CE221295 – PSoC 6 MCU Cryptography: True Random Number Generation

Objective

This code example demonstrates generating a One-Time Password (OTP) using the True Random Number generation feature of PSoC® 6 MCU cryptography block.

Overview

This example demonstrates generating a One-Time-Password (OTP) of eight characters in length. Using the True Random Number generation feature of PSoC 6 MCU crypto block, a random number corresponding to each character of the OTP is generated. The generated random number is such that it corresponds to alpha-numeric and special characters of the ASCII code. The generated OTP is then displayed on a UART terminal emulator.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure that the kit is configured correctly.

Software Setup

This example uses Tera Term as the UART terminal for displaying the generated OTP. Set the UART configuration settings as the same as that used by the UART SCB on PSoC 6 MCU.

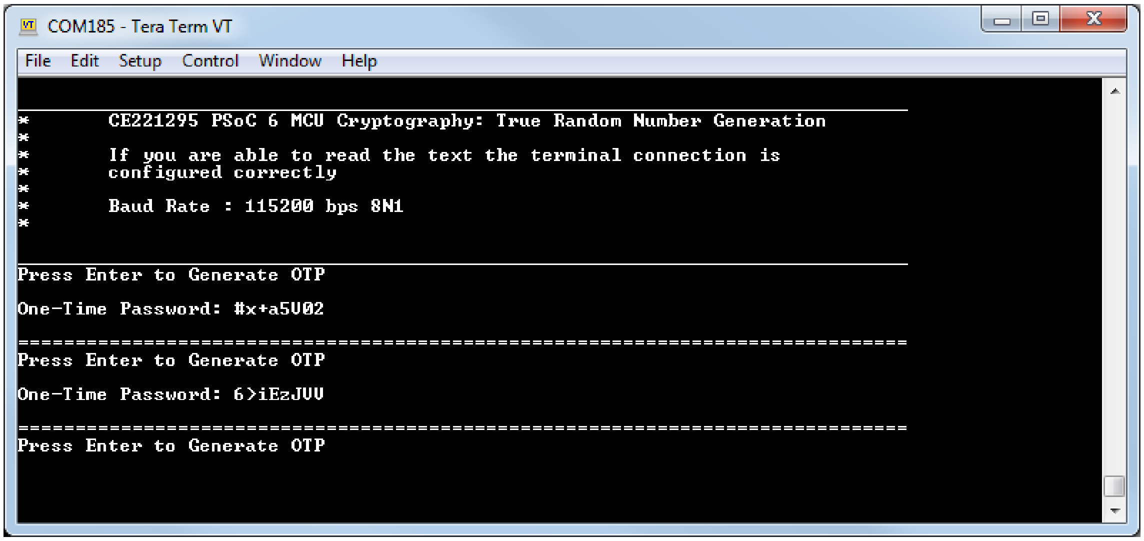
Operation

1. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
2. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.

   Note: If during the build process, if the PSoC Creator prompts you to replace *stdio_user.h* file, **DO NOT** replace the file.
3. Open Tera Term and connect to the "KitProg2 USB-UART" bridge COM port. Set the baud rate as 115200 bps.
4. Press the **Enter** key to generate an OTP. The generated OTP will be displayed on the UART terminal. Note that you must press the **Enter** key every time when you need to generate an OTP.

Figure 1 shows a sample output as displayed on Tera Term UART Terminal.

