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: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.1

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

Overview

This code example shows how to generate a 20-byte hash value or message digest for an arbitrary user input message using 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. See 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. Plug the CY8CKIT-062 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.
3. Open your terminal software and select the KitProg COM port. Set the other serial port parameters as follows:
   a. Baud rate: 115200 bps
   b. Data: 8 bit
   c. Parity: None
   d. Stop: 1 bit
   e. Flow control: None
4. 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 Tera Term UAT terminal.

Figure 1. Sample Output Hash Value Generated Using SHA

```
*               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:
```

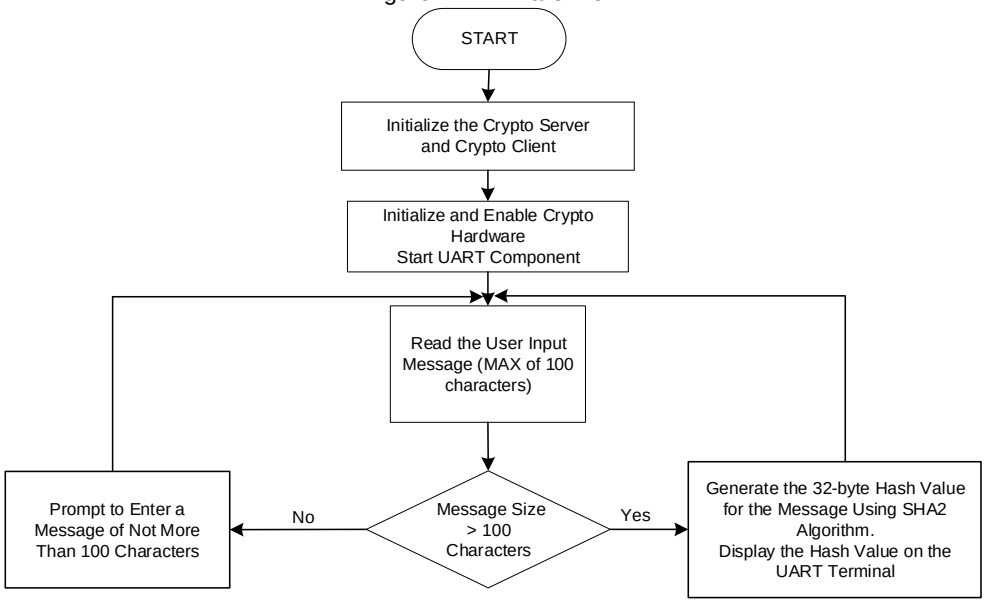
Design and Implementation

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



Components and Settings

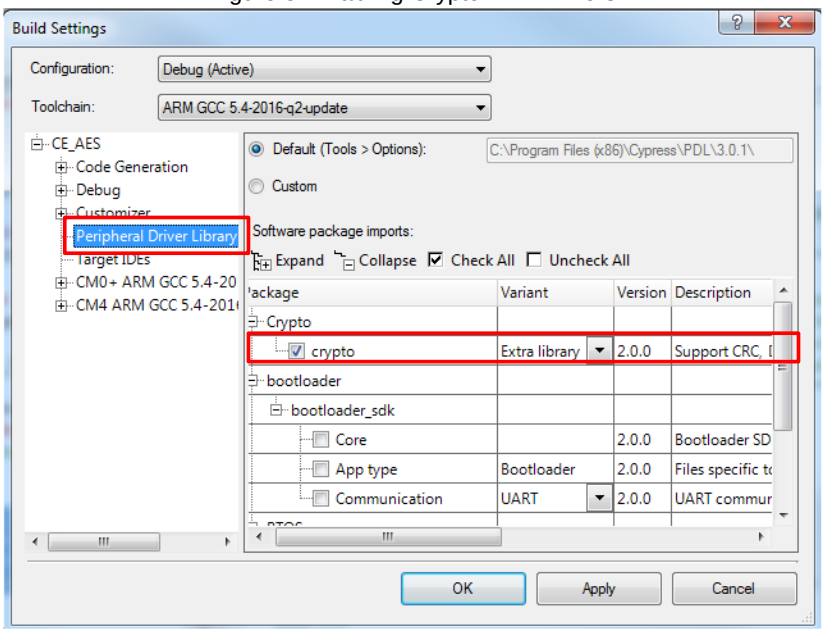
Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
UART	KIT_UART	Send to and receive data from the UART Terminal	Baud Rate set to 115200 bps

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 3.

Figure 3. Enabling Crypto PDL Drivers



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.

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]

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 and how to build your first PSoC Creator project
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
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
PSoC Creator Component Datasheets	
UART	Supports standard UART interface
Device Documentation	
PSoC 6 MCU Datasheets	PSoC 6 Technical Reference Manuals
Development Kit Documentation	
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	
CY8CPROTO-063 BLE PSoC 6 BLE Prototyping Kit	
Tool Documentation	
PSoC Creator	Look in the downloads tab for Quick Start and User Guides
Peripheral Driver Library (PDL)	Get the latest version for use with PSoC Creator. Look in the *<PDL install folder>/doc* for the User Guide and the API Reference

Document History

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

Document Number: 002-20511

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5846644	VKVK	08/07/2017	New code example
*A	6002370	VKVK	12/22/2017	Updated for PSoC Creator 4.2
*B	6514278	VKVK	03/19/2019	Updated for PDL 3.1.0

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

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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

© Cypress Semiconductor Corporation, 2017-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.
Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.