Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec No: 001-31343

Spec Title: PSOC(R) 1 PSEUDO-RANDOM SEQUENCE GENERATOR USER MODULE AS A ONE-SHOT PULSE WIDTH DISCRIMINATOR AND DEBOUNCER - AN2249

Sunset Owner: Meenakshi Sundaram Ravindran (MSUR)

Replaced By: 001-16681

PSoC® 1 Pseudo-Random Sequence Generator User Module as a One-Shot Pulse Width Discriminator and Debouncer

AN2249

Author: Ilya Mamontov
Associated Project: Yes
Associated Part Family: CY8C29x66, CY8C28xxx, CY8C27x43, CY8C24xxx, CY8C21xxx
Software Version: PSoC® Designer ™ 5.1 SP1
Associated Application Notes: AN2108, AN2156, AN2231

OBSOLETE

Application Note Abstract

AN2249 describes how to use a Pseudo-random sequence generator (PRS) user module to debounce noisy comparator output signals. Real world signals often cross comparator trip points multiple times as they transition. Most often, these multiple transitions are unwanted. A PRS user module may be configured as a one-shot and used to debounce these signals. An example with multiple implementations is demonstrated.

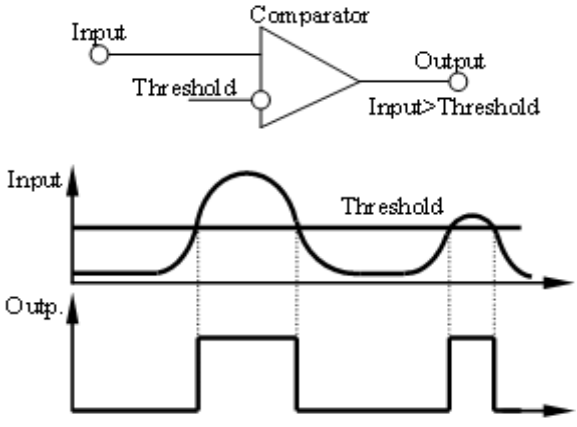
Introduction

Sometimes a device must produce a single pulse in response to an input signal. Such devices are called one-shots (or “univibrators” or “monostable multivibrators”) and are used to delay and reshape input pulses.

One-shots are also used as debouncers. The source of the bounce does not always originate from mechanical switches.

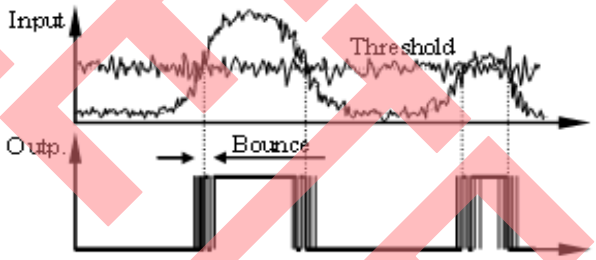
The input signal has curved rising and falling edges. A comparator responds to an incoming pulse by triggering a signal as shown in Figure 1.

Figure 1. Ideal Operation of a Comparator



In reality, the input pulses (and thresholds) have a noise component. The comparator also has its own noise. As a result, it responds with multiple triggers when the signal crosses the threshold (Figure 2). This phenomenon is called “bouncing.”

Figure 2. Actual Comparator Operation Waveforms



One way to eliminate false triggers is to add a hysteresis to the comparator. Application Note *AN2108 - Standard - Hysteresis Comparator with PSoC,* describes this method. *AN2156* also describes how to use a comparator with hysteresis in PSoC 1. However, it may not always be possible to correctly route the signals because of the chip’s architecture (for example, see Application Note *AN2231 - Standard - Ratemeter with a Precise Pulse Discriminator for Spectrometry*).

