# AT07909: SAM4C/4E Advanced Encryption Standard (AES) Driver

**APPLICATION NOTE**

## Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's Advanced Encryption Standard functionality.

The Advanced Encryption Standard module supports all five confidentiality modes of operation for symmetrical key block cipher algorithms (as specified in the NIST Special Publication 800-38A Recommendation):

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Output Feedback (OFB)
- Cipher Feedback (CFB)
- Counter (CTR)

Devices from the following series can use this module:

- Atmel | SMART SAM4C
- Atmel | SMART SAM4E

The outline of this documentation is as follows:


- Prerequisites
- Module Overview
- Special Considerations
- Extra Information
- Examples
- API Overview

# Table of Contents


Introduction....1

1. Software License....4

2. Prerequisites....5

3. Module Overview....6

4. Special Considerations....7
   - 4.1. Power Management....7
   - 4.2. Interrupt....7

5. Extra Information....8

6. Examples....9

7. API Overview....10
   - 7.1. Variable and Type Definitions....10
     - 7.1.1. Type aes_callback_t....10
     - 7.1.2. Type aes_interrupt_source_t....10
   - 7.2. Structure Definitions....10
     - 7.2.1. Struct aes_config....10
   - 7.3. Macro Definitions....10
     - 7.3.1. Macro AES_DMA_RX_IDX....10
     - 7.3.2. Macro AES_DMA_TX_IDX....11
   - 7.4. Function Definitions....11
     - 7.4.1. Function aes_disable()....11
     - 7.4.2. Function aes_disable_interrupt()....11
     - 7.4.3. Function aes_enable()....11
     - 7.4.4. Function aes_enable_interrupt()....11
     - 7.4.5. Function aes_get_config_defaults()....12
     - 7.4.6. Function aes_get_pdc_base()....12
     - 7.4.7. Function aes_init()....12
     - 7.4.8. Function aes_read_authen_datalength()....13
     - 7.4.9. Function aes_read_gcm_counter()....13
     - 7.4.10. Function aes_read_gcmh()....13
     - 7.4.11. Function aes_read_ghash()....14
     - 7.4.12. Function aes_read_interrupt_mask()....14
     - 7.4.13. Function aes_read_interrupt_status()....14
     - 7.4.14. Function aes_read_output_data()....15
     - 7.4.15. Function aes_read_pctext_length()....15
     - 7.4.16. Function aes_read_tag()....15
     - 7.4.17. Function aes_reset()....16
     - 7.4.18. Function aes_set_callback()....16
     - 7.4.19. Function aes_set_config()....16
     - 7.4.20. Function aes_start()....17

7.4.21. Function aes_write_authen_datalength() 17
7.4.22. Function aes_write_gcmh() 17
7.4.23. Function aes_write_ghash() 18
7.4.24. Function aes_write_initvector() 18
7.4.25. Function aes_write_input_data() 18
7.4.26. Function aes_write_key() 19
7.4.27. Function aes_write_pctext_length() 19
7.5. Enumeration Definitions 19
7.5.1. Enum aes_cfb_size 19
7.5.2. Enum aes_encrypt_mode 20
7.5.3. Enum aes_interrupt_source 20
7.5.4. Enum aes_key_size 20
7.5.5. Enum aes_opmode 21
7.5.6. Enum aes_start_mode 21

8. Extra Information for Advanced Encryption Standard 22
8.1. Acronyms 22
8.2. Dependencies 22
8.3. Errata 22
8.4. Module History 22

9. Examples for Advanced Encryption Standard 24
9.1. Quick Start Guide for the AES Driver 24
9.1.1. Use Cases 24
9.1.2. AES Basic Usage 24
9.1.3. Setup Steps 24
9.1.4. Usage Steps 25
9.2. Advanced Encryption Standard - Example Cipher Operating Modes and DMA 25
9.2.1. Purpose 25
9.2.2. Requirements 25
9.2.3. Description 25
9.2.4. Main Files 25
9.2.5. Compilation Info 25
9.2.6. Usage 26

10. Document Revision History 27

# 1. Software License

# 2. Prerequisites

There are no prerequisites for this module.

