

---

---

---

# SimpleLink™ CC1200 AES Operation

---

---

---

Martin Brinkmann

## ABSTRACT

This application report shows how to use the Advanced Encryption Standard (AES) capability of the SimpleLink CC1200 low-power high performance transceiver. It shows how to do stand-alone CBS block operations and how to do in-line CTR mode security operations on TX/RX FIFO content. The application report assumes that the reader is familiar with the basic concepts of AES and the different commonly used modes such as Counter (CTR) and Cipher Block Chaining (CBC).

---

## Contents

1	Introduction .....	2
2	Background .....	2
3	Address Space Used by the AES Module .....	2
4	AES Block Operation .....	3
5	AES FIFO Operations .....	4
6	References .....	9

## List of Figures

1	CC1200 Memory Space .....	2
---	---------------------------	---

## List of Tables

1	Acronyms Used in This Document .....	2
2	SPI Access Type for AES Operation .....	3
3	AES Commands .....	4
4	AES TX FIFO Parameters .....	5
5	AES RX FIFO Parameters .....	5

## 1 Introduction

This application report describes the key elements of the SimpleLink CC1200 transceiver AES encryption module. The AES encryption module is typically used to encrypt RF communication, to provide a secure RF link. The main objective of this application report is to explain how to utilize the CC1200 AES encryption module to do AES counter mode (CTR) encryption and decryption.

### 1.1 Acronyms

**Table 1. Acronyms Used in This Document**

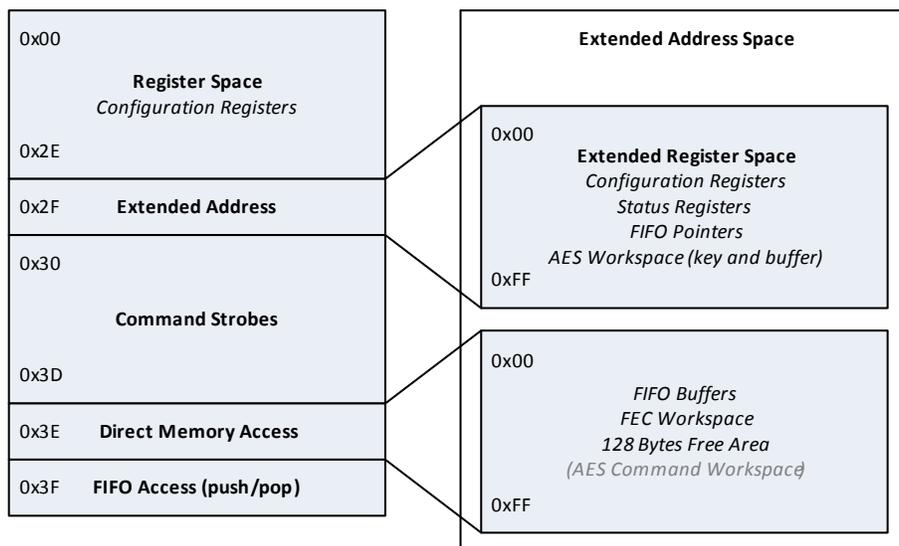
Acronym	Description
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CTR	Counter
FIFO	First In First Out
IV	Initialization Vector
RF	Radio Frequency
RX	Receive
SPI	Serial Peripheral Interface
TX	Transmit

## 2 Background

The CC1200 contains an AES module that can be used to do stand-alone AES block operations or to do security operations on the FIFO content. The AES module uses cipher block chaining (CBC) for block operations. When using AES operations on the TX FIFO and RX FIFO content, the AES module uses CBC block operation to do CTR encryption and decryption.

## 3 Address Space Used by the AES Module

The CC1200 uses two different parts of the extended memory space to do AES operations; the two memory spaces are the AES workspace and the AES command workspace. Different serial peripheral interface (SPI) access needs to be used to get access to the two memory spaces. [Figure 1](#) shows the SPI memory map for the different SPI access types. [Section 3.1](#) and [Section 3.2](#) provide more details about the different workspaces and how they are used for AES operations.



**Figure 1. CC1200 Memory Space**

### 3.1 AES Workspace

The AES workspace is in the extended memory section. It holds the 128-bit AES key and the 128 data buffer for block operations. The extended register space needs to be accessed to use the AES workspace. In the extended register space the AES key is located from address 0xE0 to 0xEF and the data buffer from address 0xF0 to 0xFF. For more details, see the *CC120X Low-Power High Performance Sub-1 GHz RF Transceivers User's Guide* ([SWRU346](#)).

### 3.2 AES Command Workspace

The AES command workspace is used for AES parameters when doing AES operations on FIFO content. It holds the Initialization Vector (IV)/nonce along with length information and pointers to the FIFO content to be encrypted or decrypted. The parameters used are described in [Section 5.2](#). The AES command workspace can be accessed through the direct memory access command described in [Table 2](#).

