

## AN2326

**Author:** Volodymyr Sokil

**Associated Project:** Yes

**Associated Part Family:** All

**Software Version:** PSoC Designer™ 5.1

**Associated Application Notes:** [AN2249](#)

### Application Note Abstract

This application note describes how to implement a bitstream sequence recognizer in hardware using the PSoC® Pseudo Random Sequence Generator (PRS) user module. The PRS can be used in digital communication systems with the serial data interface for automatic preamble detection and extraction, control words selection, and so on.

### Introduction

Communication systems often use a serial interface for data transmission. In such cases, the bit flow must contain auxiliary service bit sequences, such as a preamble sequence for the receiver adjustment or a synchronization sequence (synchroword) for byte parsing.

The preamble sequences can be extracted on both the firmware and hardware levels, but the receiver synchronization can be performed only in hardware. The

common and necessary element of these applications is the bitstream sequence recognizer. An application of this type, with 8, 16, 24, or 32 bits, can be easily built on the PSoC device PRS user module.

### Recognizer Implementation

The PRS block diagram is shown in [Figure 1](#). It is a modular linear feedback shift register, which generates a pseudo random bit sequence.

Figure 1. PRS User Module Block Diagram (Data Width  $n = 8, 16, 24, 32$ )

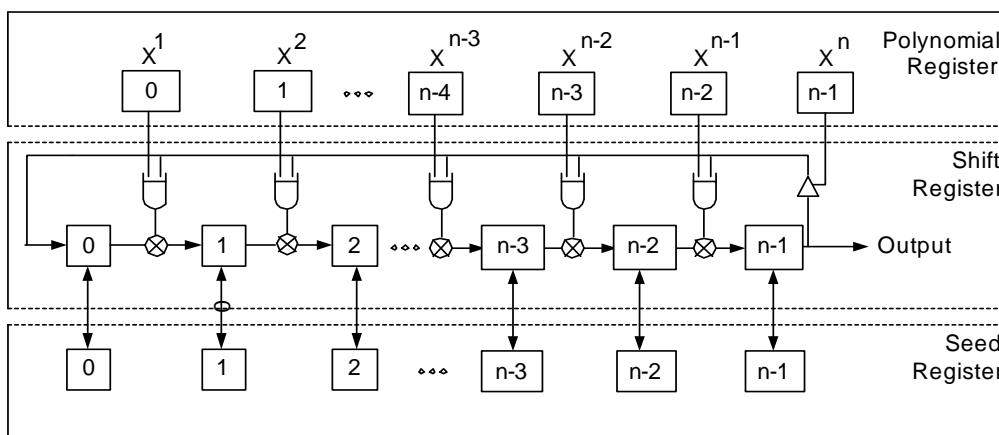
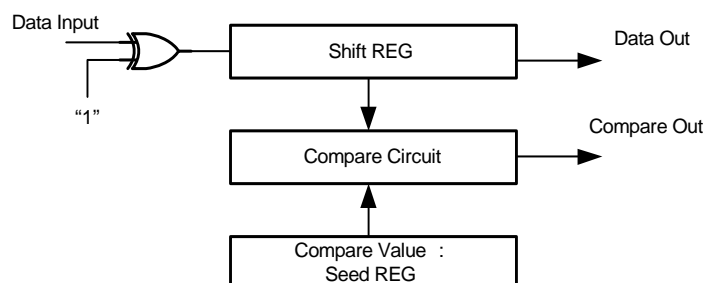


Figure 2. PRS User Module Block Diagram (Polynomial Register = 0x00, Data Width n = 8)



The Polynomial register value defines the internal block structure. If this register initializes to a zero value, the PRS transfers to the Shift register (Figure 2). This register has an important feature—the value of this register can be compared with a predefined Seed register value. This is an excellent base for the bits fragment recognizer.

The Shift register in the standard PRS mode has no data input and output connections. These connections must be defined manually by initializing the special Control register. In this application, it is necessary only to route a register data input and clock to the internal structure of the target device. The clock source is easily set in PSoC Designer™ Device Editor. The input data source is defined by the PRSxx\_x\_INPUT\_REG (DxBxxIN) register. Table 1 shows all possible values of this register.

Note that the input data stream is inverted by the input exclusive-OR circuit. To return the input data to its normal status, set the Data Invert bit (0x80 mask) in the function register PRSxx\_x\_FUNC\_REG (DxBxxFN).

The CompareType parameter can be set to 'Equal', 'Less Than', or 'Less Than or Equal' in the Device Editor. In this case, set the CompareType parameter to 'Equal'.

The bitstream sequence, which needs to be recognized, is loaded into the PRS Seed register. If the input bitstream contains the defined bits, the Seed register and Shift register values are equal to each other. The CompareOut parameter signal goes to active high.

Note that on startup, the Seed register value is copied into the Shift register, causing the compare circuit to trigger immediately. Therefore, skip the first CompareOut signal, after the module has started.

