CE220465 – PSoC 6 MCU Cryptography: AES Demonstration

Objective

This code example demonstrates encryption and decryption of data using the Advanced Encryption Scheme (AES) algorithm in PSoC® 6 MCU.

Overview

This code example encrypts and decrypts user input data using the AES algorithm using a 128-bit long key. The encrypted and decrypted data are displayed on a UART terminal emulator.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (ARM® GCC 5.4)

Associated Parts: PSoC 6 MCU

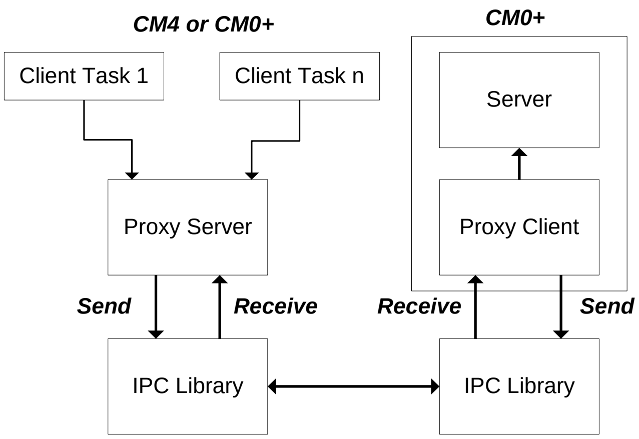
Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Design

AES is a symmetric block cipher data encryption algorithm. The AES operation works on 128-bit block size. AES is a symmetric algorithm which means that it uses same key for encryption and decryption of data. The AES algorithm uses keys of 128 bits, 192 bits, or 256 bits of length.

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. Access to the server is through the Inter-Process Communication (IPC) driver. Figure 1 shows the client-server model for the Crypto block in PSoC 6 MCU.

Figure 1 Crypto Client-Server Architecture in PSoC 6 MCU



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 AES encryption technique and requests the crypto server to run the cryptographic operation.

The user input message is read from the UART terminal and encrypted using the AES algorithm with a key length of 128 bits. The 128-bit-long encrypted data is displayed on the UART terminal. Then, the user can view the decrypted message on the UART terminal and verify that the decryption operation produces the same original encrypted message. Figure 2 shows the firmware flowchart.

Figure 2. Firmware Flow

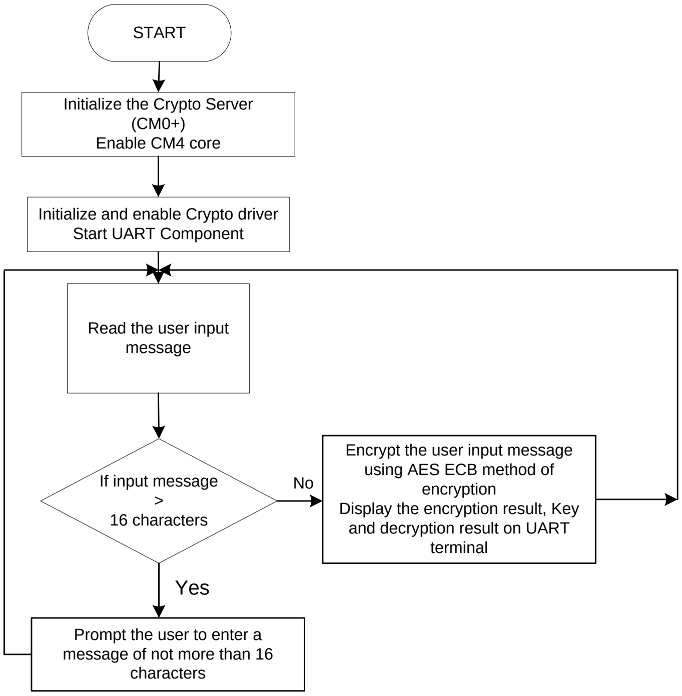
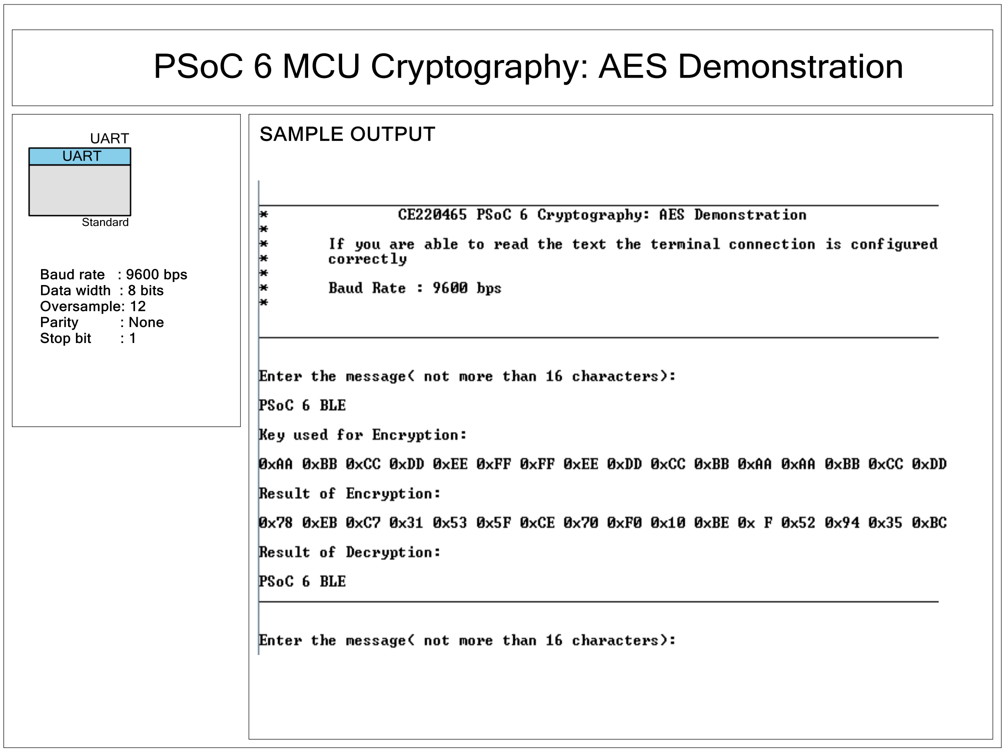

