Pseudo Random Sequence Generator

Example Name: Example_PRS8
Programming Language: C
Associated Part Families: CY8C24x23, CY8C27x43, CY8C29x66
CY8C24x94, CY8C21x34
Software Version: PSoC® Designer™ 5.2
Related Hardware: CY3210-PSoCEval1
Author: Dineshbabu Mani

Objective

This project demonstrates the operation of an 8-bit pseudo random sequence (PRS) generator using PSoC® 1.

Overview

An 8-bit PRS generator generates a random number sequence for the given modular polynomial and seed value at an interval of 10 ms and transmits the series using a TX8 serial transmitter. The generation of the PRS sequence is initiated by a switch press.

User Module List and Placement

The following table lists the hardware resources occupied by each user module.

User Module	Placement
Counter16	DBB01 (LSB), CB02(MSB)
PRS8	DBB00
TX8	DCB03

User Module Parameter Settings

The following table lists the parameter settings of the user modules.

Counter16		
Parameter	**Value**	**Comments**
Clock	VC2	Input clock is 100 kHz. (VC2 = VC1 / 15 where VC1 = Sys.Clk / 16)
Enable	High	Enables the counter
CompareOut	Row_0_Output_0	The output is used as clock for PRS generator
TerminalCountOut	None	Not applicable
Period	999	Divides the input clock by 1000 to generate the output frequency of 100 Hz
CompareValue	500	Sets the duty cycle of the Counter Output. Duty cycle is set to 50%
CompareType	Less Than or Equal	Not applicable
InterruptType	Terminal Count	Counter generates an interrupt on terminal count. PRS8 is read in the interrupt
ClockSync	Sync To SysClk	Synchronizes the input clock with source clock (Sys.Clk)
InvertEnable	Normal	Enables the Counter to Active High

PRS8		
Parameter	**Parameter**	**Parameter**
Clock	Row_0_Output_0	The clock to the PRS8 is derived from the Counter
OutputBitStream	None	Not applicable
CompareOut	None	Not applicable
CompareType	Less than Or Equal	Not applicable
ClockSync	Sync To SysClk	Synchronizes the input clock with source clock (Sys.Clk)

TX8		
Parameter	**Value**	**Comments**
Clock	VC3	Set to 153.846 KHz which is eight times 19.230 kbps (required Baud rate)
Output	Row_0_Output_1	Port pin P0.1 is assigned as the output for serial Transmitter. The output from the TX8 block is routed to this pin through the Row_0_Output_1 net and Global_Out_Even_1
TX Interrupt Mode	TxRegEmpty	Not applicable
ClockSync	Sync to SysClock	Synchronized with the SysClock (Source Clock)
Data Clock Out	None	Not applicable

Note: The clock to the TX8 user module should be eight times the desired baud rate.

Global Resources

Important Global Resources		
Parameter	**Value**	**Comments**
CPU Clock	SysClk/2	Sets the CPU frequency to 12 MHz
VC1	16	Divides the Sys.Clk by 16
VC2	15	Divides VC1 by 15
VC3 Source	SysClk/1	System clock (24 MHz) is the source for VC3 divider
VC3 Divider	156	Divides Sys.Clk by 156 to get 153.846 KHz clock which is eight times the desired baud rate (19.230 Kbps)