Table 1. Input Data Source Connection

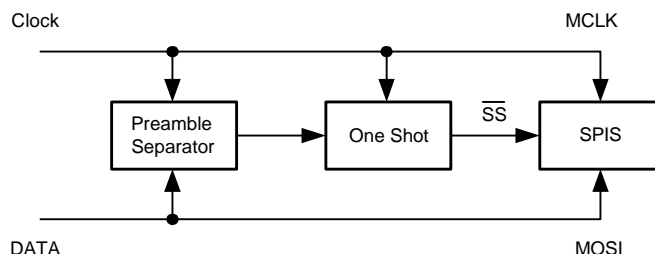
Value	Connection
0x00	Low Level (0)
0x10	High Level (1)
0x20	Row Broadcast Net
0x30	Chain Function To Previous Block
0x40	Analog Column Comparator 0
0x50	Analog Column Comparator 1
0x60	Analog Column Comparator 2
0x70	Analog Column Comparator 3
0x80	Row Output 0
0x90	Row Output 1
0xA0	Row Output 2
0xB0	Row Output 3
0xC0	Row Input 0
0xD0	Row Input 1
0xE0	Row Input 2
0xF0	Row Input 3

The CompareOut parameter can be used as a control signal for different hardware receivers (for example, as a slave select signal for the SPIS User Module). Moreover, the interrupts can be used to control firmware. To enable the interrupts, the corresponding block mask must be set in the Interrupt Mask Register, INT\_MSKx. The interrupt is triggered by the rising edge of the CompareOut signal.

## Practical Example

Consider an example of the bitstream sequence recognizer application—a serial bit receiver with automatic preamble synchronization. The test communication system uses a two-wire serial interface with clock and data signals. The transmitter inserts 16 bits of preamble in the output bitstream. The first byte (0xFF) is the synchronization flag; the second byte (0x0F) is the “dummy” for the receiver resynchronization. This byte is omitted by the receiver. [Figure 3](#) illustrates the receiver flowchart.

Figure 3. Receiver Flowchart



The SPIS User Module is used as the hardware receiver. For byte synchronization, delay the preamble separator CompareOut signal for six and a half MCLK periods. This delay is provided by the one-shot user module, which uses the inverted clock signal. Implementation and operation principles of the one-shot user module are described in [AN2249 - PSoC® 1 - PRS User Module as a One-Shot Pulse Width Discriminator and Debouncer](#). The delayed signal is used as the SPIS slave select input signal.

The internal structure of the receiver and oscilloscope waveforms for this implementation is shown in [Figure 4](#) and [Figure 5](#), respectively.

Figure 4. PSoC Receiver Internal User Module Configuration

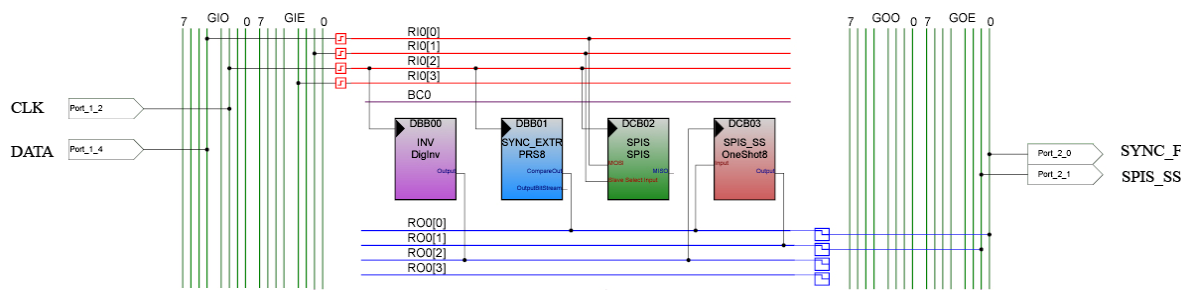
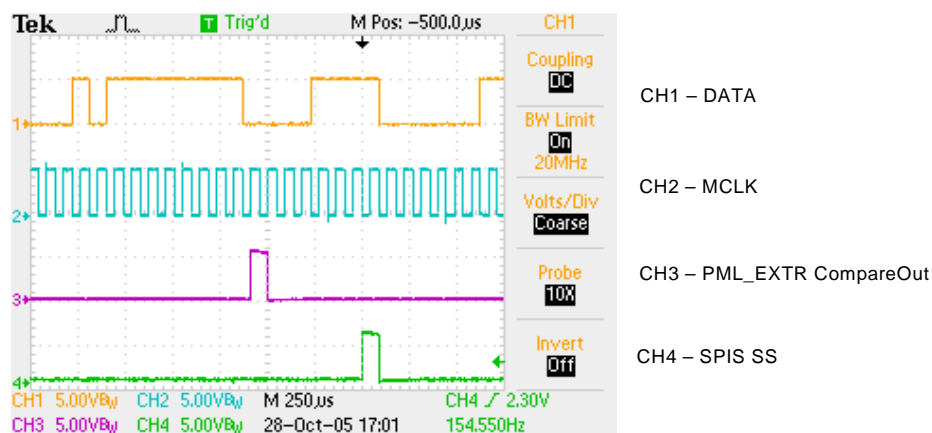


Figure 5. Receiver Preamble Synchronization Waveform



## About the Author

**Name:** Volodymyr Sokil

**Title:** Associate Professor

**Background:** Volodymyr obtained a PhD degree in computing machines, systems, and networks in 2007 from the National University, Lvivska Polytechnika (Lviv, Ukraine). His interests include embedded systems design and information security.

**Contact:** [vsok@cypress.com](mailto:vsok@cypress.com)

## Document History

**Document Title:** Hardware Bitstream Sequence Recognizer – AN2326

**Document Number:** 001-26059

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1448764	SSFTMP4	09/20/2007	New application note.
*A	3025195	YARD_UKR	09/08/2010	Updated PSoC Designer version. Minor content edits.
*B	3183319	BIOL_UKR	03/23/2011	Minor content edits.

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. PSoC Designer is a trademark 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, 2005–2011. 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.