START
Initialize the Crypto Server (CM0+)
Enable CM4 core
Initialize and enable Crypto driver
Start UART Component
Read the user input message
If input message > 16 characters
No
Encrypt the user input message using AES ECB method of encryption
Display the encryption result, Key and decryption result on UART terminal
Yes
Prompt the user to enter a message of not more than 16 characters

Figure 3 shows the PSoC Creator project schematic.

Figure 3. PSoC Creator Project Schematic



PSoC 6 MCU Cryptography: AES Demonstration

Baud rate : 9600 bps
Data width : 8 bits
Oversample: 12
Parity : None
Stop bit : 1

SAMPLE OUTPUT

```
_____________________________________________________________________________
*               CE220465 PSoC 6 Cryptography: AES Demonstration
*
*       If you are able to read the text the terminal connection is configured
*       correctly
*
*       Baud Rate : 9600 bps
*
_____________________________________________________________________________

Enter the message( not more than 16 characters):

PSoC 6 BLE

Key used for Encryption:

0xAA 0xBB 0xCC 0xDD 0xEE 0xFF 0xFF 0xEE 0xDD 0xCC 0xBB 0xAA 0xAA 0xBB 0xCC 0xDD

Result of Encryption:

0x78 0xEB 0xC7 0x31 0x53 0x5F 0xCE 0x70 0xF0 0x10 0xBE 0x F 0x52 0x94 0x35 0xBC

Result of Decryption:

PSoC 6 BLE
_____________________________________________________________________________

Enter the message( not more than 16 characters):
```

Hardware Setup

No special hardware setup is needed for CY8CKIT-062-BLE. Connect the kit's USB port to your computer's USB port. The KitProg2 system on the kit acts as both a programmer for direct programming, and as a USB-UART bridge for displaying the encryption and decryption results on a UART terminal.

Software Setup

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

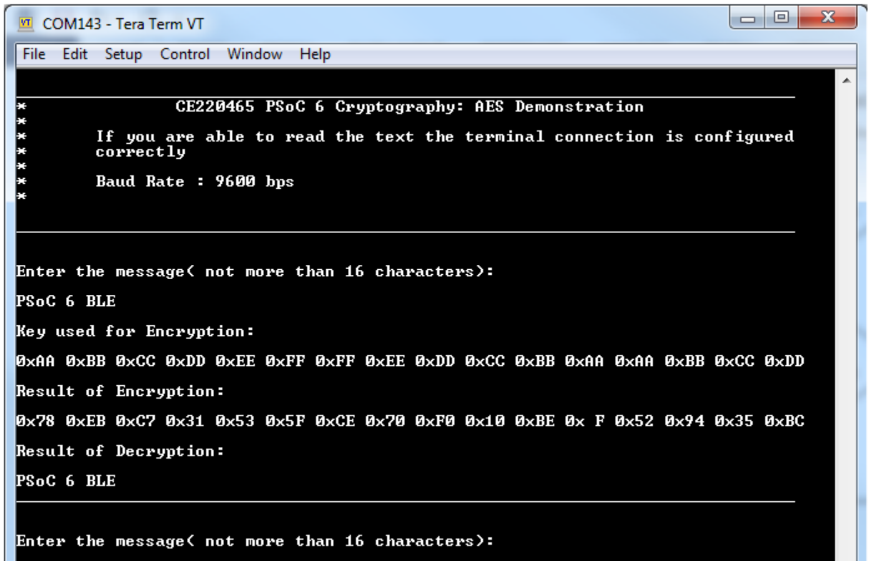
Operation

1. Plug the CY8CKIT-062 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 Tera Term and connect to the "KitProg2 USB-UART" bridge COM port. Set the baud rate as 9600 bps and enable the local echo option under **Setup** > **Terminal**.
4. Press the reset button on the kit and enter the message (not more than 16 characters) to be encrypted.