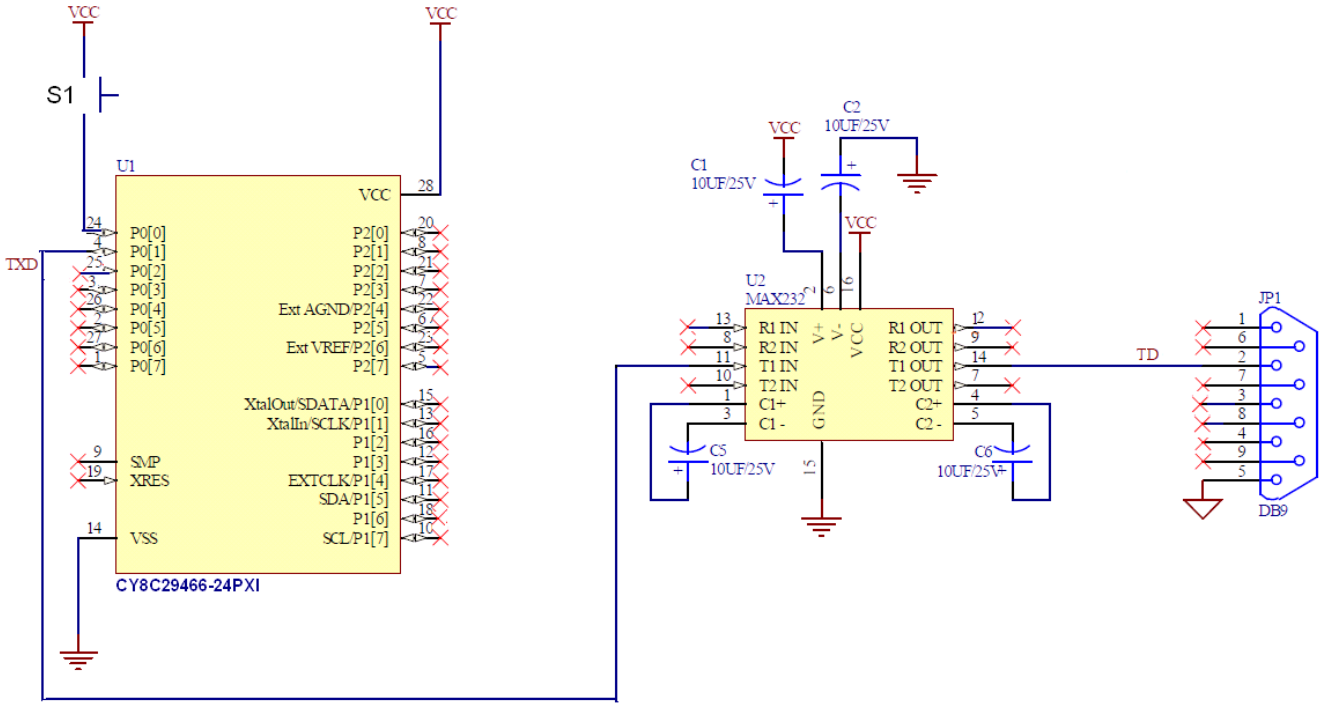
Note: Leave all other global resources at their default.

Pin Configuration

PinOut			
Pin	**Select**	**Drive**	**Interrupt**
Port 0_0	StdCPU	Pull-down	Rising edge
Port 0_1	GlobalOutEven_1	Strong	Disable Int

Hardware Connections

Figure 1. Project Schematic



U2 is MAX232, an RS232 transceiver, which is used to translate the TTL level TX signal from PSoC to ±10 V RS232 level. PSoC is connected to the PC through a JP1, a DB9 connector.

The code example can be tested using the CY3210-PSoCEval1 board. This board has an RS232 transceiver and a serial port connector. To test the code example using the CY3210 board, make the following connections:

- Connect P00 of J6 to SW of J5 (which is connected to switch S1)
- Connect P01 of J6 to TX of J13.

Operation

Load all the hardware settings from the device configuration into the device and execute *main.c*. The following operations are performed in *main.c*:

- Seed and polynomial for the PRS are set and the PRS is started. The input clock for the PRS generator is derived from a 16-bit counter. The counter divides the input clock of 100 kHz (VC2) by 1000 to produce a 100 Hz clock to the PRS generator. This code example uses the following polynomial and seed values:
  - Seed Value = 1
  - Modular Polynomial = (8, 6, 5, 4)
  - Code word Length = 255
- The polynomial and seed values are entered in hex form inside *main.c*. For example, if the modular polynomial and seed values are 8, 6, 5, 4, and 1 respectively, then they are entered in *main.c* in the following manner:
  - Modular polynomial → 0xb8 (in binary form 10111000)
  - Seed value → 0x01

    For more details about the modular polynomial and seed value representations, refer to PRS8 User Module datasheet.
- TX8 is started with no parity. The welcome string is sent over the serial port to the HyperTerminal after clearing the HyperTerminal by sending a new page character.