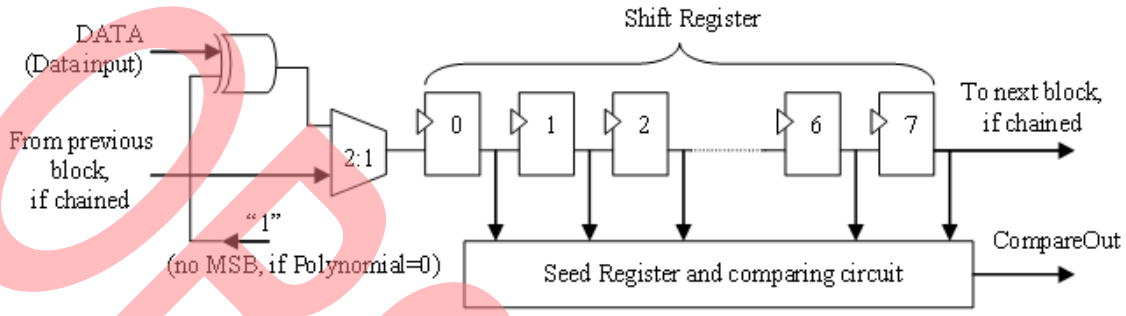
An alternative method of generating pulses is to use one-shots. One-shots are PWMs that produce a set length pulse when triggered. When the first pulse is sent out for a specified period of time, the one-shot ignores any other input pulses. As a result, bounces are removed, and very clean output pulses are received.

PSoC Implementation

The PRS user module is capable of acting as a one shot PWM when properly configured. This configuration is achieved when its polynomial register is set to 0. The internal block structure of the PRS with a zero polynomial value is shown in Figure 3.

In this mode of operation, the value of the Seed register should be set to 00000000 or 11111111 and the type of compare function must be “Equal.”

Figure 3. PRS Block Structure (Polynomial Register=00000000)



In this configuration, the PRS Shift register has no real data output, and the data input is inverted (by an XOR with “1”). The PRS user module typically uses a clock as its only input, but to create a one-shot from it, we must manually connect our input signal. To route the input, the control registers must be populated in code. This is shown in Code 1.

Table 1 lists all possible connections and code values for the input.

Code 1. Modifying the PRS Control Registers in Assembly

```
M8C_SetBank1
;prepare (clear) 7..4 bits
and reg[PRS8_1_INPUT_REG],0x0F
;connect to Row_N_Input_3
or  reg[PRS8_1_INPUT_REG],0xF0
;the Data Input should be inverted
;(for normal operation)
or  reg[PRS8_1_FUNC_REG],0x80
M8C_SetBank0
```

Note `PRS8_1_INPUT_REG` is an alias for the DxBxxIN register.

Table 1. Possible PRS Input Connections (DxBxxIN)

Code	Connection To
00h	Low Level (0)
10h	High Level (1)
20h	Row Broadcast Net
30h	Chain Function To Previous Block
40h	Analog Column Comparator 0
50h	Analog Column Comparator 1
60h	Analog Column Comparator 2
70h	Analog Column Comparator 3
80h	Row Output 0
90h	Row Output 1
A0h	Row Output 2
B0h	Row Output 3
C0h	Row Input 0
D0h	Row Input 1
E0h	Row Input 2
F0h	Row Input 3

Note This information may also be found in the PSoC 1 Technical Reference Manual (TRM), in the register reference.

OBSOLETE

Operation of the One-Shot

Operation of the one-shot executes as follows:

1. With every clock tick, the shift register is filled with a uniform sequence of 0s or 1s (defined by data input).
2. The comparing circuit sets the output to 0 or 1 (defined in the Seed register). This is a “stable” (or initial) state of the one-shot.
3. With varying input data, the shift register is filled and the comparing circuit reacts to the combinations by outputting a 1 or 0.
4. The input then returns to initial state and the shift register is filled by the initial sequence (returned to “stable” state).

The operating variants of the 8-bit PRS with different parameters are listed in Table 2. The operating diagrams are shown in Figure 5 on page 4, Figure 6 on page 4, Figure 7 on page 4, and Figure 8 on page 5.

The configuration code for the PRS can be found in the *main.asm* file in the associated project.

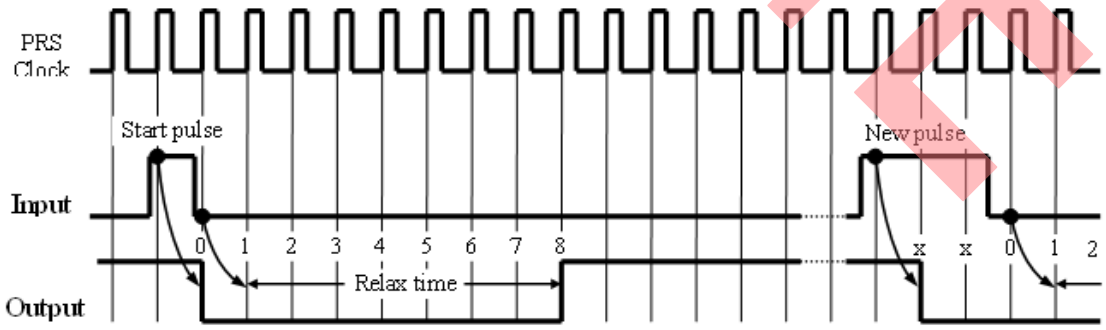
Table 2. Operating Variants for the 8-Bit PRS

