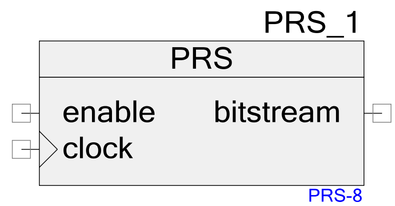
Pseudo Random Sequence (PRS)

1.10

Features



- 2 to 64-bit PRS sequence length
- Serial output bit stream
- Continuous or single step run modes
- Standard or custom polynomial
- Standard or custom seed value
- Enable input provides synchronized operation with other components
- Computed pseudo-random number can be read directly from the Linear Feedback Shift Register (LFSR)

General Description

The Pseudo Random Sequence (PRS) component uses a LFSR to generate a pseudo random sequence, which outputs a pseudo random bitstream. The LFSR is of the Galois form (sometimes known as the modular form) and utilizes the provided maximal code length, or period. The PRS component runs continuously after started as long as Enable Input is held high. The PRS number generator may be started with any valid seed value excluding 0.

When to use a PRS

LFSRs can be implemented in hardware, and this makes them useful in applications that require very fast generation of a pseudo-random sequence, such as direct-sequence spread spectrum radio.

Global positioning systems use an LFSR to rapidly transmit a sequence that indicates high-precision relative time offsets. Some video game consoles also use an LFSR as part of the sound system.

Uses as counters

The repeating sequence of states of an LFSR allows it to be used as a divider, or as a counter when a non-binary sequence is acceptable. LFSR counters have simpler feedback logic than natural binary counters or Gray code counters, and therefore can operate at higher clock rates. However it is necessary to ensure that the LFSR never enters an all-zeros state, for example by presetting it at start-up to any other state in the sequence.

PRELIMINARY

Input/Output Connections

This section describes the various input and output connections for the PRS Component. An asterisk (*) in the list of I/O's states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

clock – Input *

The clock input defines the signal to compute PRS. This input is not available when the single step run mode is chosen.

enable – Input

The PRS component runs after started and as long as the Enable input is held high. This input provides synchronized operation with other components.

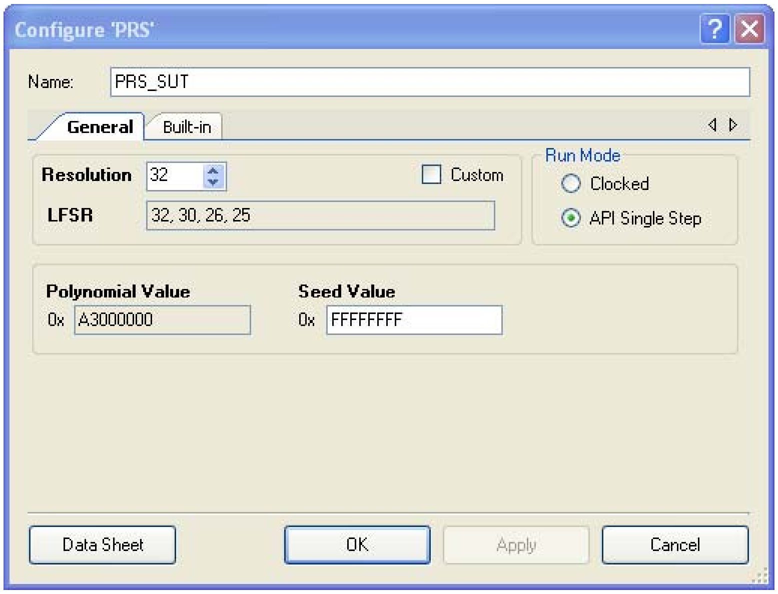
bitstream – Output

Output of the LFSR.

Parameters and Setup

Drag a PRS component onto your design and double-click it to open the Configure dialog. This dialog has several tabs to guide you through the process of setting up the PRS component.

Figure 1 Configure PRS Dialog



The PRS dialog contains the following settings:

Resolution

This defines the PRS sequence length. This value can be set from 2 to 64. The default is 8.

By default, Resolution defines LFSR coefficients and Polynomial Value. Coefficients are taken from the following table. This parameter also defines the maximal code length, or period, as shown in the following table.

