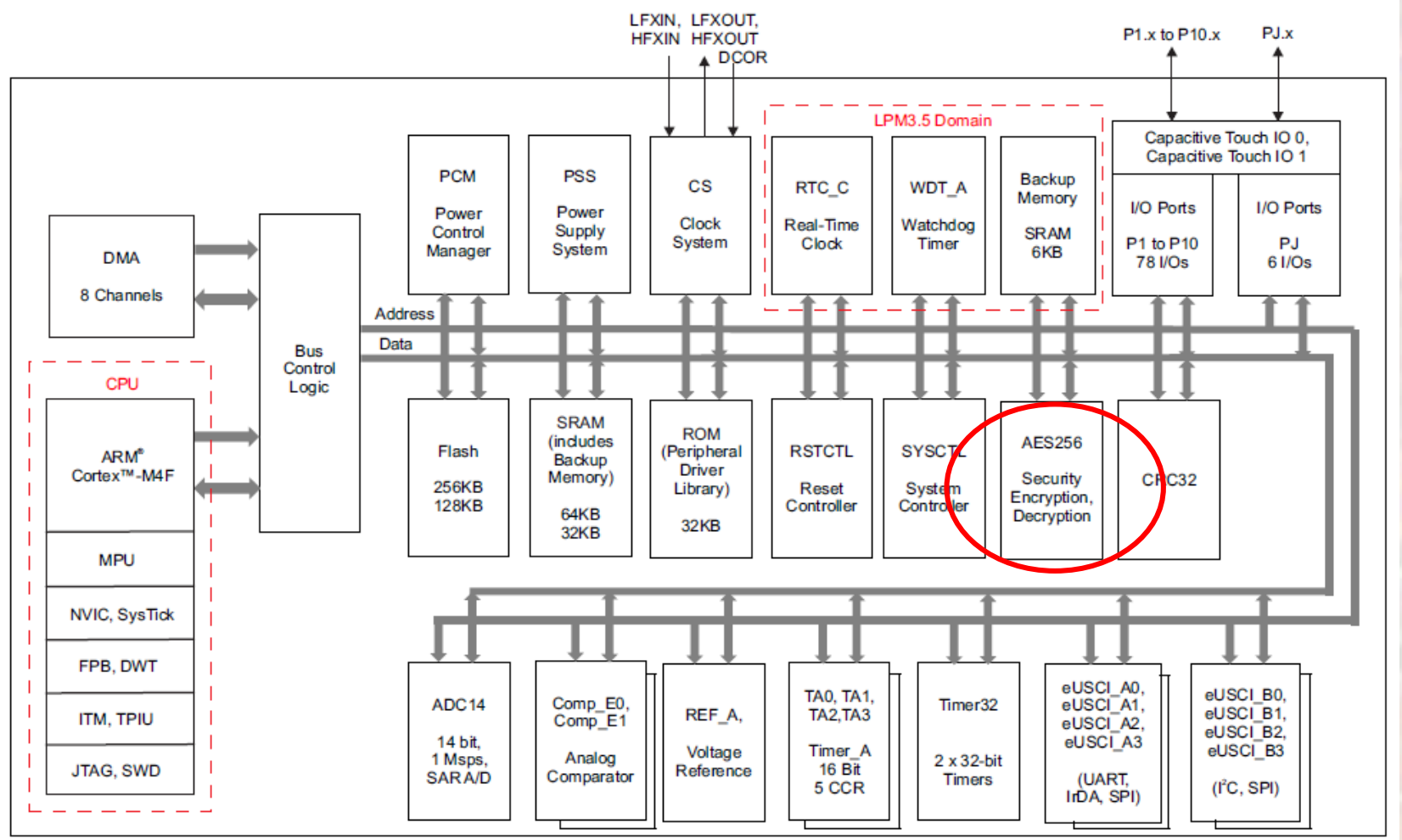
Advanced Encryption Standard

Last updated 6/17/19

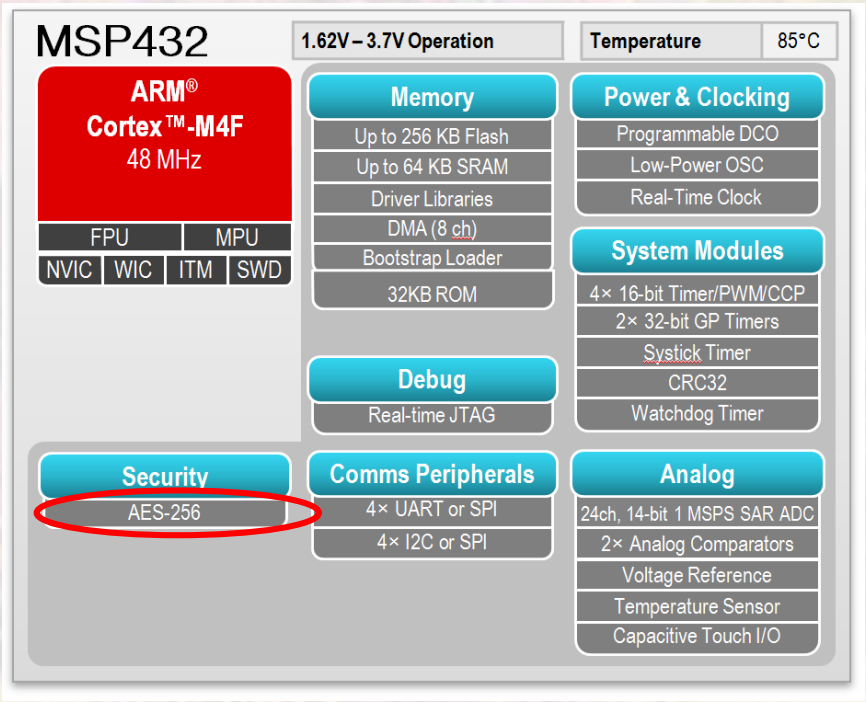
AES

- MSP432 AES



AES

- MSP432 AES
  - AMBA Compliant
  - 128 bit data
  - 128, 192, 256 bit keys



AES

- AES
  - Secret key (private key) – used for encryption and decryption
  - Data stored in an array
  - Several transformations are performed on the array
    - Substitution
    - Row shifting
    - Column mixing
  - The number of rounds is determined by the key length
    - 10 rounds for 128-bit keys
    - 12 rounds for 192-bit keys
    - 14 rounds for 256-bit keys.