Initial Data Input	Seed Register Value	Initial Output	Operating as One-Shot	Operating as Pulse Width Discriminator	Operating as Debouncer
0	00000000	1	First “1” triggers the output to low for 8 ticks; every subsequent “1” prolongs the low state for 8 ticks (Figure 4).		At bouncing, the output is active, there is an 8-tick “relax time” after bouncing (Figure 6).
0	11111111	0	Only 8 or more adjacent “1s” set the output to high. First incoming “0” triggers the output to low (Figure 5).	Only 8 or more ticks of “1” produce the output pulse (Figure 5).	At bouncing, the output is in initial state, there is an 8- tick “latent time” (delay between first non-bouncing pulse and output trigger). (Figure 7)
1	00000000	0	Only 8 or more adjacent “0s” set the output to high. First incoming “1” triggers the output to low.	Only 8 or more ticks of “0” produce the output pulse.	At bouncing, the output is in initial state, there is an 8- tick “latent time” (delay between first non-bouncing pulse and output trigger).
1	11111111	1	First “0” triggers the output to low for 8 ticks; every subsequent “0” prolongs the low state for 8 ticks.		At bouncing, the output is active; there is an 8- tick “relax time” after bouncing.

Important Notes

1. All reactions are delayed by 1 tick of the PRS clock as shown in Figure 4 on page 3, Figure 5 on page 4, Figure 6 on page 4, and Figure 7 on page 4 (a feature of the internal comparing circuit).
2. The input is sampled by a rising edge of the PRS clock.
3. The input is inverted by setting bit 7 in the DxBxxFN register (alias, PRS_FUNC_REG) to “1” in order to obtain “normal” (that is transparent) input logic. You must not modify this register to obtain the “inverted” input logic.
4. Also, you can use the “Less Than” compare function with a Seed Value=11111111 to obtain inverted output logic.

Figure 4. One-Shot Operation (Seed Register=00000000)



OBSOLETE

Figure 5. Pulse Width Selection (Seed Register=11111111)

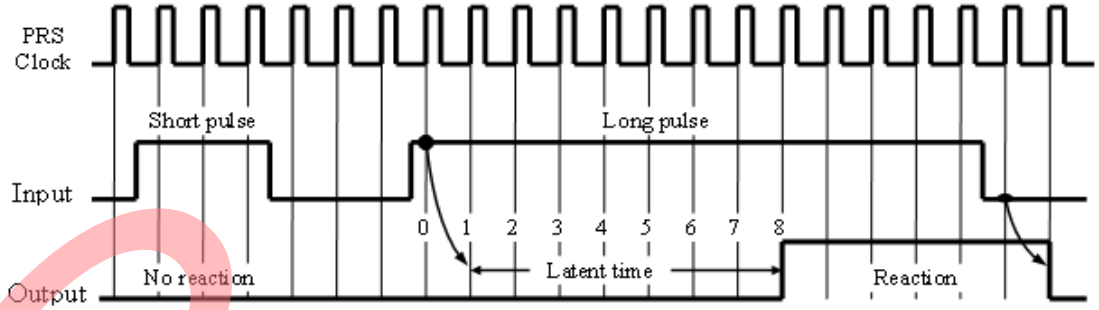


Figure 6. Pulse Width Denouncing (Seed Register=00000000) Output Pulse is “Longed”

