# Software User Manual

## LibMFCrypto (AX8052/AXM0 Crypto Library)

ON Semiconductor®

# TABLE OF CONTENTS

1. Introduction ..... 3
2. Acronyms and Abbreviations ..... 4
3. Microprocessor Functions ..... 5
3.1. ax8052crypto.h ..... 5
3.2. ax8052cryptoregaddr.h ..... 5
3.3. libmfcrypto.h ..... 5
4. Advanced Encryption Standard (AES) ..... 6
4.1. libmfaes.h ..... 6
4.1.1. void aes128_setup(const uint8_t key[16], uint8_t __xdata *keyschedule) void aes192_setup(const uint8_t key[24], uint8_t __xdata *keyschedule) void aes256_setup(const uint8_t key[32], uint8_t __xdata *keyschedule) ..... 6
4.1.2. void aes_keyexp(uint8_t __xdata *keysched, uint8_t Nk, uint8_t Nr) ..... 6
4.1.3. void aes_setup(const uint8_t *key, uint8_t __xdata *keyschedule, uint8_t flag) ..... 6
4.1.4. void aes_keyexp(uint8_t __xdata *keysched, uint8_t nk, uint8_t nr) ..... 7
4.1.5. uint32_t aes_encrypt(uint8_t __xdata *pdata, uint8_t __xdata *cdata, uint32_t block_len, uint8_t __xdata *keyschedule, uint8_t flag) ..... 7
4.1.6. uint32_t aes_decrypt(uint8_t __xdata *cdata, uint8_t __xdata *pdata, uint32_t block_len, uint8_t __xdata *keyschedule, uint8_t flag) ..... 7
5. Data Encryption Standard (DES) ..... 8
5.1. libmfdes.h ..... 8
5.1.1. void des_encrypt(const uint8_t __xdata *inptr, uint8_t __xdata *outptr, const uint8_t __xdata *keysched, uint8_t nr) ..... 8
5.1.2. void des_decrypt(const uint8_t __xdata *inptr, uint8_t __xdata *outptr, const uint8_t __xdata *keysched, uint8_t nr) ..... 8
5.1.3. void des_keyexp(const uint8_t key[8], uint8_t __xdata *keysched) ..... 8
6. History ..... 9
7. Contact Information ..... 10

# 1. INTRODUCTION

LibMFCrypto is a library of cryptographic functions for the AX8052/AXM0 Microprocessor. It contains the following features:

- Advanced Encryption Standard (AES) Key Schedule Generation and Register Definitions for the AX8052 AES Engine
- Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) and Output Feedback (OFB) mode for the AXM0F243 MCU.
- Data Encryption Standard (DES) Encryption and Decryption for AX8052
- Register Definitions for the AX8052 true random number generator

LibMFCrypto is available in source and binary form for SDCC, Keil C51 and IAR ICC for AX8052 MCU.

LibMFCrypto is available in source and binary form for ARM GCC for AXM0F243 MCU.

## 2. Acronyms and Abbreviations

| | |
|---|---|
| AX8052 | MCU 8052 |
| AX8052F143 | MCU 8052 + RADIO AX5043 |
| AXM0 | MCU ARM Cortex M0 Plus |
| AXM0F243 | MCU ARM Cortex M0 Plus + RADIO AX5043 |
| SDCC | Small Device C Compiler |
| GCC | GNU Compiler Collection |

# 3. Microprocessor Functions

## 3.1. ax8052crypto.h

This header define the special function registers (SFR) of the AX8052 Microprocessor AES and True Random Number Generator peripherals.

## 3.2. ax8052cryptoregaddr.h

This header provides defines for the AX8052 special function register (SFR) addresses of the AX8052 Microprocessor AES and True Random Number Generator peripherals. Contrary to ax8052.h, which provides defines that, when used like variables, access the registers, the ax8052regaddr.h header file only provides defines for the register addresses.

## 3.3. libmfcrypto.h

This header includes the libmfaes.h and libmfdes.h headers, and provides the LIBMFCRYPTOVERSION define, indicating the library version.

# 4. Advanced Encryption Standard (AES)

## 4.1. libmfaes.h

**AX8052:**

For AX8052 microprocessor, this header provides functions for computing the AES key schedule, as well as setting the AES engine registers to the computed key schedule.

### 4.1.1. void aes128_setup(const uint8_t key[16], uint8_t __xdata *keyschedule)
### void aes192_setup(const uint8_t key[24], uint8_t __xdata *keyschedule)
### void aes256_setup(const uint8_t key[32], uint8_t __xdata *keyschedule)

These routines compute the AES key schedule for AES-128, AES-192 and AES-256, respectively. They receive a pointer to the key, as well as a pointer to a buffer in XRAM that receives the key schedule. The buffer must be able to hold at least 176, 208 or 240 bytes for AES-128, AES-192 and AES-256, respectively. The buffer should be even address aligned for performance reasons. This routine also updates the AES core registers with a pointer to the newly computed key schedule.

### 4.1.2. void aes_keyexp(uint8_t __xdata *keysched, uint8_t Nk, uint8_t Nr)

This function does the actual key schedule expansion. It must be given the correct key length and number of rounds for the AES mode. This routine is used by the aes setup family of functions, and normally needs not be called by the user.

**AXM0F243:**

For AXM0 microprocessor, this header provides functions for computing the AES key schedule and for performing the AES encryption and decryption in Cipher Block Chaining (CBC) and Output Feedback (OFB) mode.

### 4.1.3. void AES_SETUP(const uint8_t *key, uint8_t __xdata *keyschedule, uint8_t flag)

This function compute the AES key schedule for AES-128, AES-192 and AES-256. It receives a pointer to the key and a pointer to the buffer that receives the key schedule. The buffer must be able to hold at least 176, 208 or 240 bytes for AES-128, AES-192 and AES-256, respectively. The appropriate AES key size must be given in the flag parameter.