# 3. Module Overview

The Advanced Encryption Standard (AES) module is compliant with the American FIPS (Federal Information Processing Standard) Publication 197 specification.

The AES supports all five confidentiality modes of operation for symmetrical key block cipher algorithms (as specified in the NIST Special Publication 800-38A Recommendation) :

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Output Feedback (OFB)
- Cipher Feedback (CFB)
- Counter (CTR)

Data transfers both to and from the AES module can occur using the Peripheral DMA Controller (PDC) channels (thus minimizing processor intervention for large data buffer transfers).

As soon as the initialization vector, the input data, and the key are configured, the encryption/decryption process may be started. Once the process has completed the encrypted/decrypted data can be read out via registers or through DMA channels.

# 4. Special Considerations

## 4.1. Power Management

The AES module may be clocked through the Power Management Controller (PMC), in which case the user application must first configure the PMC to enable the AES clock.

## 4.2. Interrupt

When using the AES module's interrupt, the configuration of the device's Nested Vectored Interrupt Controller (NVIC) needs to be carried out before the AES module is configured.

# 5. Extra Information

For extra information, see Extra Information for Advanced Encryption Standard. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

# 6. Examples

For a list of examples related to this driver, see Examples for Advanced Encryption Standard.

# 7. API Overview

## 7.1. Variable and Type Definitions

### 7.1.1. Type aes_callback_t

```
typedef void(* aes_callback_t )(void)
```

AES interrupt callback function type.

### 7.1.2. Type aes_interrupt_source_t

```
typedef enum aes_interrupt_source aes_interrupt_source_t
```

AES interrupt source type.

## 7.2. Structure Definitions

### 7.2.1. Struct aes_config

AES Configuration structure.

**Table 7-1 Members**

| Type | Name | Description |
|---|---|---|
| enum aes_cfb_size | cfb_size | Cipher feedback data size. |
| enum aes_encrypt_mode | encrypt_mode | AES data mode (decryption or encryption). |
| bool | gtag_en | Galois Counter Mode (GCM) automatic tag generation enable/disable (SAM4C devices only). |
| enum aes_key_size | key_size | AES key size. |
| bool | lod | Last output data mode enable/disable. |
| enum aes_opmode | opmode | AES block cipher operation mode. |
| uint32_t | processing_delay | Processing delay parameter. |
| enum aes_start_mode | start_mode | Start mode. |

## 7.3. Macro Definitions

### 7.3.1. Macro AES_DMA_RX_IDX

```
#define AES_DMA_RX_IDX
```

AES DMAC RX channel interface number.

### 7.3.2. Macro AES_DMA_TX_IDX

```
#define AES_DMA_TX_IDX
```

AES DMAC TX channel interface number.

## 7.4. Function Definitions

### 7.4.1. Function aes_disable()

Disable the AES module.

```
void aes_disable( void )
```

### 7.4.2. Function aes_disable_interrupt()

Disable an AES interrupt.

```
void aes_disable_interrupt(
        Aes *const p_aes,
        aes_interrupt_source_t source)
```

**Table 7-2 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | source | Interrupt source |

### 7.4.3. Function aes_enable()

Enable the AES module.

```
void aes_enable( void )
```

### 7.4.4. Function aes_enable_interrupt()

Enable an AES interrupt.

```
void aes_enable_interrupt(
        Aes *const p_aes,
        aes_interrupt_source_t source)
```

**Table 7-3 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | source | Interrupt source |

### 7.4.5. Function aes_get_config_defaults()

Initializes an AES configuration structure to defaults.

```
void aes_get_config_defaults(
        struct aes_config *const p_cfg)
```

Initializes the specified AES configuration structure to a set of known default values.

