start the OneShot!

.terminate:
    ; Insert your main assembly code here.
    jmp .terminate
```

Configuration Registers

Configuration registers vary depending on the bit width of your OneShot User Module.

8-Bit OneShot Configuration Registers

The 8-bit OneShot uses a single digital PSoC block named OneShot8. Each 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 (DxBxxFN), Bank 1

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

BCEN gates the compare output onto the row broadcast bus line.

Table 2. Input Register (DxBxxIN), Bank 1

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

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

Table 3. Output Register (DxBxxOU), 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 (DxBxxDR0), Bank 0

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

Shift Register is the OneShot stochastic counter Shift register.

Table 5. Polynomial Register (DxBxxDR1), Bank 0

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

Table 6. Compare Register (DxBxxDR2), Bank 0

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

Table 7. Control Register (DxBxxCR0), Bank 0

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

Start/Stop indicates that the OneShot is enabled when set. It is modified using the OneShot API.

16-Bit OneShot Configuration Registers

The 16-bit OneShot uses two digital PSoC blocks. In placement order from left to right they are named OneShot_LSB and OneShot_MSB. Each 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 8. Function Register (DxBxxFN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	BCEN	1	0	0	0	1	0
LSB	0	0	0	0	0	0	1	0

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.

Table 9. Input Register (DxBxxIN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock			
LSB	Input				Clock			

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

Table 10. Output Register (DxBxxOU), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	ClockSync		Output			0	0	0
LSB	ClockSync		0	0	0	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 11. Shift Register (DxBxxDR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Shift Register(MSB)							
LSB	Shift Register(LSB)							

Table 12. Polynomial Register (DxBxxDR1), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

Table 13. Compare Register (DxBxxDR2), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

Table 14. Control Register (DxBxxCR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the OneShot is enabled when set. It is modified using the OneShot API.

24-Bit OneShot Configuration Registers

The 24-bit OneShot uses three digital PSoC blocks. In placement order from left to right they are named OneShot_LSB, OneShot_ISB, and OneShot_MSB. Each 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 15. Function Register (DxBxxFN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	BCEN	1	0	0	0	1	0
ISB	0	0	0	0	0	0	1	0
LSB	0	0	0	0	0	0	1	0

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.

Table 16. Input Register (DxBxxIN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock			
ISB	0	0	1	1	Clock			
LSB	Input				Clock			

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

Table 17. Output Register (DxBxxIN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	ClockSync		Output			0	0	0
ISB	ClockSync		0	0	0	0	0	0
LSB	ClockSync		0	0	0	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 18. Shift Register (DxBxxDR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Shift Register(MSB)							
ISB	Shift Register(ISB)							
LSB	Shift Register(LSB)							

Table 19. Polynomial Register (DxBxxDR1), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

Table 20. Compare Register (DxBxxDR2), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

Table 21. Control Register (DxBxxCR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the OneShot is enabled when set. It is modified using the OneShot API.

32-Bit OneShot Configuration Registers

The 32-bit OneShot uses four digital PSoC blocks. In placement order from left to right they are named OneShot_LSB, OneShot_ISB1, OneShot_ISB2 and OneShot_MSB. Each 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 22. Function Register (DxBxxFN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	BCEN	1	0	0	0	1	0
ISB2	0	0	0	0	0	0	1	0
ISB1	0	0	0	0	0	0	1	0
LSB	0	0	0	0	0	0	1	0

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.

Table 23. Input Register (DxBxxIN), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock			
ISB2	0	0	1	1	Clock			
ISB1	0	0	1	1	Clock			
LSB	Input				Clock			

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

Table 24. Output Register (DxBxxOU), Bank 1

Block/Bit	7	6	5	4	3	2	1	0
MSB	ClockSync		Output			0	0	0
ISB2	ClockSync		0	0	0	0	0	0
ISB1	ClockSync		0	0	0	0	0	0
LSB	ClockSync		0	0	0	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 25. Shift Register (DxBxxDR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	Shift Register(MSB)							
ISB2	Shift Register(ISB2)							
ISB1	Shift Register(ISB2)							
LSB	Shift Register(LSB)							

Table 26. Polynomial Register (DxBxxDR1), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB2	0	0	0	0	0	0	0	0
ISB1	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

Table 27. Compare Register (DxBxxDR2), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB2	0	0	0	0	0	0	0	0
ISB1	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0

Table 28. Control Register (DxBxxCR0), Bank 0

Block/Bit	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0
ISB2	0	0	0	0	0	0	0	0
ISB1	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the OneShot is enabled when set. It is modified using the OneShot 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 © 2005-2013 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.