Resolution	LFSR	Period ($2^{Resolution}$-1)
2	2, 1	3
3	3, 2	7
4	4, 3	15
5	5, 4, 3, 2	31
6	6, 5, 3, 2	63
7	7, 6, 5, 4	127
8	8, 6, 5, 4	255
9	9, 8, 6, 5	511
10	10, 9, 7, 6	1023
11	11, 10, 9, 7	2047
12	12, 11, 8, 6	4095
13	13, 12, 10, 9	8191
14	14, 13, 11, 9	16383
15	15, 14, 13, 11	32767
16	16, 14, 13, 11	65535
17	17, 16, 15, 14	131071
18	18, 17, 16, 13	262143
19	19, 18, 17, 14	524187
20	20, 19, 16, 14	1048575
21	21, 20, 19, 16	2097151
22	22, 19, 18, 17	4194303
23	23, 22, 20, 18	8388607
24	24, 23, 21, 20	16777215
25	25, 24, 23, 22	33554431
26	26, 25, 24, 20	67108863
27	27, 26, 25, 22	134217727
28	28, 27, 24, 22	268435455
29	29, 28, 27, 25	536870911
30	30, 29, 26, 24	1073741823
31	31, 30, 29, 28	2147483647
32	32, 30, 26, 25	4294967295
33	33, 32, 29, 27	8589934592

Resolution	LFSR	Period ($2^{Resolution}-1$)
34	34, 31, 30, 26	17179869184
35	35, 34, 28, 27	34359738368
36	36, 35, 29, 28	68719476736
37	37, 36, 33, 31	137438953472
38	38, 37, 33, 32	274877906944
39	39, 38, 35, 32	549755813888
40	40, 37, 36, 35	1099511627776
41	41, 40, 39, 38	2199023255552
42	42, 40, 37, 35	4398046511104
43	43, 42, 38, 37	8796093022208
44	44, 42, 39, 38	17592186044416
45	45, 44, 42, 41	35184372088832
46	46, 40, 39, 38	70368744177664
47	47, 46, 43, 42	140737488355328
48	48, 44, 41, 39	281474976710656
49	49, 45, 44, 43	562949953421312
50	50, 48, 47, 46	1125899906842624
51	51, 50, 48, 45	2251799813685248
52	52, 51, 49, 46	4503599627370496
53	53, 52, 51, 47	9007199254740992
54	54, 51, 48, 46	18014398509481984
55	55, 54, 53, 49	36028797018963968
56	56, 54, 52, 49	72057594037927936
57	57, 55, 54, 52	144115188075855872
58	58, 57, 53, 52	288230376151711744
59	59, 57, 55, 52	576460752303423488
60	60, 58, 56, 55	1152921504606846976
61	61, 60, 59, 56	2305843009213693952
62	62, 59, 57, 56	4611686018427387904
63	63, 62, 59, 58	9223372036854775808
64	64, 63, 61, 60	18446744073709551616

To set LFSR coefficients manually:

1. Define **Resolution**.
2. Check the **Custom** check box.
3. Enter coefficients separated by comma in the LFSR text box and press [**Enter**]. The Polynomial value will be recalculated automatically.

The Polynomial value is represented in hexadecimal.

Note No LFSR coefficient value can be greater than the **Resolution** value.

The Seed value by default is set to the maximum possible value ($2^{Resolution}$ - 1). Its value can be changed to any other except 0. The Seed value is represented in hexadecimal.

Note Changing the Resolution resets the Seed value to the default value.

Run Mode

This parameter defines continuous or single step run modes. This parameter defines the component operation mode. Possible values include: “Clocked” (default) and “APISingleStep”.

Local Parameters (For API usage)

These parameters are used in the API and are not exposed in the GUI:

- **PolyValueLower (uint32) –** Contains the lower half of the polynomial value in hexadecimal. The default is 0xB8h (LFSR= [8,6,5,4]) because the default Resolution is 8.
- **PolyValueUpper (uint32) –** Contains the upper half of the polynomial value in hexadecimal. The default is 0x00h because the default Resolution is 8.
- **SeedValueLower(uint32) –** Contains the lower half of the seed value in hexadecimal. The default is 0xFFh because the default Resolution is 8.
- **SeedValueUpper(uint32) –** Contains the upper half of the seed value in hexadecimal. The default is 0 because the default Resolution is 8.

Clock Selection

There is no internal clock in this component. You must attach a clock source.

Note Generation of proper PRS sequence for Resolution, which is greater 8, requires two clock transitions.

Placement

The PRS is placed throughout the UDB array and all placement information is provided to the API through the cyfitter.h file.