Figure 1. Sample Output as Displayed on Tera Term



Design and Implementation

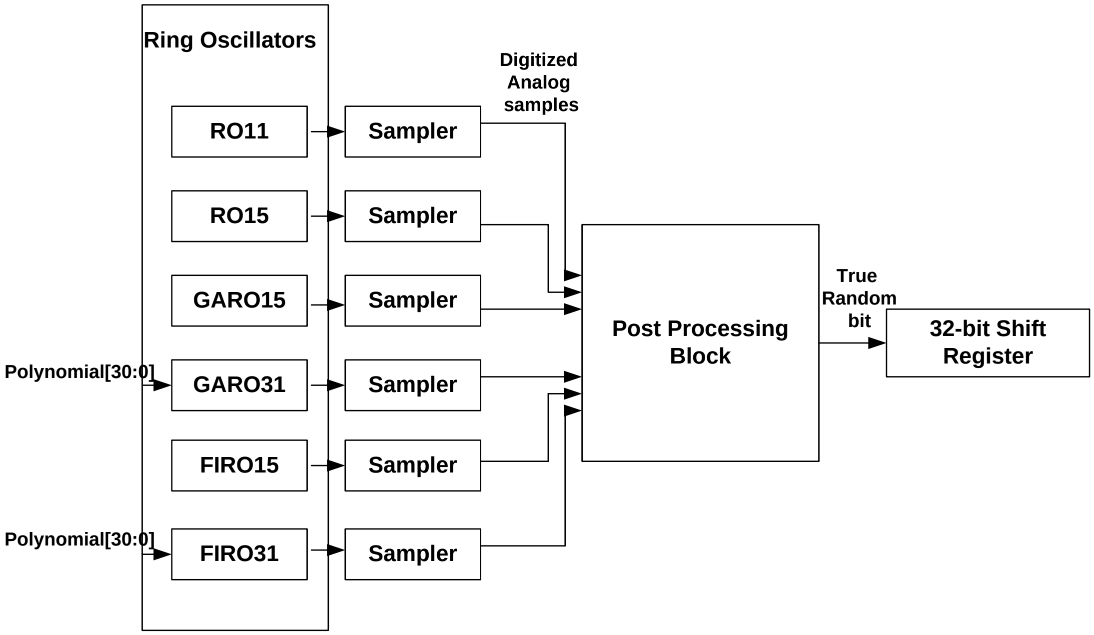
Random number generation is the generation of a sequence of numbers or symbols that cannot be predicted based on the previous knowledge of the generated sequence. Random number generators have applications in cryptography, statistical sampling, gambling and other areas where producing an unpredictable result is desirable.

A true random number is generated using a hardware random number generator that generates random numbers from a physical process. The true random number generator (TRNG) in PSoC 6 MCU generates true random numbers of programmable bit size ranging from 0–32 bits. The TRNG relies on up to six ring oscillators to provide physical noise sources namely:

- Two fixed ring oscillators consisting of 11 and 15 inverters (RO11 and RO15).
- A fixed Galois-based ring oscillator (GARO15) and a fixed Fibonacci-based ring oscillator (FIRO15) each consisting of 15 inverters.
- A flexible Galois-based (GARO31) and a flexible Fibonacci-based oscillator (FIRO31) consisting of 31 inverters with a programmable polynomial of up to order 31.

A ring oscillator consists of a series of inverters connected in a feedback loop to form a ring. Due to (temperature) sensitivity of the inverter delays, jitter is introduced on a ring's oscillating signal. The jittered oscillating signal is sampled to produce a digitized analog signal (DAS). This is done for all multiple ring oscillators. To increase entropy and to reduce bias in DAS bits, the DAS bits are further post-processed. Post-processing produces bit samples that are considered true random bit samples. The true random bit samples are shifted into a register to provide random values of up to 32 bits. Figure 2 shows an overview of how generation of true random generation is implemented in PSoC 6 MCU.

Figure 2. True Random Number Generation in PSoC 6 MCU



Cryptography in PSoC 6 MCU is based on a Client-Server model. The firmware initializes and starts the Crypto server. The server runs only on the CM0+ core, and works with the crypto hardware. The Crypto client can run on either core. In this example, the client runs on the CM4 core. The firmware initializes and starts the client. The firmware then provides the configuration data required for generation of true random number and requests the crypto server to run the cryptographic operation.

In this example, an OTP of eight characters in length is generated. Using the true random number generator, a random number is generated corresponding to each character of the OTP. The generated random number is such that it corresponds to alpha-numeric and special characters of the ASCII code. The generated OTP is then displayed on a UART terminal emulator. Each time, the firmware waits for the user to press the “Enter” key to generate a new OTP.

