CE220511 – PSoC 6 MCU Cryptography: SHA Demonstration

Objective

This code example demonstrates generating a unique hash value or message digest for an arbitrary message using Secure Hash Algorithm (SHA) in PSoC® 6 MCU.

Requirements

Tool: ModusToolbox™ IDE 1.1

Programming Language: C

Associated Parts: All PSoC 6 MCU parts

Related Hardware: PSoC 6 BLE Pioneer Kit, PSoC 6 WiFi-BT Pioneer Kit, PSoC 6 WiFi-Prototyping Kit

Overview

This code example shows how to generate a 32-byte hash value or message digest for an arbitrary user input message with the SHA2 algorithm using the Cryptographic hardware block in PSoC 6 MCU. The example further shows that any change in the message results in a unique hash value for the message. The hash value generated for the message is displayed on a UART terminal emulator.

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 message digest. If you don't have one, install one; this example uses Tera Term.

Operation

1. Connect the kit to your PC using the provided USB cable.
2. Import the code example into a new workspace. See KBA225201.
3. Program the PSoC 6 MCU device. In the project explorer, select the **mainapp** project. In the Quick Panel, scroll to the **Launches** section and click the **Program (KitProg3)** configuration. Program configurations also build the code.
4. Open Tera Term and connect to the USB-UART bridge COM port. Set the connection to 115200 bps, 8N1.
5. Press the reset button on the kit and enter the message for which hash value or the message digest should be generated. The generated hash value is printed on the UART terminal. Note that for every input message, the SHA operation generates a unique hash value.

   For example, the message "The quick brown fox jumps over the lazy dog", which uses every letter in the English alphabets, has a message digest value completely different from the message digest value for the message "The quick brown fox jumps over the lazy Dog" even though they have a very small variation ('D' instead of 'd').

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

Figure 1. Sample Output as Displayed on Tera Term

```
________________________________________________________________________________
*                CE220511 PSoC 6 MCU Cryptography: SHA Demonstration
*
*        This code example shows how to generate a 32-byte hash value for an
*        arbitrary user input message using the SHA2 algorithmin PSoC 6 MCU
*        UART Terminal Settings: Baud Rate - 115200 bps, 8N1
*
________________________________________________________________________________

Enter the message:
The quick brown fox jumps over the lazy dog

Hash Value for the message:

0xD7 0xA8 0xFB 0xB3 0x07 0xD7 0x80 0x94 0x69 0xCA 0x9A 0xBC 0xB0 0x08 0x2E 0x4F
0x8D 0x56 0x51 0xE4 0x6D 0x3C 0xDB 0x76 0x2D 0x02 0xD0 0xBF 0x37 0xC9 0xE5 0x92

________________________________________________________________________________

Enter the message:
The quick brown fox jumps over the lazy Dog

Hash Value for the message:

0x73 0x25 0xE8 0x31 0x81 0x6C 0x7C 0xE2 0x6F 0xF5 0xD8 0x82 0xDA 0x79 0x18 0xEA
0x3B 0x24 0xCE 0xE1 0x51 0x42 0x8E 0x63 0x74 0x75 0x83 0xF2 0xBE 0xAD 0x53 0x8D

________________________________________________________________________________

Enter the message:
```

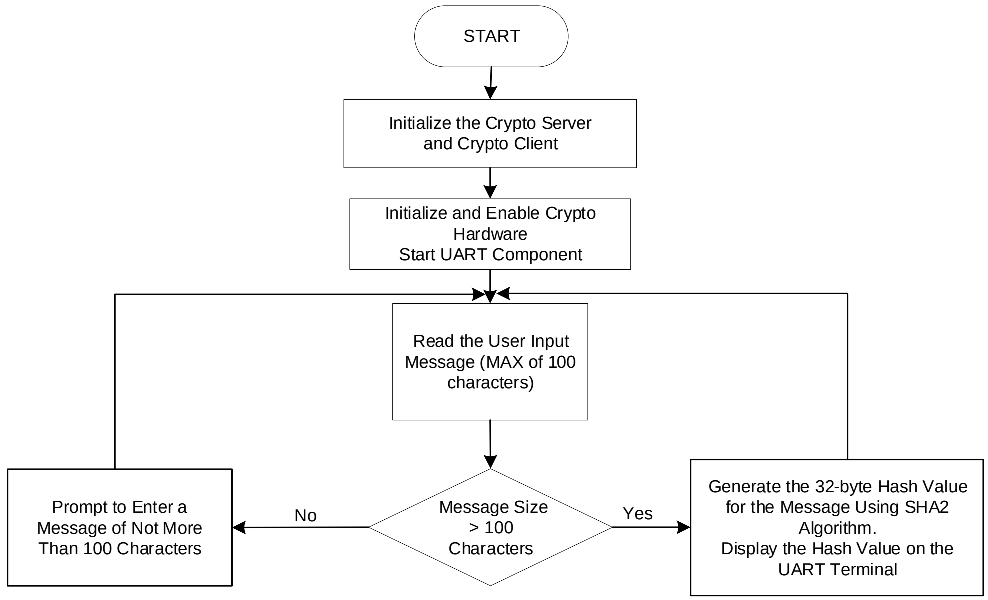
Design and Implementation

Secure Hash Algorithm is a function that takes a message of arbitrary length and reduces it to a fixed length residue or message digest after performing a series of mathematically defined operations that practically guarantee that any change in the message will change the hash value. A hash value is used for message authentication by transmitting a message with a hash value appended to it and recalculating the message hash value using the same algorithm at the recipient's end. If the hashes differ, it indicates that the message has been corrupted.