The flag parameter value can be any one of the following AES_KEYSIZE_128, AES_KEYSIZE_192, AES_KEYSIZE_256, AES_CBC_128, AES_CBC_192, AES_CBC_256, AES_OFB_128, AES_OFB_192 or AES_OFB_256 as defined in enum aes_mode.

### 4.1.4. VOID AES_KEYEXP(UINT8_T __XDATA *KEYSCHED, UINT8_T NK, UINT8_T NR)

This function does the actual key schedule expansion. It must be given the correct key length and number of rounds for the AES mode. This routine is used by the aes_setup function and normally needs not be called by the user.

### 4.1.5. UINT32_T AES_ENCRYPT(UINT8_T __XDATA *PDATA, UINT8_T __XDATA *CDATA, UINT32_T BLOCK_LEN, UINT8_T __XDATA *KEYSCHEDULE, UINT8_T FLAG)

This function performs the encryption either in Cipher Block Chaining (CBC) mode or Output Feedback (OFB) mode. It checks for valid block length and returns 1 for invalid block length value. If block length is valid, it checks the flag to determine the mode of encryption and performs encryption in the corresponding mode. After encryption, it returns the block length which is the number of blocks of encrypted plain text.

pdata is pointer to the input plain text. The first block of the input plain text will be used as the Initialization vector (IV). cdata is pointer to the cipher text. The parameter block_len is the number of blocks in the input plain text excluding the Initialization vector (IV).

The parameter keyschedule is pointer to the key schedule. aes_setup function must be called before the aes_encrypt function to generate the key schedule.

The flag parameter value can be any one of the following AES_CBC_128, AES_CBC_192, AES_CBC_256, AES_OFB_128, AES_OFB_192 or AES_OFB_256 as defined in enum aes_mode.

### 4.1.6. UINT32_T AES_DECRYPT(UINT8_T __XDATA *CDATA, UINT8_T __XDATA *PDATA, UINT32_T BLOCK_LEN, UINT8_T __XDATA *KEYSCHEDULE, UINT8_T FLAG)

This function performs the decryption either in Cipher Block Chaining (CBC) mode or Output Feedback (OFB) mode. It checks for valid block length and returns 1 for invalid block length value. If block length is valid, it checks the flag to determine the mode of encryption and performs decryption in the corresponding mode. After decryption, it returns the block length which is the number of blocks of decrypted cipher text.

cdata is pointer to the input cipher text. The first block of the input cipher text will be used as the Initialization vector (IV). pdata is pointer to the plain text. The parameter block_len is the number of blocks in the input cipher text excluding the Initialization vector (IV).

The parameter keyschedule is pointer to the key schedule. aes_setup function must be called before the aes_decrypt function to generate the key schedule.

The flag parameter value can be any one of the following AES_CBC_128, AES_CBC_192, AES_CBC_256, AES_OFB_128, AES_OFB_192 or AES_OFB_256 as defined in enum aes_mode.

# 5. Data Encryption Standard (DES)

## 5.1. libmfdes.h

**AX8052:**

### 5.1.1. void des_encrypt(const uint8_t __xdata *inptr, uint8_t __xdata *outptr, const uint8_t __xdata *keysched, uint8_t nr)

This function encrypts a data block using the DES algorithm. It receives pointer to XRAM buffers for the input data, the output data, and the expanded key schedule (see chapter 4.1.3). nr specifies the number of 8 byte (64 bit) blocks to encrypt.

The SDCC version takes about 12000 cycles to compute encryption of one 8 byte block. Other compilers may vary.

### 5.1.2. void des_decrypt(const uint8_t __xdata *inptr, uint8_t __xdata *outptr, const uint8_t __xdata *keysched, uint8_t nr)

This function decrypts a data block using the DES algorithm. It receives pointer to XRAM buffers for the input data, the output data, and the expanded key schedule (see chapter 4.1.3). nr specifies the number of 8 byte (64 bit) blocks to decrypt.

The SDCC version takes about 12000 cycles to compute decryption of one 8 byte block. Other compilers may vary.

### 5.1.3. void des_keyexp(const uint8_t key[8], uint8_t __xdata *keysched)

This function computes the expanded key schedule from the original key. It receives a pointer to the original key data, as well as an XRAM buffer to place the key schedule in. This buffer must be able to hold 128 bytes.

The SDCC version takes about 10000 cycles to compute the key schedule. Other compilers may vary.

# 6. HISTORY

| Version | Date | Comments |
|---|---|---|
| 1.0 | | Added support for AX8052/AX8052F143 Crypto library |
| 1.1 | 17-Aug-2018 | Added support for AXM0F243 Crypto library |

# 7. Contact Information

**ON Semiconductor**
Oskar-Bider-Strasse 1
CH-8600 Dübendorf
SWITZERLAND

Phone +41 44 882 17 07
Fax +41 44 882 17 09
Email sales@onsemi.com
www.onsemi.com

For further product related or sales information please visit our website or contact your local representative.

ON Semiconductor and are trademarks of Semiconductor Components Industries, LLC dba ON Semiconductor or its subsidiaries in the United States and/or other countries. ON Semiconductor owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of ON Semiconductor's product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent-Marking.pdf. ON Semiconductor reserves the right to make changes without further notice to any products herein. ON Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ON Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using ON Semiconductor products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by ON Semiconductor. "Typical" parameters which may be provided in ON Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. ON Semiconductor does not convey any license under its patent rights nor the rights of others. ON Semiconductor products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use ON Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold ON Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that ON Semiconductor was negligent regarding the design or manufacture of the part. ON Semiconductor is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.