- GPIO interrupt is enabled on port pin P0 [0].This generates an interrupt when the switch is pressed.
- The rest of the operations are performed inside the GPIO and Counter ISR. The ISRs for GPIO and Counter are written in C and are declared as ISRs using the `#pragama interrupt_handler` directive. An `ljmp` instruction to the C GPIO_ISR is placed inside the *PSoCGPIOINT.asm* file within the user code markers. An `ljmp` instruction to the CounterISR is placed inside the interrupt handler function inside *CounterINT.asm*.

  GPIO ISR: The following operations are performed inside the GPIO interrupt when the switch is pressed.

  1. Bring the cursor to the beginning of line in HyperTerminal.
  2. Send the seed value over serial port as the first number in the random number sequence.
  3. Start Counter.
  4. Mask GPIO interrupts.
  5. Reading the random numbers now takes place inside the Counter's ISR.

  Counter ISR: The counter generates an interrupt every 10 ms. The following operations are performed inside this interrupt:

  1. Stop both PRS and Counter and get the random number from PRS module, by using the bReadPRS function. Store the value in the random sequence array in RAM.
  2. Check if the read number is equal to the seed value. If the number is equal to the seed it means that the PRS has generated one full sequence of random numbers. If the number is equal to the seed, perform the following operations:

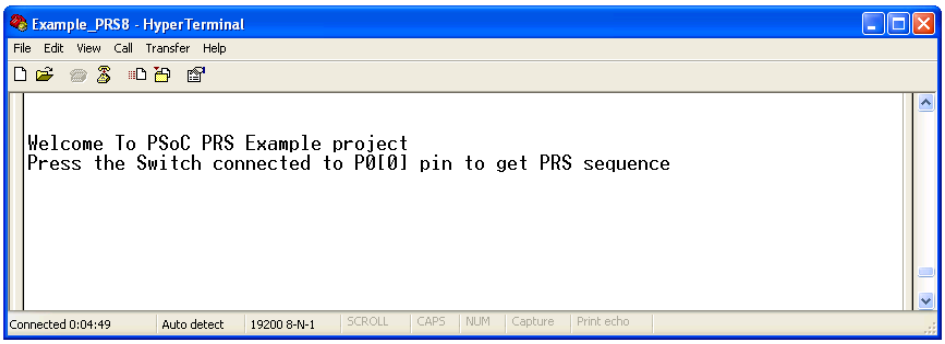
- Send the values of seed, polynomial, and length of sequence over serial port.
- Print Welcome.
- Reset the pointer to the random number array.
- Start PRS.
- Clear all posted GPIO interrupts.
- Unmask GPIO interrupt.

  3. If the random number is not equal to seed value:

- Send the obtained random number to HyperTerminal over serial port.
- Start PRS and Counter.

After setting up the HyperTerminal as explained in "Testing the Project" on page 6, click **Call** in the HyperTerminal and reset the PSoC. The HyperTerminal window displays the following message.