Cryptographic operation in this example is based on a Client-Server model. In this example, both the Crypto Server and the Crypto Client runs on the CM4 CPU. The firmware initializes and starts the Crypto Server and the Crypto Client. The firmware then provides the configuration data required for the SHA operation and requests the Crypto Server to run the cryptographic operation. Secure Hash Algorithm is directly implemented in hardware using the Crypto block of PSoC 6 MCU.

In this example, the user input message is read from the UART terminal and a 32-byte long hash value is generated using the SHA2 algorithm. For any arbitrary message, a 32-byte long hash value is generated. The 32-byte hash value for the user input message is then displayed on the UART terminal emulator. Note that in this example, the maximum message size is restricted to 100 characters. If you need to increase the message size change the macro `MAX_MESSAGE_SIZE` in the *main.c* to the message size that you require.

Figure 2. Firmware Flow



Resources and Settings

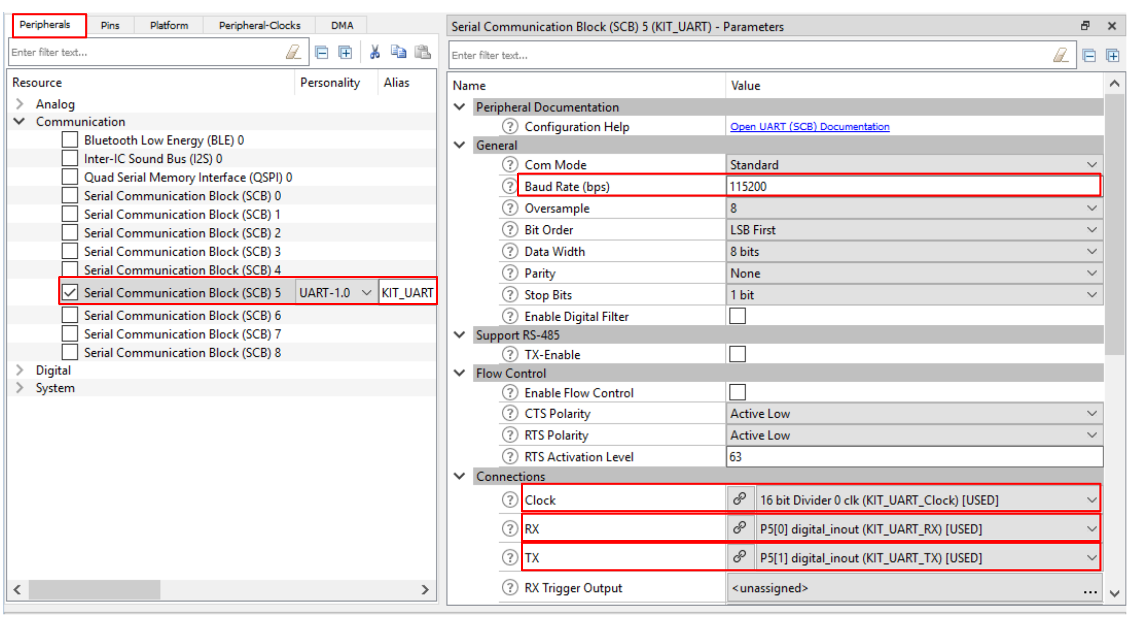
Table 1 lists the ModusToolbox resources used in this example, and how they are used in the design. For pin usage and configuration, open the **Pins** tab of the *design.modus* file.

Table 1. ModusToolbox Resources

Resource	Alias	Purpose
Serial Communication Block (SCB) – UART mode	KIT_UART	Send to and receive data from the UART Terminal.

Figure 3 highlights the non-default settings for the Serial Communication Block (SCB) in UART mode.

Figure 3. SCB (UART mode) Configuration



Reusing This Example

This example is designed for the supported kits. To port the design to a different PSoC 6 MCU device, right-click the application project and choose **Change Device**. If changing to a different kit, you may need to reassign pins.

Table 2. Device and Pin Mapping Table Across PSoC 6 MCU Kits

Kit Name	Device Used	KIT_UART_RX	KIT_UART_TX
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	P5[0]	P5[1]
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	P5[0]	P5[1]
CY8CPROTO-062-4343W	CY8C624ABZI-D44	P5[0]	P5[1]

In some cases, a resource used by a code example (for example, an IP block) is not supported on another device. In that case, the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on what a particular device supports.

Related Documents

For a comprehensive list of PSoC 6 MCU resources, see KBA223067 in the Cypress community.

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices.
AN215656 – PSoC 6 MCU: Dual-CPU System Design	Describes the dual-CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-CPU design
Code Examples	
Visit the Cypress GitHub site for a comprehensive collection of code examples using ModusToolbox IDE	
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kits	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit	
Tool Documentation	
ModusToolbox IDE	The Cypress IDE for IoT designers

Cypress Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right device, and quickly and effectively integrate the device into your design.

For PSoC 6 MCU devices, see KBA223067 in the Cypress community for a comprehensive list of PSoC 6 MCU resources.

Document History

Document Title: CE220511 – PSoC 6 MCU Cryptography: SHA Demonstration

Document Number: 002-25971

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6482080	VKVK	02/21/2019	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 | Code Examples | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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