AES

- MSP432 AES

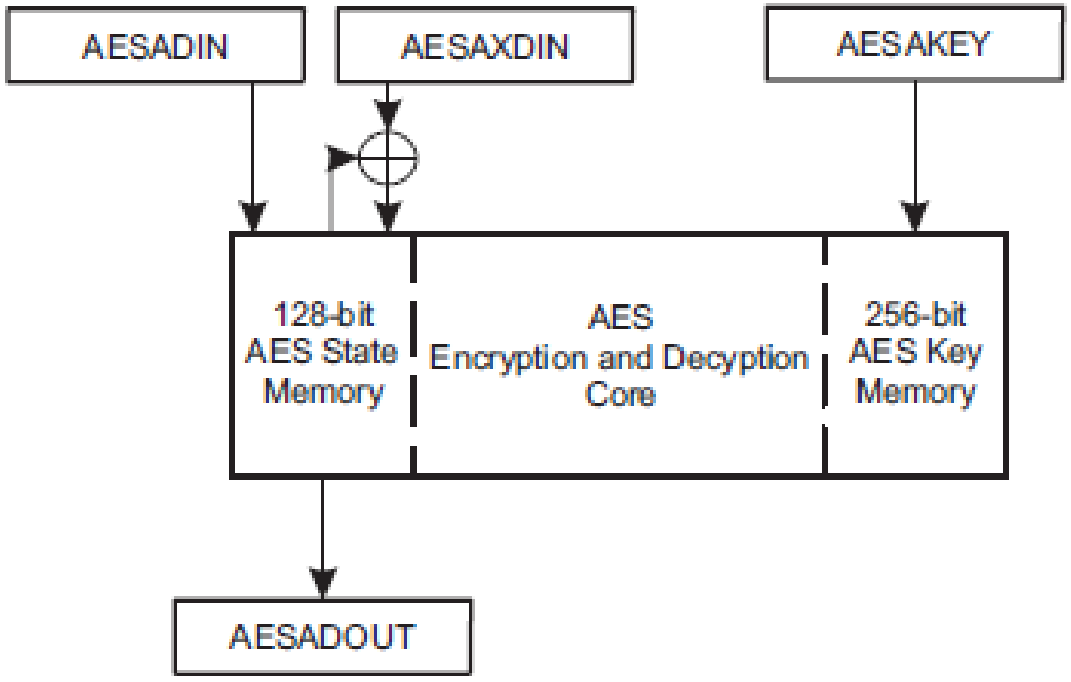


Figure 14-1. AES Accelerator Block Diagram

AES

- MSP432 AES

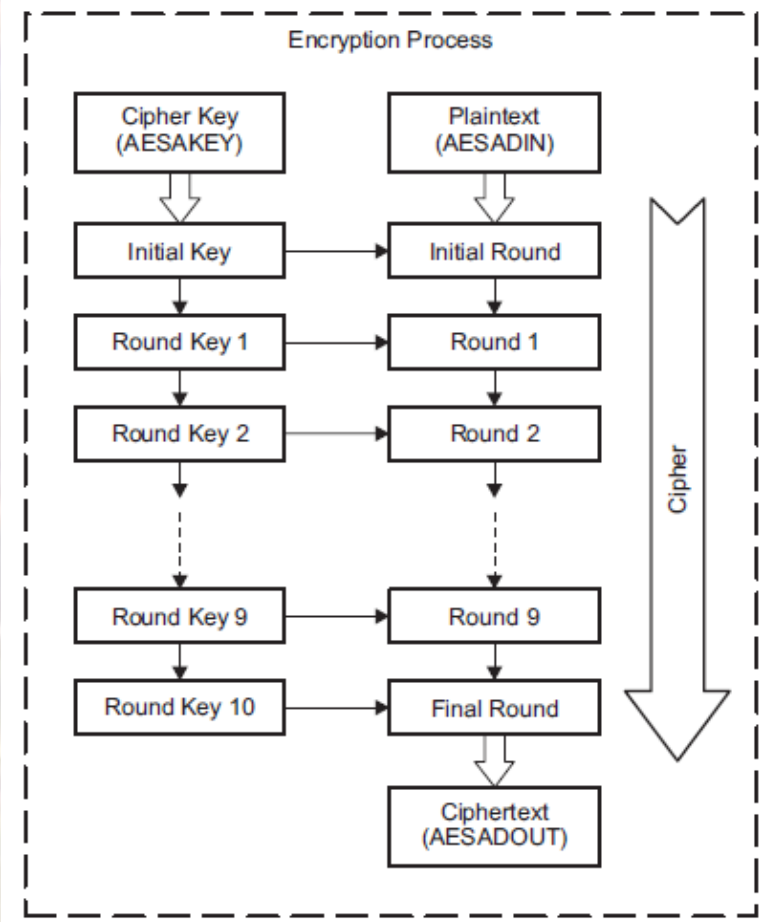


Figure 14-3. AES Encryption Process for 128-Bit Key

AES

- MSP432 AES

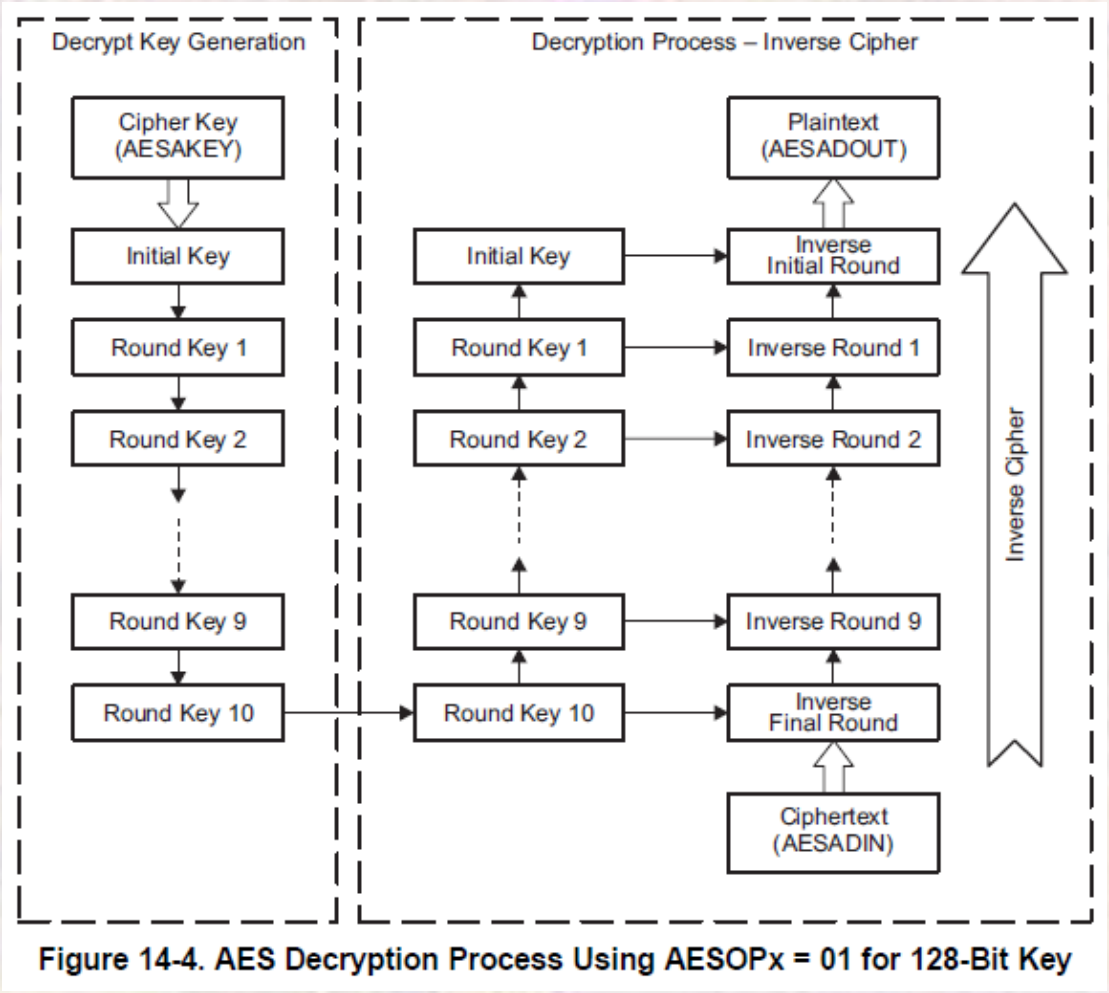


Figure 14-4. AES Decryption Process Using AESOPx = 01 for 128-Bit Key

AES

- MSP432 AES

Table 14-1. AES Operation Modes Overview

AESOPx	AESKLx	Operation	Clock Cycles
00	00	AES128 encryption	168
	01	AES192 encryption	204
	10	AES256 encryption	234
01	00	AES128 decryption (with initial roundkey) is performed	215
	01	AES192 decryption (with initial roundkey) is performed	255
	10	AES256 decryption (with initial roundkey) is performed	292
10	00	AES128 encryption key schedule is performed	53
	01	AES192 encryption key schedule is performed	57
	10	AES256 encryption key schedule is performed	68
11	00	AES128 (with last roundkey) decryption is performed	168
	01	AES192 (with last roundkey) decryption is performed	206
	10	AES256 (with last roundkey) decryption is performed	234