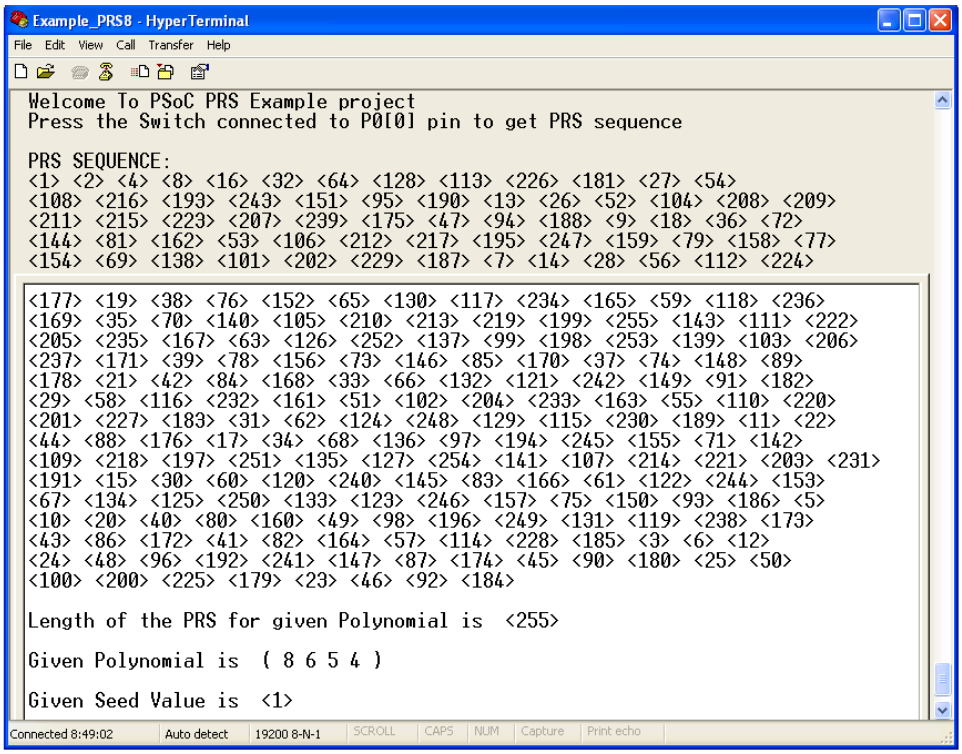
Figure 2. Welcome Screen on HyperTerminal



After the switch connected to the P0[0] pin is turned on, the PRS generator generates random numbers for every 10 ms and these are displayed in the HyperTerminal (a maximum of 13 random numbers in a row). The generated random numbers are stored in a character array and the length of PR sequence for the given polynomial is calculated by PSoC and is transmitted over the serial port to HyperTerminal.

The following figure shows the entire PR sequence and its length observed in HyperTerminal.

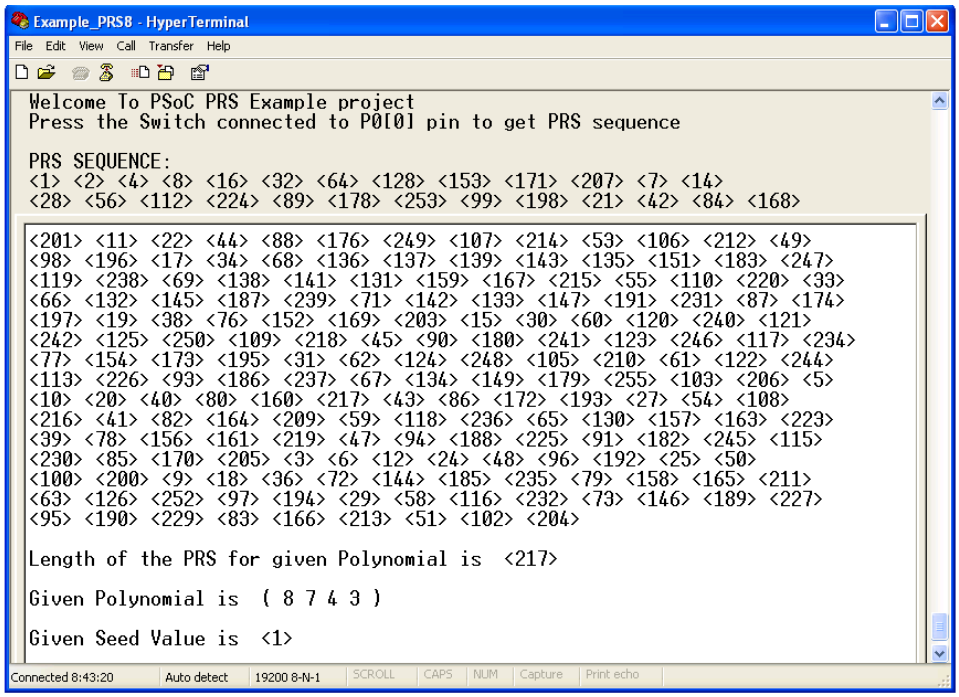
Figure 3. PR Sequence for Polynomial (8 6 5 4) and Seed Equal to 1