**Note:** This function should be called to initialize *all* new instances of AES configuration structures before they are further modified by the user application.

The default configuration is as follows:

- Data encryption
- 128-bit AES key size
- 128-bit cipher feedback size
- Manual start mode
- Electronic Codebook (ECB) mode
- Last output data mode is disabled
- No extra delay

**Table 7-4 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_cfg | Pointer to an AES configuration structure |

### 7.4.6. Function aes_get_pdc_base()

Get AES PDC base address.

```
Pdc * aes_get_pdc_base(
        Aes * p_aes)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-5 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |

**Returns**
The PDC registers base address for the AES module.

### 7.4.7. Function aes_init()

Initialize the AES module.

```
void aes_init(
        Aes *const p_aes,
        struct aes_config *const p_cfg)
```

**Table 7-6 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | p_cfg | Pointer to an AES configuration structure |

### 7.4.8. Function aes_read_authen_datalength()

Get the AES Additional Authenticated Data (AAD) length in bytes.

```
uint32_t aes_read_authen_datalength(
        Aes *const p_aes)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-7 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |

**Returns**
The Additional Authenticated Data (AAD) length in bytes.

### 7.4.9. Function aes_read_gcm_counter()

Get the AES GCM Encryption Counter.

```
uint32_t aes_read_gcm_counter(
        Aes *const p_aes)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-8 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |

**Returns**
The AES GCM encryption counter.

### 7.4.10. Function aes_read_gcmh()

Get AES GCM Hash subkey word.

```
uint32_t aes_read_gcmh(
        Aes *const p_aes,
        uint32_t id)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-9 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |
| **[in]** | id | Index into the GCMHR array (range 0 to 3) |

**Returns**
The contents of the AES_GCMHRx[x = 0...3] register specified.

### 7.4.11. Function aes_read_ghash()

Get the AES GCM Intermediate Hash Word.

```
uint32_t aes_read_ghash(
        Aes *const p_aes,
        uint32_t id)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-10 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |
| **[in]** | id | Index into the GHASH array (range 0 to 3) |

**Returns**
The content of the AES_GHASHRx[x = 0...3] register.

### 7.4.12. Function aes_read_interrupt_mask()

Get the AES interrupt mask status.

```
uint32_t aes_read_interrupt_mask(
        Aes *const p_aes)
```

**Table 7-11 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |

**Returns**
The AES interrupt mask contents.

### 7.4.13. Function aes_read_interrupt_status()

Get the AES interrupt status.

```
uint32_t aes_read_interrupt_status(
        Aes *const p_aes)
```

**Table 7-12 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |

**Returns**
The AES interrupt status register contents.

### 7.4.14. Function aes_read_output_data()

Read the output data.

```
void aes_read_output_data(
        Aes *const p_aes,
        uint32_t * p_output_data_buffer)
```

**Note:** The data buffer that holds the processed data must be large enough to hold four consecutive 32-bit words.

**Table 7-13 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |
| **[in]** | *p_output_data_buffer | Pointer to an output buffer |

### 7.4.15. Function aes_read_pctext_length()

Get the AES plaintext/ciphertext length in bytes.

```
uint32_t aes_read_pctext_length(
        Aes *const p_aes)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-14 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |

**Returns**
The plaintext/ciphertext length in bytes.

### 7.4.16. Function aes_read_tag()

Get AES GCM Authentication Tag Word.

```
uint32_t aes_read_tag(
        Aes *const p_aes,
        uint32_t id)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-15 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[in]** | p_aes | Module hardware register base address pointer |
| **[in]** | id | Index into the TAGR array (range 0 to 3) |

**Returns**
The contents of the AES_TAGRx[x = 0...3] register specified.

### 7.4.17. Function aes_reset()

Perform an AES software reset.