AES

- MSP432 AES

Table 14-11. AES256 Registers

Offset	Acronym	Register Name	Section
00h	AESACTL0	AES accelerator control register 0	Section 14.3.1
02h	AESACTL1	AES accelerator control register 1	Section 14.3.2
04h	AESASTAT	AES accelerator status register	Section 14.3.3
06h	AESAKEY	AES accelerator key register	Section 14.3.4
08h	AESADIN	AES accelerator data in register	Section 14.3.5
0Ah	AESADOUT	AES accelerator data out register	Section 14.3.6
0Ch	AESAXDIN	AES accelerator XORed data in register	Section 14.3.7
0Eh	AESAXIN	AES accelerator XORed data in register (no trigger)	Section 14.3.8

AES

- MSP432 AES

```
/*
 * aes.c
 *
 *  Created on: Aug 13, 2019
 *      Author: johnsontimoj
 */
///////////////////////////////////
//
// AES encryption example
//
// Using simple functions to create the key and original data
//
// encrypting then decrypting the data and printing the results
//
///////////////////////////////////
#include <stdio.h>
#include "msp.h"

void create_data(uint8_t array[], uint8_t length);
void create_key(uint8_t array[]);
void aes_mode_encrypt(void);
void aes_mode_decrypt(void);
void write_key(const uint8_t array[]);
void encrypt(const uint8_t data_array[], uint8_t encrypted_data[], uint8_t length);
void decrypt(const uint8_t data_array[], uint8_t decrypted_data[], uint8_t length);
void print_array(uint8_t array[], uint8_t length);

#define len 16
```

AES

- MSP432 AES

```
int main(void){
    // arrays
    uint8_t aes_key[32];
    uint8_t data_orig[len];
    uint8_t data_encrypted[len];
    uint8_t data_decrypted[len];

    // generate key
    create_key(aes_key);

    // generate original data
    create_data(data_orig, len);

    // Set to encryption mode
    aes_mode_encrypt();

    // write key
    write_key(aes_key);

    // write original data and retrieve encrypted data
    encrypt(data_orig, data_encrypted, len);

    // Set to decryption mode
    aes_mode_decrypt();

    // write key
    write_key(aes_key);

    // write encrypted data and retrieve decrypted data
    decrypt(data_encrypted, data_decrypted, len);

    // print key, original data, encrypted data and decrypted data
    printf("\naes_key: ");
    print_array(aes_key, 32);
    printf("\nOriginal Data: \t\t");
    print_array(data_orig, len);
    printf("\nEncrypted Data: \t");
    print_array(data_encrypted, len);
    printf("\nDecrypted Data: \t");
    print_array(data_decrypted, len);

    return 0;
}
```