**Table 2. SPI Access Type for AES Operation**

Access Type	Command/Address Byte	Description
Direct Access to 128 Bytes Free Area (AES Command Workspace)	Command: R/W; B 1 1 1 1 1 0  Address: A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>  SPI_DIRECT_ACCESS_CFG in SERIAL_STATUS must be 1  0x80 ≤ A <sub>7</sub> - 0 ≤ 0xFF	This access mode starts with a specific command (0x3E) which makes it possible to access 128 bytes of free memory.  The first byte following this command is interpreted as the address. The next byte is read/written to this address. If burst is enabled, consecutive bytes will be read/written by incrementing the address.

## 4 AES Block Operation

The CC1200 AES module can be used to do CBC encryption on 128-bit data blocks. The MCU needs to write the data block to the CC1200 over SPI in order to do block operations. After the encryption is completed the MCU needs to read the encrypted data back. [Section 4.1](#) describes the steps needed to do block operation on 128-bit data blocks and [Example 1](#) shows an example on how this can be done in the software. Note that the CC1200 is only capable of CBC encryption. For both encryption and decryption CTR mode must be used.

### 4.1 AES Block Operation Procedure

The following steps must be done to encrypt a 128-bit data block using CBC:

1. Write the 128 bit long AES key to the key location in the AES workspace.
2. XOR plain text with initialization vector or cipher text from previous block.
3. Write the 128 bit long XOR'ed data block to the buffer location in the AES workspace.
4. Execute encryption by setting the AES\_AES\_RUN bit to 1. The bit will be set low by HW when encryption is finished.
5. Read the encrypted data block.

### Example 1. AES Block Operation Procedure

```

#define AES_BLOCK_SIZE      16
static uint8 writeByte;
static uint8 plainData[AES_BLOCK_SIZE];
static uint8 aesKey[AES_BLOCK_SIZE];
static uint8 dataBlock[AES_BLOCK_SIZE];
static uint8 initializationVector[AES_BLOCK_SIZE];
static uint8 cipherBlock[AES_BLOCK_SIZE];

// 1) Write 128 bit AES key into key memory input
cc120xSpiWriteReg(CC120X_AES_KEY, aesKey, AES_BLOCK_SIZE);

// 2) XOR initialization vector with plain text data
for (uint8 i = 0; i < AES_BLOCK_SIZE; i++)
{
    dataBlock[i] = (plainData[i] ^ initializationVector[i]);
}

// 3) Write 128 bit data block into plain data memory input
cc120xSpiWriteReg(CC120X_AES_BUFFER, dataBlock, AES_BLOCK_SIZE);

// 4) Execute AES_RUN
writeByte = 0x01;
cc120xSpiWriteReg(CC120X_AES, &writeByte, 1);

// Wait for AES operation to finish
while((writeByte & 0x01) == 0x01 ){
    cc120xSpiReadReg(CC120X_AES, &writeByte, 1);
}

// 5) Read cipher block from AES buffer
cc120xSpiReadReg(CC120X_AES_BUFFER, cipherBlock, AES_BLOCK_SIZE);
    
```

## 5 AES FIFO Operations

The AES module can also be used to do AES operations directly on data placed in the TX and RX FIFO. The AES commands used on the FIFO content will encrypt and decrypt using AES CTR. The following section describes the different AES commands and AES parameters needed to do AES FIFO operations.

### 5.1 AES Commands

The CC1200 has two AES high level commands that can be used to do AES operation on data placed in the TX and RX FIFO. [Table 3](#) shows the different AES commands. The two commands are set in the AES\_COMMANDS bit field in the MARC\_SPARE register and will be executed by an SIDLE command strobe.

**Table 3. AES Commands**

MARC_SPARE.AES_COMMANDS	Command	Description
0x09	AES_TXFIFO	AES CTR on the TX FIFO content
0x0A	AES_RXFIFO	AES CTR on the RX FIFO content

### 5.2 AES Parameters

Two parameters are needed for the AES module to perform FIFO operations, a pointer to the first byte in the FIFO and the number of bytes to be encrypted or decrypted. [Section 5.2.1](#) and [Section 5.2.2](#) describe the two parameters needed for each operation and where in the memory these parameters must be written to in the AES workspace. This can be done using the direct memory access shown in [Table 2](#).

### 5.2.1 AES TX FIFO

The `AES_TXFIFO` command is used to do CTR encryption on the TX FIFO content. Parameters needed to be initialized are shown in [Table 4](#). The initialization vector/nonce is located in memory location 0x80 in the AES command workspace.