```
void aes_reset(
        Aes *const p_aes)
```

**Table 7-16 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |

### 7.4.18. Function aes_set_callback()

Set the AES interrupt callback.

```
void aes_set_callback(
        Aes *const p_aes,
        aes_interrupt_source_t source,
        aes_callback_t callback,
        uint8_t irq_level)
```

**Table 7-17 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | source | Interrupt source |
| **[in]** | callback | Interrupt callback function pointer |
| **[in]** | irq_level | Interrupt priority level |

### 7.4.19. Function aes_set_config()

Configure the AES module.

```
void aes_set_config(
        Aes *const p_aes,
        struct aes_config *const p_cfg)
```

**Table 7-18 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | p_cfg | Pointer to an AES configuration structure |

### 7.4.20. Function aes_start()

Start a manual encryption/decryption process.

```
void aes_start(
        Aes *const p_aes)
```

**Table 7-19 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |

### 7.4.21. Function aes_write_authen_datalength()

Set the AES Additional Authenticated Data (AAD) length in bytes.

```
void aes_write_authen_datalength(
        Aes *const p_aes,
        uint32_t length)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-20 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | length | Length in bytes of the AAD data to be processed |

### 7.4.22. Function aes_write_gcmh()

Set the AES GCM Hash subkey word.

```
void aes_write_gcmh(
        Aes *const p_aes,
        uint32_t id,
        uint32_t hword)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-21 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | id | Index into the GCMHR array (range 0 to 3) |
| **[in]** | hword | GCM H Word |

### 7.4.23. Function aes_write_ghash()

Set the AES GCM Intermediate Hash Word.

```
void aes_write_ghash(
        Aes *const p_aes,
        uint32_t id,
        uint32_t ghash)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-22 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | id | Index into the GHASHx array (range 0 to 3) |
| **[in]** | ghash | Intermediate GCM Hash Word x |

### 7.4.24. Function aes_write_initvector()

Write the initialization vector (for the CBC, CFB, OFB, CTR & GCM cipher modes).

```
void aes_write_initvector(
        Aes *const p_aes,
        const uint32_t * p_vector)
```

**Table 7-23 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | p_vector | Pointer to four contiguous 32-bit words |

### 7.4.25. Function aes_write_input_data()

Write the input data (four consecutive 32-bit words).

```
void aes_write_input_data(
        Aes *const p_aes,
        const uint32_t * p_input_data_buffer)
```

**Table 7-24 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | p_input_data_buffer | Pointer to an input data buffer |

### 7.4.26. Function aes_write_key()

Write the 128/192/256-bit cryptographic key.

```
void aes_write_key(
        Aes *const p_aes,
        const uint32_t * p_key)
```

**Table 7-25 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | p_key | Pointer to 4/6/8 contiguous 32-bit words |

**Note:** The key size depends on the current AES configuration.

### 7.4.27. Function aes_write_pctext_length()

Set the AES plaintext/ciphertext length in bytes.

```
void aes_write_pctext_length(
        Aes *const p_aes,
        uint32_t length)
```

**Note:** This function is only available on SAM4C devices.

**Table 7-26 Parameters**

| Data direction | Parameter name | Description |
|---|---|---|
| **[out]** | p_aes | Module hardware register base address pointer |
| **[in]** | length | Length in bytes of the plaintext/ciphertext data |

## 7.5. Enumeration Definitions

### 7.5.1. Enum aes_cfb_size

AES Cipher FeedBack (CFB) size.

**Table 7-27 Members**

| Enum value | Description |
|---|---|
| AES_CFB_SIZE_128 | Cipher feedback data size is 128-bit. |
| AES_CFB_SIZE_64 | Cipher feedback data size is 64-bit. |

| Enum value | Description |
|---|---|
| AES_CFB_SIZE_32 | Cipher feedback data size is 32-bit. |
| AES_CFB_SIZE_16 | Cipher feedback data size is 16-bit. |
| AES_CFB_SIZE_8 | Cipher feedback data size is 8-bit. |

### 7.5.2. Enum aes_encrypt_mode

AES processing mode.