Figure 3. Firmware Flowchart

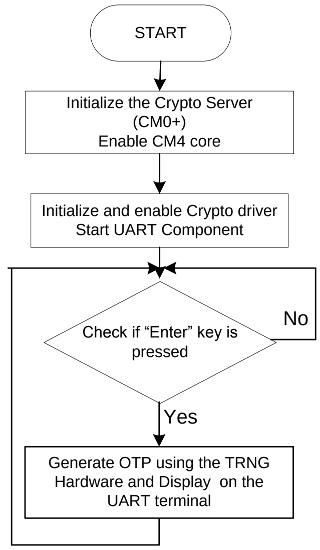
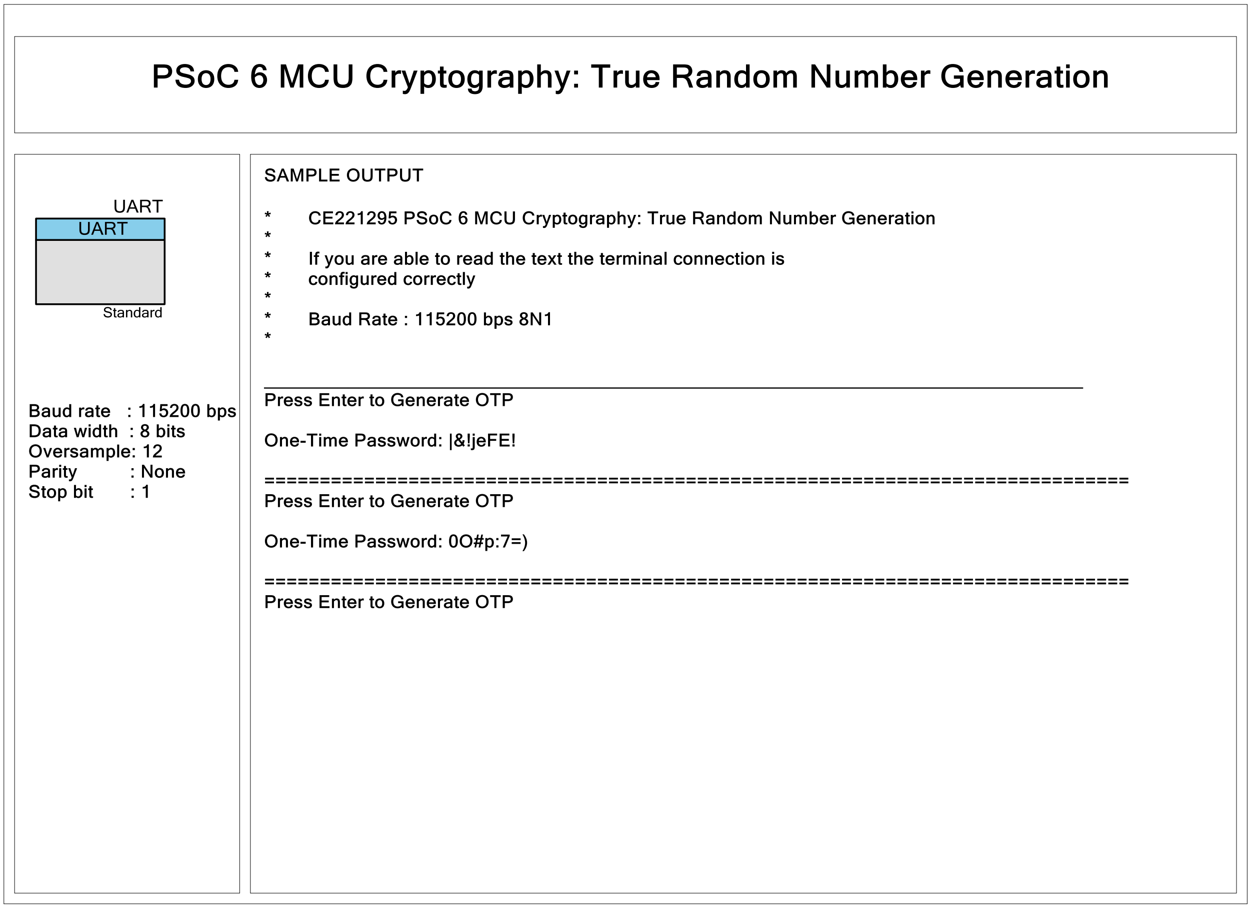