**Table 4. AES TX FIFO Parameters**

Address	Parameter
0xF0	Pointer to first entry in the TXFIFO to encrypt (the packet must always be written to a flushed TXFIFO meaning that the first byte in the packet is at location 0x00)
0xF2	Number of bytes in the TXFIFO that should be encrypted

### 5.2.2 AES RX FIFO

The `AES_RXFIFO` command is used to do CTR decryption on the RX FIFO content. Parameters needed to be initialized are shown in [Table 5](#). The initialization vector/nonce is located in memory location 0x80 in the AES command workspace.

**Table 5. AES RX FIFO Parameters**

Address	Parameter
0xF0	Pointer to first entry in the RXFIFO to decrypt (the RXFIFO must be flushed before the packet is received meaning that the first byte in the packet is at location 0x00)
0xF2	Number of bytes in the RXFIFO that should be decrypted

## 5.3 AES TX/RX FIFO Operation

[Section 5.3.1](#) describes how to encrypt TX FIFO content. [Example 2](#) provides an example on how this can be done in the software.

### 5.3.1 TX FIFO Encryption

1. Make sure the radio is in IDLE state.
2. Flush the TXFIFO using the `SFTX` command strobe.
3. Write the packet to the TXFIFO.
4. Set the `MARC_SPARE.AES_COMMAND = 0x09` (`AES_TXFIFO`).
5. Write the 128 bit long AES key to the key location in the AES workspace (`AES_KEY15` is the 7 MSB and starts at address 0x2FE0).
6. Set `SERIAL_STATUS.SPI_DIRECT_ACCESS_CFG = 1`.
7. Write the AES TXFIFO parameters (see [Table 4](#)) to the AES command workspace using direct memory access.
8. Write the nonce to address 0x80 in the AES command workspace using direct memory access.
9. Set `SERIAL_STATUS.SPI_DIRECT_ACCESS_CFG = 0`.
10. Strobe `SIDLE` to execute the `AES_TXFIFO` command.
11. A falling edge on `GPIO0` indicates that the operation is done (if `IOCFG0.PIO0_CFG = AES_COMMAND_ACTIVE (22)`) and an STX strobe can be issued.

**Example 2. AES TX FIFO Encryption**

```

#define AES_BLOCK_SIZE      16
#define PKTLEN              30
static uint8  writeByte;
uint8  txBuffer[PKTLEN];
uint8  ramData[4];
static uint8  aesKey[AES_BLOCK_SIZE];
static uint8  nonce[AES_BLOCK_SIZE];

// 1) Be sure the radio is in IDLE (radio must be in IDLE to execute AES commands)
trxSpiCmdStrobe(CC120X_SIDLE);

// 2) Flush TXFIFO
trxSpiCmdStrobe(CC120X_SFTX);

// 3) Write packet to TXFIFO
ccl20xSpiWriteTxFifo(txBuffer, sizeof(txBuffer));

// 4) Set MARC_SPARE:AES_COMMAND to 0x09 (AES TXFIFO)
writeByte = 0x09;
ccl20xSpiWriteReg(CC120X_MARC_SPARE, &writeByte, 1);

// 5) Write AES key to extended register space address 0xE0
ccl20xSpiWriteReg(CC120X_AES_KEY15, aesKey, sizeof(aesKey));

// 6) Enable SPI direct memory access (SPI_DIRECT_ACCESS_CFG = 1)
writeByte = 0x20;
ccl20xSpiWriteReg(CC120X_SERIAL_STATUS, &writeByte, 1);

// 7) Write AES TXFIFO command
// TXFIFO start pointer @ 0x3EF0 (extended memory space, Free Area 0x3EF0) (word length 2B)
// TXFIFO packet size @ 0x3EF2 (extended memory space, Free Area 0x3EF2) (word length 2B)
ramData[0] = 0x00;           // TXFIFO start pointer
ramData[1] = 0x00;           // zero-pad due to word write in ram
ramData[2] = (PKTLEN+1);     // TXFIFO byte size
ramData[3] = 0x00;           // zero-pad
trx16BitRegAccess((RADIO_BURST_ACCESS|RADIO_WRITE_ACCESS), 0x3E, 0xF0, ramData, sizeof(ramData));

// 8) Write nonce vector (16B) to 0x3880 extended memory space, Free Area 0x2E80
// This vector has to be written byte reversed to what will be put in the AES BUFFER.
// AES command will rotate the vector on execution
trx16BitRegAccess((RADIO_BURST_ACCESS|RADIO_WRITE_ACCESS), 0x3E, 0x80, nonce, sizeof(nonce));

// 9) Disable SPI direct memory access (SPI_DIRECT_ACCESS_CFG = 0)
writeByte = 0x00;
ccl20xSpiWriteReg(CC120X_SERIAL_STATUS, &writeByte, 1);

// 10) Execute AES command by issuing IDLE strobe (radio must be in IDLE state already)
trxSpiCmdStrobe(CC120X_SIDLE);

// 11) Wait for GPIO0 to go low, indicating command finished
// Assumes AES_COMMAND_ACTIVE(0x16) on GPIO0 sets aesSemaphore on falling edge interrupt
while(!aesSemaphore);

```

### 5.3.2 RX FIFO Decryption

Section 5.3.2 describes how to decrypt RX FIFO content. Example 3 provides an example on how this can be done in the software.