**Table 7-28 Members**

| Enum value | Description |
|---|---|
| AES_DECRYPTION | Decryption of data will be performed. |
| AES_ENCRYPTION | Encryption of data will be performed. |

### 7.5.3. Enum aes_interrupt_source

AES interrupt source type.

**Table 7-29 Members**

| Enum value | Description |
|---|---|
| AES_INTERRUPT_DATA_READY | Data ready interrupt. |
| AES_INTERRUPT_UNSPECIFIED_REGISTER_ACCESS | Unspecified register access detection interrupt. |
| AES_INTERRUPT_END_OF_RECEIVE_BUFFER | End of receive buffer interrupt (SAM4C devices only). |
| AES_INTERRUPT_END_OF_TRANSMIT_BUFFER | End of transmit buffer interrupt (SAM4C devices only). |
| AES_INTERRUPT_RECEIVE_BUFFER_FULL | Receive buffer full interrupt (SAM4C devices only). |
| AES_INTERRUPT_TRANSMIT_BUFFER_FULL | Transmit buffer empty interrupt (SAM4C devices only). |

### 7.5.4. Enum aes_key_size

AES cryptographic key size.

**Table 7-30 Members**

| Enum value | Description |
|---|---|
| AES_KEY_SIZE_128 | AES key size is 128 bits. |
| AES_KEY_SIZE_192 | AES key size is 192 bits. |
| AES_KEY_SIZE_256 | AES key size is 256 bits. |

### 7.5.5. Enum aes_opmode

AES cipher block mode.

**Table 7-31 Members**

| Enum value | Description |
|---|---|
| AES_ECB_MODE | Electronic Codebook (ECB). |
| AES_CBC_MODE | Cipher Block Chaining (CBC). |
| AES_OFB_MODE | Output Feedback (OFB). |
| AES_CFB_MODE | Cipher Feedback (CFB). |
| AES_CTR_MODE | Counter (CTR). |
| AES_GCM_MODE | Galois Counter Mode (GCM). |

### 7.5.6. Enum aes_start_mode

AES start mode.

**Table 7-32 Members**

| Enum value | Description |
|---|---|
| AES_MANUAL_START | Manual start mode. |
| AES_AUTO_START | Auto start mode. |
| AES_IDATAR0_START | AES_IDATAR0 access only Auto Mode. |

# 8. Extra Information for Advanced Encryption Standard

## 8.1. Acronyms

Below is a table listing the acronyms used in this module, along with their intended meanings.

| Acronym | Definition |
|---|---|
| AAD | Additional Authenticated Data |
| CBC | Cipher Block Chaining |
| CFB | Cipher Feedback |
| CTR | Counter |
| DMA | Direct Memory Access |
| DMAC | DMA Controller |
| ECB | Electronic Codebook |
| GCM | Galois Counter Mode |
| NVIC | Nested Vectored Interrupt Controller |
| OFB | Output Feedback |
| PDC | Peripheral DMA Controller |
| PMC | Power Management Controller |
| QSG | Quick Start Guide |

## 8.2. Dependencies

This driver has the following dependencies:

- None

## 8.3. Errata

There are no errata related to this driver.

## 8.4. Module History

An overview of the module history is presented in the table below, with details on the enhancements and fixes made to the module since its first release. The current version of this corresponds to the newest version in the table.

| Changelog |
| --- |
| Initial document release |

# 9. Examples for Advanced Encryption Standard

This is a list of the available Quick Start Guides (QSGs) and example applications for SAM4C/4E Advanced Encryption Standard (AES) Driver. QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that QSGs can be compiled as a standalone application or be added to the user application.