Figure 4. PSoC Creator Project Schematic



Components and Settings

Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
UART	UART	Facilitates printing on to a UART terminal	Default settings only

For information on the hardware resources used by a Component, see the Component datasheet.

In order to use the Crypto block of PSoC 6 MCU in your design, the Crypto driver must be enabled. To enable the drivers, check the crypto option under **Project** > **Build Settings** > **Peripheral Driver Library** as shown in Figure 5.

Figure 5. Enabling Crypto PDL Drivers

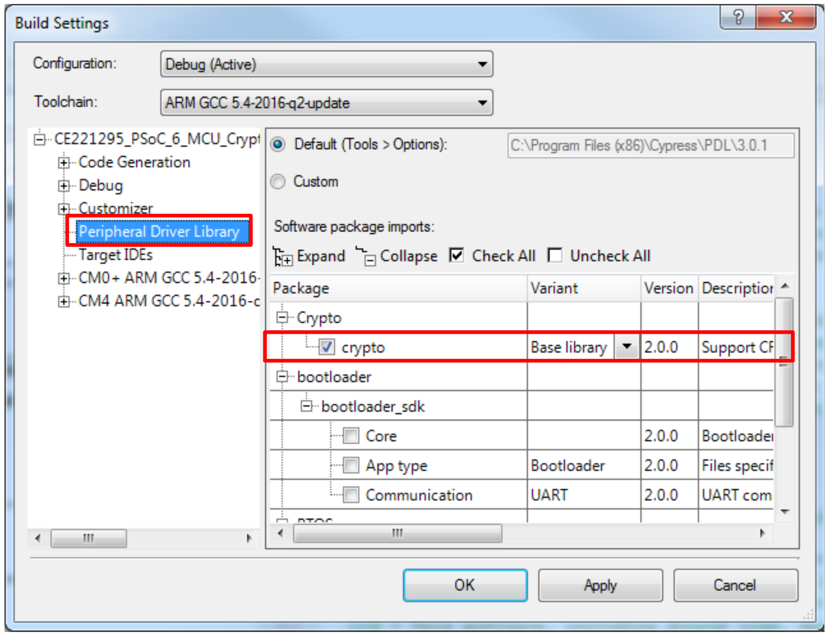
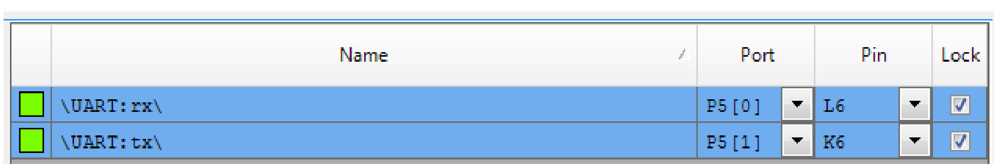


Figure 6 shows the pin assignment for the project done through the **Pins** tab in the **Design Wide Resources** window. This example uses the Kitprog2 USB-UART bridge to communicate with UART terminal emulator running on your PC. CY8CKIT-062-BLE Pioneer kit uses the pin **P5[0]** as UART Rx pin and pin **P5[1]** as UART Tx pin.

Figure 6. Device Pin Assignments



Reusing This Example

This example is designed for the CY8CKIT-062-BLE Pioneer Kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using Device Selector and update the pin assignments in the Design-Wide Resources Pins settings as needed.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
PSoC Creator Component Datasheets	
UART	Supports standard UART interface
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE221295 – PSoC 6 MCU Cryptography: True Random Number Generation

Document Number: 002-21295

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6006857	VKVK	12/27/2017	New code example

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709