Resources

Resolution	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
8-Bits	1	?	0	1	0	?	?	?
16-Bits	1	?	0	1	0	?	?	?
24-Bits	2	?	0	1	0	?	?	?
32-Bits	2	?	0	1	0	?	?	?
40-Bits	3	?	0	1	0	?	?	?
48-Bits	3	?	0	1	0	?	?	?
56-Bits	4	?	0	1	0	?	?	?
64-Bits	4	?	0	1	0	?	?	?

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 "PRS_1" to the first instance of a component in a given design. You can 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 "PRS."

Function	Description
void PRS_Start(void)	Initializes the seed and polynomial registers. The PRS computation starts on the rising edge of the input clock.
void PRS_Stop(void)	Stops PRS computation; the PRS is stored in the PRS register.
void PRS_ Step(void)	Increments the PRS by one when in API single step mode.
void PRS_WriteSeed(uint8/16/32 seed)	Writes the PRS Seed register with the start value.
void PRS_WriteSeedUpper(uint32 seed)	Writes the upper half of the Seed register with the start value. Only generated for 33-64-bit PRS.
void PRS_WriteSeedLower(uint32 seed)	Writes the lower half of Seed register with the start value. Only generated for 33-64-bit PRS.

Function	Description
uint8/16/32 PRS_Read(void)	Reads the current PRS value.
uint32 PRS_ReadUpper(void)	Reads the upper half of the current PRS value. Only generated for 33-64-bit PRS.
uint32 PRS_ReadLower(void)	Reads the lower half of the current PRS value. Only generated for 33-64-bit PRS.
void PRS_ WritePolynomial(uint8/16/32 polynomial)	Writes the PRS polynomial.
void PRS_WritePolynomialUpper(uint32 polynomial)	Writes the upper half of the PRS polynomial. Only generated for 33-64-bit PRS.
void PRS_WritePolynomialLower(uint32 polynomial)	Writes the lower half of the PRS polynomial. Only generated for 33-64-bit PRS.
uint8/16/32 PRS_ReadPolynomial(void)	Reads the PRS polynomial.
uint32 PRS_ReadPolynomialUpper(void)	Reads the upper half of the PRS polynomial. Only generated for 33-64-bit PRS.
uint32 PRS_ReadPolynomialLower(void)	Reads the lower half of the PRS polynomial. Only generated for 33-64-bit PRS.

void PRS_Start(void)

Description: Initializes the seed and polynomial registers. The PRS computation starts on the rising edge of the input clock.

Parameters: None

Return Value: None

Side Effects: None

void PRS_Stop(void)

Description: Stops PRS computation; The PRS is stored in the PRS register.

Parameters: None

Return Value: None

Side Effects: None

void PRS_Step(void)

Description: Increments the PRS by one when in API single step mode.

Parameters: None

Return Value: None

Side Effects: None

void PRS_WriteSeed(uint8/16/32 seed)

Description: Writes the PRS Seed register with the start value.

Parameters: (uint8/16/32) seed: Seed register start value.

Return Value: None

Side Effects: None

void PRS_WriteSeedUpper(uint32 seed)

Description: Writes the upper half of the Seed register with the start value. Only generated for 33-64-bit PRS.

Parameters: (uint32) seed: Upper half of Seed register start value.

Return Value: None

Side Effects: None

void PRS_WriteSeedLower(uint32 seed)

Description: Writes the lower half of Seed register with the start value. Only generated for 33-64-bit PRS.

Parameters: (uint32) seed: Lower half of Seed LSB register start value.

Return Value: None

Side Effects: None

uint8/16/32 PRS_Read(void)

Description: Reads the current PRS value.

Parameters: None

Return Value: (uint8/16/32) Current PRS value.

Side Effects: None

uint32 PRS_ReadUpper(void)

Description: Reads the upper half of the current PRS value. Only generated for 33-64-bit PRS.

Parameters: None

Return Value: (uint32) Upper half of the current PRS value.

Side Effects: None

uint32 PRS_ReadLower(void)

Description: Reads the lower half of the current PRS value. Only generated for 33-64-bit PRS.

Parameters: None

Return Value: (uint32) Lower half of the current PRS value.

Side Effects: None

void PRS_ WritePolynomial(uint8/16/32 polynomial)

Description: Writes the PRS polynomial.

Parameters: (uint8/16/32) polynomial: PRS polynomial.