- Quick Start Guide for the AES Driver
- Advanced Encryption Standard - Example Cipher Operating Modes and DMA

## 9.1. Quick Start Guide for the AES Driver

This is the quick start guide for the SAM4C/4E Advanced Encryption Standard (AES) Driver, with step-by-step instructions on how to configure and use the driver for a specific use case. The code examples can be copied into any function that needs to control the AES module, such as the main application loop, for example.

### 9.1.1. Use Cases

- AES Basic Usage

### 9.1.2. AES Basic Usage

This use case will demonstrate how to initialize the AES module to perform encryption or decryption of data.

### 9.1.3. Setup Steps

#### 9.1.3.1. Prerequisites

This module requires the following service:

- System clock (sysclk)

#### 9.1.3.2. Setup Code

Add this to the main loop or a setup function:

```
struct aes_config   g_aes_cfg;
aes_get_config_defaults(&g_aes_cfg);
aes_init(AES, &g_aes_cfg);
aes_enable();
```

#### 9.1.3.3. Workflow

1. Enable the AES module:

   ```
   aes_enable();
   ```

2. Set the AES interrupt and callback:

   ```
   aes_set_callback(AES, AES_INTERRUPT_DATA_READY,
        aes_callback, 1);
   ```

3. Initialize the AES to ECB cipher mode:

   ```
   g_aes_cfg.encrypt_mode = AES_ENCRYPTION;
   g_aes_cfg.key_size = AES_KEY_SIZE_128;
   g_aes_cfg.start_mode = AES_AUTO_MODE;
   ```

```
g_aes_cfg.opmode = AES_ECB_MODE;
g_aes_cfg.cfb_size = AES_CFB_SIZE_128;
g_aes_cfg.lod = false;
aes_set_config(AES, &g_aes_cfg);
```

#### 9.1.4. Usage Steps

##### 9.1.4.1. Usage Code

Plain text can be encrypted by:

```
aes_write_key(AES, key128);
aes_write_input_data(AES, ref_plain_text);
```

The cipher text can be retrieved after it's been encrypted by:

```
aes_read_output_data(AES, output_data);
```

## 9.2. Advanced Encryption Standard - Example Cipher Operating Modes and DMA

### 9.2.1. Purpose

This example demonstrates how to use the AES driver to perform:

- ECB encryption and decryption
- CBC encryption and decryption
- CFB encryption and decryption
- OFB encryption and decryption
- CTR encryption and decryption
- ECB encryption and decryption using DMA/PDC

### 9.2.2. Requirements

All SAM devices with an AES module can be used. This example has been tested with the following evaluation kits:

- SAM4E EK
- SAM4C EK

### 9.2.3. Description

Upon startup, the program uses the USART driver to display a menu from which several encryption/decryption modes can be tested.

### 9.2.4. Main Files

- aes.c : AES driver
- aes.h : AES header file
- aes_example.c : AES code example

### 9.2.5. Compilation Info

This software was written for the GNU GCC and IAR Systems compiler. Other compilers may or may not work.

### 9.2.6. Usage

1. Build the program and download it into the evaluation board.
2. On the computer, open and configure a terminal application (e.g., HyperTerminal on Microsoft® Windows®) with these settings:
   - 115200 baud
   - 8 bits of data
   - No parity
   - 1 stop bit
   - No flow control
3. Start the application.
4. In the terminal window, the following text should appear:

```
-- AES Example --
-- xxxxxx-xx
-- Compiled: xxx xx xxxx xx:xx:xx --

Menu :
  -- Select operation:
  h: Display menu
  1: ECB mode test.
  2: CBC mode test.
  3: CFB128 mode test.
  4: OFB mode test.
  5: CTR mode test.
  d: ECB mode test with DMA
  p: ECB mode test with PDC
```

# 10. Document Revision History

| Doc. Rev. | Date | Comments |
|---|---|---|
| 42295B | 07/20145 | Updated title of application note and added list of supported devices |
| 42295A | 05/2014 | Initial document release |

**Atmel Corporation** 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**