AES

```
void create_data(uint8_t array[], uint8_t length){
    uint8_t i;
    // simple data creator - 2xi
    for(i=0; i<length; i++)
        array[i] = 2*i;
}

void create_key(uint8_t array[]){
    // Create the key values and store in an array
    uint8_t i;

    // 256 bit key --> 32 bytes
    // using 3xi for the bytes
    for(i=0; i<32; i++){
        array[i] = 3*i;
    }

    return;
} // end create_key

void aes_mode_encrypt(void){
    // setup CTL0 to set key
    // 256b encryption
    //                  256b Encrypt
    // xxxx xxxx xxxx    10    00
    AES256->CTL0 = 0x0008;
} // end aes_mode_encrypt

void aes_mode_decrypt(void){
    // setup CTL0 to set key
    // 256b decryption
    //                  256b decrypt
    // xxxx xxxx xxxx    10    01
    AES256->CTL0 = 0x0009;
} // end aes_mode_decrypt
```

```
void write_key(const uint8_t array[]){
    uint8_t keyval;
    uint8_t i;

    // Load 256-bit cipher key
    // Key generated by loop index and loaded into KEY register
    // Note: 256b key needs 32 bytes
    // Note: 256b mode requires 16b key writes
    //    accessing the array 2x each loop
    for(i = 0; i < 32; i=i+2){
        //          lower byte          upper byte
        keyval = (uint16_t)(array[i]) | ((uint16_t)(array[i+1]) <<
8);
        // write 16b key each cycle
        AES256->KEY = keyval;
    } // end for

    // stay in fn until key write complete
    // checking for STAT bit 1 to become 1
    while(!(AES256->STAT & 0x02))
        ;

    return;
} // end write_key
```

AES

- MSP432 AES

```
void encrypt(const uint8_t data_array[], uint8_t encrypted_data[], uint8_t length){
    // Load original data and save encrypted version
    uint8_t i;
    uint16_t data_tmp;

    // Note: 256b mode requires 16b data writes
    //    accessing the array 2x each loop
    for(i = 0; i < length; i=i+2){
        // Access and concatentae data
        //          lower byte                              upper byte
        data_tmp = (uint16_t)(data_array[i]) | ((uint16_t)(data_array[i+1]) << 8);
        // Write data to DIN each cycle
        AES256->DIN = data_tmp;
     } // end for

    // stay in fn until key encrypt complete
    // checking for STAT bit 0 (busy) to become 0
    while(AES256->STAT & 0x01)
        ;

    // Note: 256b mode requires 16b data reads
    //    accessing the array 2x each loop
    for(i = 0; i < length; i = i+2){
        // read 16 bit word
        data_tmp = AES256->DOUT;

        //Split word and save in encrypted data array
        encrypted_data[i] = (uint8_t)data_tmp;
        encrypted_data[i+1] = (uint8_t)(data_tmp >> 8);
     } // end for

     return;
}// end encrypt
```

AES

- MSP432 AES

```
void decrypt(const uint8_t data_array[], uint8_t decrypted_data[], uint8_t length){
    // Load encrypted data and save decrypted version
    uint8_t i;
    uint16_t data_tmp;

    // Note: 256b mode requires 16b data writes
    //    accessing the array 2x each loop
    for(i = 0; i < length; i=i+2){
        // Access and concatentae data
        //          lower byte                              upper byte
        data_tmp = (uint16_t)(data_array[i]) | ((uint16_t)(data_array[i+1]) << 8);
        // Write data to DIN each cycle
        AES256->DIN = data_tmp;
     } // end for

    // stay in fn until key decrypt complete
    // checking for STAT bit 0 (busy) to become 0
    while(AES256->STAT & 0x01)
        ;

    // Note: 256b mode requires 16b data reads
    //    accessing the array 2x each loop
    for(i = 0; i < length; i = i+2){
        // read 16 bit word
        data_tmp = AES256->DOUT;

        //Split word and save in encrypted data array
        decrypted_data[i] = (uint8_t)data_tmp;
        decrypted_data[i+1] = (uint8_t)(data_tmp >> 8);
     } // end for

     return;
}// end decrypt
```

AES

- MSP432 AES

```
void print_array(uint8_t array[], uint8_t length){
    uint8_t i;
    for(i=0; i<length; i++)
        printf("%02x", array[i]);
}
```

```
[CORTEX_M4_0]
aes_key: 000306090c0f1215181b1e2124272a2d303336393c3f4245484b4e5154575a5d
Original Data:          00020406080a0c0e10121416181a1c1e
Encrypted Data:         49741928496c91fee8f19dcb7b7ba934
Decrypted Data:         00020406080a0c0e10121416181a1c1e
```