1. Make sure the radio is in IDLE state.
2. Flush the RXFIFO using the `SFRX` command strobe.
3. Issue an `SRX` strobe and wait for a packet to be received.
4. Set the `MARC_SPARE.AES_COMMAND = 0x0A` (AES\_RXFIFO).
5. Write the 128 bit long AES key to the key location in the AES workspace (`AES_KEY15` is the 7 MSB and starts at address `0x2FE0`).
6. Set `SERIAL_STATUS.SPI_DIRECT_ACCESS_CFG = 1`.
7. Write the AES RXFIFO parameters (see Table 5) to the AES command workspace using direct memory access.
8. Write the nonce to address `0x80` in the AES command workspace using direct memory access.
9. Set `SERIAL_STATUS.SPI_DIRECT_ACCESS_CFG = 0`.
10. Strobe `SIDLE` to execute the `AES_RXFIFO` command.
11. A falling edge on `GPIO0` indicates that the operation is done (if `IOCFG0.GPIO0_CFG = AES_COMMAND_ACTIVE (22)`) and the RX FIFO can be read.

**Example 3. AES RX FIFO Decryption**

```

#define AES_BLOCK_SIZE      16
#define PKTLEN              30
static uint8 writeByte;
static uint8 rxBytes;
static uint8 ramData[4];
static uint8 aesKey[AES_BLOCK_SIZE];
static uint8 nonce[AES_BLOCK_SIZE];

// 1) Make sure radio is in IDLE state
trxSpiCmdStrobe(CC120X_SIDLE);

// 2) Flush RX FIFO to reset pointers
trxSpiCmdStrobe(CC120X_SFRX);

// 3) Set radio in RX
trxSpiCmdStrobe(CC120X_SRX);

// Wait for packet
while(!packetSemaphore);

// Check available bytes in the RXFIFO
ccl120xSpiReadReg(CC120X_NUM_RXBYTES, &rxBytes,1);

// 4) Set MARC_SPARE.AES_COMMAND to 0x0A (AES RXFIFO)
writeByte = 0x0A;
ccl120xSpiWriteReg(CC120X_MARC_SPARE, &writeByte,1);

// 5) Write AES key to extended register space address 0xE0
ccl120xSpiWriteReg(CC120X_AES_KEY15, aesKey, sizeof(aesKey));

// 6) Enable SPI direct memory access (SPI_DIRECT_ACCESS_CFG = 1)
writeByte = 0x20;
ccl120xSpiWriteReg(CC120X_SERIAL_STATUS,&writeByte,1);

// 7) Write AES TXFIFO marc parameters to RAM: (may need to zero RAM content)
// TXFIFO start pointer @ 0x38F0 (extended memory space, Free Area 0x3EF0) (word length 2B)
// TXFIFO packet size & 0x38F2 (extended memory space, Free Area 0x3EF2) (word length 2B)
ramData[0] = 0x00;           // TXFIFO start pointer
ramData[1] = 0x00;           // zero-pad due to word write in ram
ramData[2] = rxBytes-2;      // payload = number of bytes in the FIFO - appended bytes
ramData[3] = 0x00;           // zero-pad

trx16BitRegAccess((RADIO_BURST_ACCESS|RADIO_WRITE_ACCESS),0x3E,0xF0,ramData,sizeof(ramData));

// 8) Write nonce vector (16B) to 0x3E80 extended memory space.
// This vector has to be written byte reversed to what will be put in the AES BUFFER.
// AES command will rotate the vector on execution
trx16BitRegAccess((RADIO_BURST_ACCESS|RADIO_WRITE_ACCESS),0x3E,0x80,nonce,sizeof(nonce));

// 9) Disable SPI direct memory access (SPI_DIRECT_ACCESS_CFG = 0)
writeByte = 0x00;
ccl120xSpiWriteReg(CC120X_SERIAL_STATUS,&writeByte,1);

// 10) Execute AES command by issuing IDLE strobe (radio must be in IDLE state already)
trxSpiCmdStrobe(CC120X_SIDLE);

// 11) Wait for GPIO0 to go low, indicating command finished
// Assumes AES_COMMAND_ACTIVE(0x16) on GPIO0 sets aesSemaphore on falling edge interrupt
while(!aesSemaphore);

```

## 6 References

- *CC120X Low-Power High Performance Sub-1 GHz RF Transceivers User's Guide* ([SWRU346](#))

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)