www.qorvo.com
lpw.support@qorvo.com

# gpEncryption Reference Manual

## API Description

Version 2.10.2.0
November 15, 2021

# Contents

# Chapter 1

# Introduction

This document describes the API interface of the Encryption (gpEncryption) component.

gpEncryption supports the following functions:

- AES Encryption
- CCM Encryption
- CCM Decryption

The encrypted/decrypted result will be provided in the same (source) buffer.

For the AES encryption, the length of the data to be encrypted is fixed to 16 bytes. The AES encryption key size depends on the chip version:

- K8 and further supports; 128, 196 and 256 bits.
- Earlier versions of the chip supports 128 bits key.

The CCM encryption/decryption supports all security levels defined by IEEE Std 802.15.4 for CCM*. Also here the encryption key size is fixed to 128 bits.

# Chapter 2

# Module Documentation

## 2.1 AES Encryption

### Functions

- gpEncryption_Result_t gpEncryption_AESEncrypt (UInt8 *pInplaceBuffer, UInt8 *pAesKey, gpEncryption_AESOptions_t AESoptions)

  *Performs a synchronous AES Encryption.*

### 2.1.1 Detailed Description

### 2.1.2 Function Documentation

#### gpEncryption_AESEncrypt()

```
gpEncryption_Result_t gpEncryption_AESEncrypt (
            UInt8 * pInplaceBuffer,
            UInt8 * pAesKey,
            gpEncryption_AESOptions_t AESoptions )
```

The function will encrypt 16 bytes with the AES algorithm and return the result in place.

**Parameters**

| | |
|---|---|
| *pInplaceBuffer* | Pointer to the buffer of the 16 to be encrypted bytes. The encrypted result will be returned in the same buffer. |
| *pAesKey* | Pointer to the 16-byte key. This parameter is only used when gpEncryption_KeyIdKeyPtr is specified in the options parameter. When NULL is specified in combination with gpEncryption_KeyIdKeyPtr, 0 will be used as key value. |
| *options* | This parameter is an 8bit bitmask specifying the options: bits[6:0] specify a keyid defined by gpEncryption_KeyId_t; bit[7] indicates additional hardening. |

Returns

- gpEncryption_ResultSuccess
- gpEncryption_ResultBusy

## 2.2 CCM Encryption and Decryption

### Functions

- gpEncryption_Result_t gpEncryption_CCMEncrypt (gpEncryption_CCMOptions_t *pCCMOptions)
  *Performs a synchronous CCM Encryption.*
- gpEncryption_Result_t gpEncryption_CCMDecrypt (gpEncryption_CCMOptions_t *pCCMOptions)
  *Performs a synchronous CCM Decryption.*

### 2.2.1 Detailed Description

### 2.2.2 Function Documentation

#### gpEncryption_CCMDecrypt()

```
gpEncryption_Result_t gpEncryption_CCMDecrypt (
            gpEncryption_CCMOptions_t * pCCMOptions )
```

The function will decrypt the bytes with the CCM algorithm according to the specified options in the gpEncryption_CCMOptions structure.

**Parameters**

| | |
|---|---|
| *pCCMOptions* | Pointer to the gpEncryption_CCMOptions structure. |

Returns

- gpEncryption_ResultSuccess
- gpEncryption_ResultBusy
- gpEncryption_ResultInvalidParameter

#### gpEncryption_CCMEncrypt()

```
gpEncryption_Result_t gpEncryption_CCMEncrypt (
            gpEncryption_CCMOptions_t * pCCMOptions )
```

The function will encrypt the bytes with the CCM algorithm according to the specified options in the gpEncryption_CCMOptions structure.

**Parameters**

| | |
|---|---|
| *pCCMOptions* | Pointer to the gpEncryption_CCMOptions structure. |

Returns

- gpEncryption_ResultSuccess
- gpEncryption_ResultBusy

## 2.3 Initialization

### Functions

- void gpEncryption_Init (void)

  *Initializes the gpEncryption component.*

### 2.3.1 Detailed Description

### 2.3.2 Function Documentation

#### gpEncryption_Init()

```
void gpEncryption_Init (
            void  )
```

This function initializes the gpEncryption component. It should be called before calling any other function.

This primitive is typically called via the gpBaseComps_StackInit() method of the gpBaseComps component.

# Chapter 3

# Data Structure Documentation

## 3.1 gpEncryption_AESOptions_t Struct Reference

### Data Fields

- gpEncryption_AESKeyLen_t **keylen**
- UInt8 **options**

### 3.1.1 Detailed Description

**Parameters**

| *gpEncryption_AESKeyLen_t* | |
|---|---|
| *This* | parameter is an 8bit bitmask specifying the options: bits[6:0] specify the keyid to be used (see gpEncryption_API_Manual); bit[7] indicates additional hardening |

## 3.2 gpEncryption_CCMOptions_t Struct Reference

The gpEncryption_CCMOptions structure contains all the parameters for the CCM operations.