The same sequence is repeated when the switch is pressed again.

If the modular polynomial is changed to 8,7,4,3 (0xCC); the following PR sequence with length 217 is generated.

Figure 4. PR Sequence for Polynomial (8 7 4 3) and Seed Equal to 1



Testing the Project

To test the project, HyperTerminal (or any other terminal program) may be used. The following explains how HyperTerminal can be configured in Windows.

1. Connect the CY3210 board to the PC serial port using a serial port cable.
2. Start HyperTerminal using **Start → All Programs → Accessories → Communication → HyperTerminal**
3. Enter a name for the connection, such as **Example_PRS8** and select **OK**.

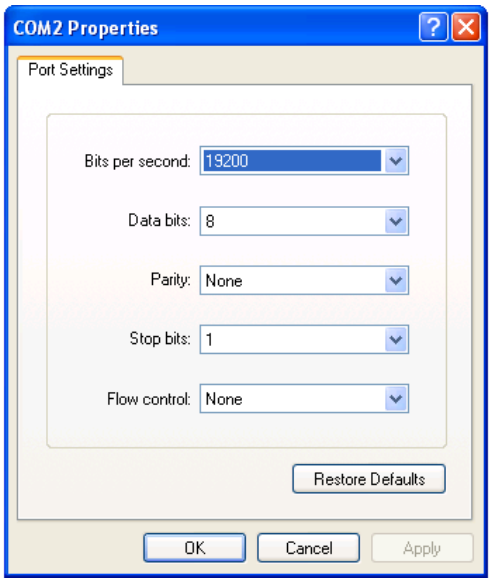
Figure 5. Give a Name to the Connection

4. In **Connect To** option, select the desired serial port (for example, COM2) from the **Connect using** list and click **OK**.

Figure 6. Select the Serial Port

5. In **COM2 Properties**, configure the following parameters:
   - Bits per second = 19200
   - Data bits = 8
   - Parity = None
   - Stop Bits = 1
   - Flow Control = None
   - Click **OK**

Figure 7. COM2 Properties Window



6. At this point, the HyperTerminal connects to COM2 and is ready to be used with the PRS8 code example.

Turn on the switch connected to P0[0] and get the random numbers printed on HyperTerminal for every 10 ms. The length of the PR sequence for the given modular polynomial is calculated by PSoC and is displayed in HyperTerminal. After printing the entire sequence, if the switch is again turned on, the same sequence is repeated on HyperTerminal.

Upgrade Information

The `ljmp` instructions to the C ISRs for GPIO and Counter are placed inside the user code markers that are inside the *PSoCGPIOINT.asm* and *CounterINT.asm* files. These changes are preserved when the project is generated and built. If the source files for the *PSoCGPIOINT.asm* and *CounterINT.asm* files change in the future releases of PSoC Designer™, these instructions may be overwritten. After PSoC Designer upgrade, if the project is not working, verify the following:

- Open *PSoCGPIOINT.asm* file and check if there is an `ljmp _GPIO_ISR` instruction present in the user code area in the `PSoC_GPIO_ISR` function. If not, add this line of code within the user code markers.
- Open the *CounterINT.asm* file and check if there is an `ljmp _CounterISR` instruction present inside the user code marker of the `_Counter_ISR` function. If not, add this line within the user code markers.

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