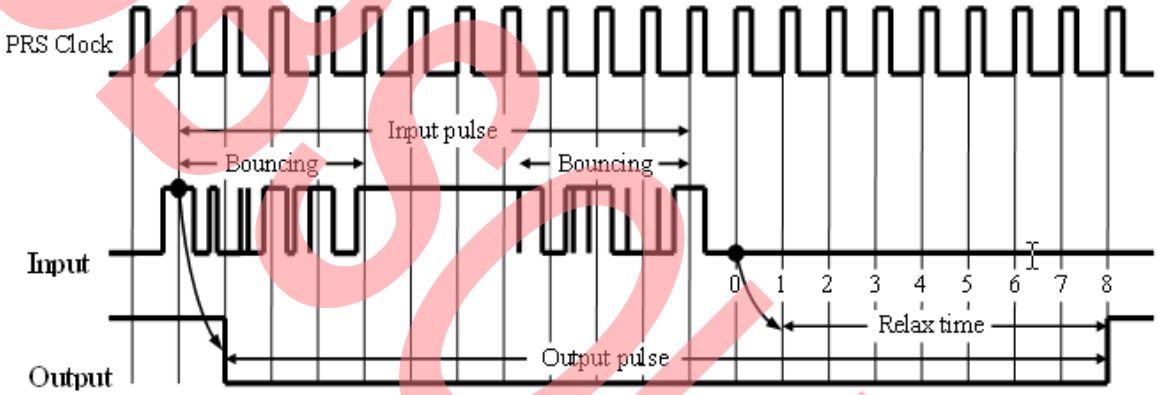
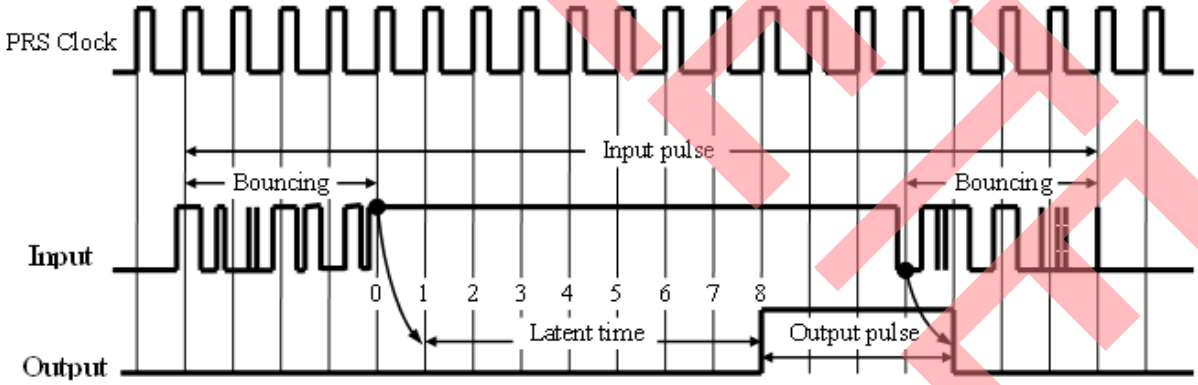


Figure 7. Pulse Width Debouncing (Seed Register=11111111) Output Pulse is “Shorted”



OBSOLETE

Notes

The PRS User Module is useful because it allows the user to choose 8-, 16-, 24-, or 32-bit capacity to obtain a required clock reaction time. The CRC16 Generator User Module should be used instead of a 16-bit PRS. (The DigBuf is exclusively 8 bits.) With the CRC, the data input can easily be reconfigured in the Device Editor.

Also note that a special instruction sequence (in, for instance, *main.asm*) can eliminate the long initial (power-on) output pulse of the one-shot using a Seed register value of 11111111. The one-shot with a Seed register value of 00000000 always starts with a one-tick reaction. The user should be prepared to accommodate this, if necessary.

In this configuration, input is sampled on a rising edge of the PRS clock. If the input pulse is too short and does not coincide with a rising edge, it is ignored. This is not a problem if you use a PRS clock-based (or slower) input signal. In other cases, you should anticipate probable losses.

A second important note, a CRC- and PRS-based user module should not be placed in the high adjacent position of the PSoC® 1 digital array. These types of user module use the internal, common MSB tristate bus and disturb one-shot operation.

Testing

To test, create the circuit shown in Figure 8 and program the part using the associated project. The sequences from port 2 should be connected as the input to port 7, one at a time. They may also be connected to an LED for observation. The outputs of the two different PRS user modules can be observed on the other LEDs.

The sequences from ports P2 [7], P2 [5], P2 [3], and P2 [1] have a sample rate of 8 Hz and are 16 bits in length. The "bouncing" with a sub-pulse period of 0.25 s can easily be seen.