Note that the crypto block in PSoC 6 MCU accepts 128 bits (16 bytes) as input data. If you need to enter a message of more than 16 characters, modify the firmware to process the data in blocks of 16 characters.

Figure 4 shows a sample output as displayed on Tera Term UART terminal.

Figure 4. Sample Output showing AES Encryption



```
*               CE220465 PSoC 6 Cryptography: AES Demonstration
*
*       If you are able to read the text the terminal connection is configured
*       correctly
*
*       Baud Rate : 9600 bps
*


Enter the message( not more than 16 characters):

PSoC 6 BLE

Key used for Encryption:

0xAA 0xBB 0xCC 0xDD 0xEE 0xFF 0xFF 0xEE 0xDD 0xCC 0xBB 0xAA 0xAA 0xBB 0xCC 0xDD

Result of Encryption:

0x78 0xEB 0xC7 0x31 0x53 0x5F 0xCE 0x70 0xF0 0x10 0xBE 0x F 0x52 0x94 0x35 0xBC

Result of Decryption:

PSoC 6 BLE


Enter the message( not more than 16 characters):
```

The sections that follow discuss the Components, parameter settings, and resources used to make the example.

Components

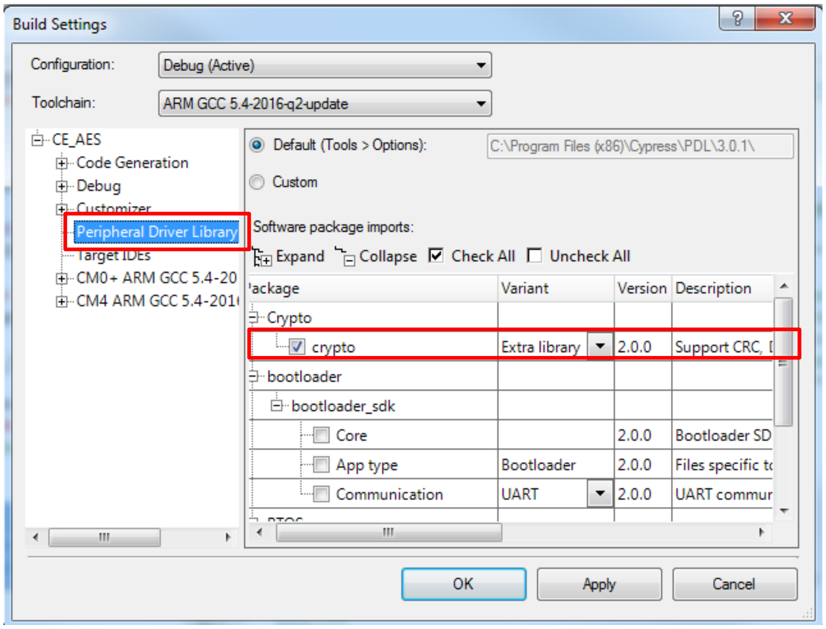
Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Table 1. PSoC Creator Components

Component	Instance Name	Hardware Resources	Parameter Settings
UART	UART	1 SCB	Baud Rate set to 9600 bps

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



Build Settings

Configuration: Debug (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

● Default (Tools > Options): C:\Program Files (x86)\Cypress\PDL\3.0.1\

○ Custom

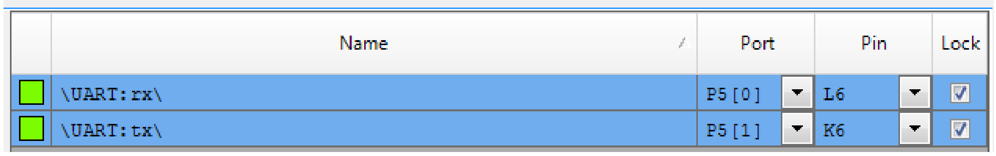
Software package imports:

'ackage	Variant	Version	Description
Crypto			
☑ crypto	Extra library	2.0.0	Support CRC, [
bootloader			
bootloader_sdk			
□ Core		2.0.0	Bootloader SD
□ App type	Bootloader	2.0.0	Files specific t
□ Communication	UART	2.0.0	UART commur

Design-Wide Resources

Figure 6 shows the pin assignments for the CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit required for the UART Component.

Figure 6. Device Pin Assignments



	Name	Port	Pin	Lock
🟩	\UART:rx\	P5[0]	L6	☑
🟩	\UART:tx\	P5[1]	K6	☑

Related Documents

Application Notes	
AN210781 - Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 MCU with BLE, and how to build this code example.
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 (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE220465 - PSoC 6 MCU Cryptography: AES Demonstration

Document Number: 002-20465

Revision	ECN	Orig. of Change	Submission Date	Description of Change
*A	5887785	VKVK	9/18/2017	Initial public release version

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

Forums | WICED IOT Forums | 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

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. 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. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, 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 or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.