Return Value: None

Side Effects: None

void PRS_ WritePolynomialUpper(uint32 polynomial)

Description: Writes the upper half of the PRS polynomial. Only generated for 33-64-bit PRS.

Parameters: (uint32) polynomial: Upper half of the PRS polynomial.

Return Value: None

Side Effects: None

void PRS_ WritePolynomialLower(uint32 polynomial)

Description: Writes the lower half of the PRS polynomial. Only generated for 33-64-bit PRS.

Parameters: (uint32) polynomial: Lower half of the PRS polynomial.

Return Value: None

Side Effects: None

uint8/16/32 PRS_ReadPolynomial(void)

Description: Reads the PRS polynomial.

Parameters: None

Return Value: (uint8/16/32) PRS polynomial.

Side Effects: None

uint32 PRS_ReadPolynomialUpper(void)

Description: Reads the upper half of PRS polynomial. Only generated for 33-64-bit PRS.

Parameters: None

Return Value: (uint32) Upper half of PRS polynomial.

Side Effects: None

uint32 PRS_ReadPolynomialLower(void)

Description: Reads the lower half of the PRS polynomial. Only generated for 33-64-bit PRS.

Parameters: None

Return Value: (uint32) Lower half of the PRS polynomial.

Side Effects: None

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the PRS component. This example assumes the component has been placed in a design with the default name PRS_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>
#include "PRS_1.h"

main()
{
    PRS_1_Start();
}
```

Functional Description

PRS Run Mode: Clocked

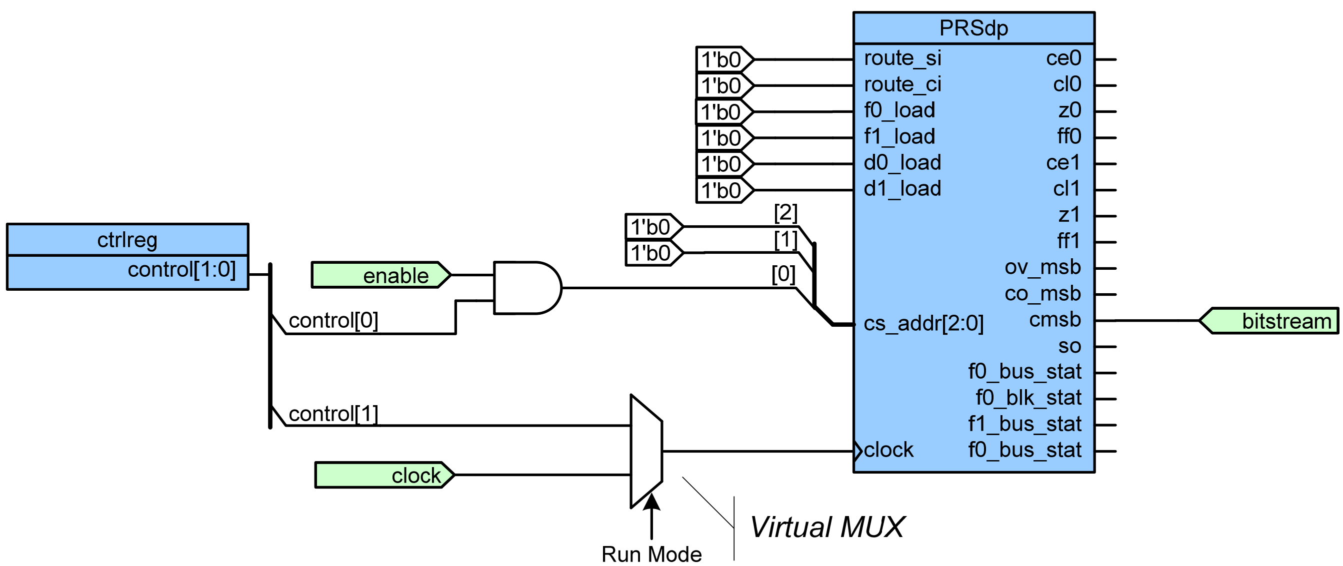
The PRS component runs continuously after it is started and as long as the Enable Input is held high.

PRS Run Mode: API Single Step

In this mode, the PRS is incremented by an API call.

Block Diagram and Configuration

The PRS is only implemented at a set of configured UDBs. The implementation is shown in the following block diagram.



Registers

TBD

References

Not applicable

DC and AC Electrical Characteristics

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Input Capacitance	---		---	pF	
Input Impedance	---		---	Ω	
Maximum Clock Rate	---		100	MHz	