P2 [7] Output – Short Pulses

```
1000000000000000100000000000000...
```

P2 [5] Output – Long Pulses

```
1111111100000000111111110000000...
```

P2 [3] Output – Short Pulses with "Bouncing"

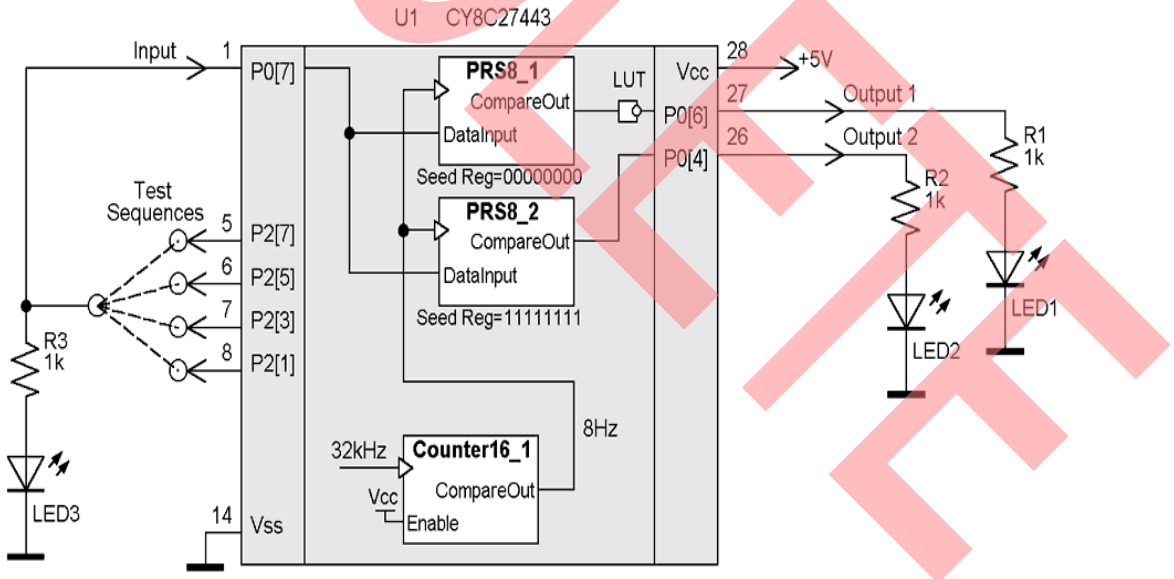
```
1011010000000000101101000000000...
```

P2[1] Output – Long Pulses with "Bouncing"

```
1010111111110000101011111110000...
```

OBSOLETE

Figure 8. PSoC 1 Internal and External Configuration for Project



Test Results

The reactions to the test sequences were as follows:

- P2[7] - LED1 responded with an 8-tick flash duration. LED2 had no response (input pulses are too short).
- P2[5] - LED1 responded with a 15-tick flash duration. LED2 responded with a 1- tick flash duration.
- P2[3] - LED1 responded with a 13-tick flash duration. LED2 had no response (input pulses are too short).
- P2[1] - LED1 stayed lit continuously (the intervals between pulses were too short). LED2 responded with a 1-tick flash duration.

During testing, the PRS responded with a single pulse, despite the signal bouncing, as expected.

Summary

The CRC- and PRS-based user modules may be configured as one-shot pulse width discriminators and debouncers. This allows you to reshape pulses in different applications.

There are a few disadvantages to this approach:

- Only 8 bits may be used.
- CRC- and PRS-based user modules should not be placed in the high adjacent position of the digital block array.
- Input pulses that occur between clock rising edges will be missed.

About the Author

Name:	ILYA Mamontov
Title:	Electronic Engineer
Background:	Design, modernization, repair, and technical service of measurement devices.
Contact:	ilka@elsite.ru Subject: PSOC AN or illinoys@narod.ru Subject: PSOC AN

OBSOLETE

Document History Page

Document Title: PSoC® 1 Pseudo-Random Sequence Generator User Module as a One-Shot Pulse Width Discriminator and Debouncer – AN2249

Document Number: 001-31343

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1486965	MAXK	09/25/2007	Original Document.
*A	3223317	MAXK	04/25/2011	Updated project and document for PSoC Designer 5.1 SP1 Update title and abstract. Miscellaneous updates.
*B	3353336	MAXK	08/24/2011	No change.
*C	4506771	MSUR	09/18/2014	Obsolete document.

OBSOLETE

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," and PSoC Designer are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/

© Cypress Semiconductor Corporation, 2007-2014. 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.

This 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.