### Data Fields

- gpPd_Handle_t pdHandle

  *This field contains the pd (packet descriptor) identifier where the encryption/decryption will take place.*
- gpPd_Offset_t dataOffset

  *This field contains the offset in the pd indicating the start of the data (m-data in CCM∗).*
- UInt8 dataLength

  *This field contains the data length. This is the length of the data where the security operation will be performed (m-data in CCM∗).*
- gpPd_Offset_t auxOffset

  *This field contains the offset in the pd indicating the start of the auxiliary data (a-data in CCM∗).*
- UInt8 auxLength

  *This field contains the length of the auxiliary data (a-data in CCM∗).*

- UInt8 micLength

  *This field contains the expected MIC length.*
- UInt8 * pKey

  *This field contains the pointer to the encryption key. The key size is fixed to 16 bytes.*
- UInt8 * pNonce

  *This field contains the pointer to the nonce used for operation. The nonce length is fixed to 13 bytes.*

# Chapter 4

# File Documentation

## 4.1 gpEncryption.h File Reference

### Data Structures

- struct gpEncryption_AESOptions_t
- struct gpEncryption_CCMOptions_t

  *The gpEncryption_CCMOptions structure contains all the parameters for the CCM operations.*

### Macros

- #define gpEncryption_Hardened 0x80

  *Enable additional security hardening.*

### Functions

- gpEncryption_Result_t gpEncryption_AESEncrypt (UInt8 *pInplaceBuffer, UInt8 *pAesKey, gpEncryption_AESOptions_t AESoptions)

  *Performs a synchronous AES Encryption.*
- gpEncryption_Result_t gpEncryption_CCMEncrypt (gpEncryption_CCMOptions_t *pCCMOptions)

  *Performs a synchronous CCM Encryption.*
- gpEncryption_Result_t gpEncryption_CCMDecrypt (gpEncryption_CCMOptions_t *pCCMOptions)

  *Performs a synchronous CCM Decryption.*
- void gpEncryption_Init (void)

  *Initializes the gpEncryption component.*

### gpEncryption_AESKeyLen_t

- #define gpEncryption_AESKeyLen128 (128>>3)

  *128 bits key len*
- #define gpEncryption_AESKeyLen192 (192>>3)

  *192 bits key len*
- #define gpEncryption_AESKeyLen256 (256>>3)

  *256 bits key len*
- #define gpEncryption_AESKeyLenInv 0xFF

  *Identifier for invalid value.*

- #define **GP_ENCRYPTION_OPTIONS_IS_HARDENED**(id) (((id) & gpEncryption_Hardened) != 0)
- #define **GP_ENCRYPTION_OPTIONS_GET_KEYID**(id) (id & ∼gpEncryption_Hardened)
- #define **GP_ENCRYPTION_KEYID_IS_USER**(id) (id <= gpEncryption_KeyIdUserKey7)
- #define **GP_ENCRYPTION_KEYID_IS_PRODUCT**(id) (id == gpEncryption_KeyIdProductKey0 || id == gpEncryption_KeyIdProductKey1)
- #define **GP_ENCRYPTION_KEYID_IS_KEYPTR**(id) (id == gpEncryption_KeyIdKeyPtr)
- typedef UInt8 gpEncryption_AESKeyLen_t

  *gpEncryption_AESKeyLen_t possible values are: 16, 24 or 32 bytes.*

## gpEncryption_SecLevel_t

- #define gpEncryption_SecLevelNothing 0

  *Mode 0 No encryption, no MIC added.*
- #define gpEncryption_SecLevelMIC32 1

  *Mode 1 No encryption, 32 bit MIC added.*
- #define gpEncryption_SecLevelMIC64 2

  *Mode 2 No encryption, 64 bit MIC added.*
- #define gpEncryption_SecLevelMIC128 3

  *Mode 3 No encryption, 128 bit MIC added.*
- #define gpEncryption_SecLevelENC 4

  *Mode 4 Encryption of payload, no MIC added.*
- #define gpEncryption_SecLevelENC_MIC32 5

  *Mode 5 Encryption of payload, 32 bit MIC added.*
- #define gpEncryption_SecLevelENC_MIC64 6

  *Mode 6 Encryption of payload, 64 bit MIC added.*
- #define gpEncryption_SecLevelENC_MIC128 7

  *Mode 7 Encryption of payload, 128 bit MIC added.*
- #define GP_ENCRYPTION_SECLEVEL2MICLENGTH(secLevel) (((secLevel&0x03)∗4)==12?16:((secLevel&0

  *Convert security level into mic length.*
- typedef UInt8 gpEncryption_SecLevel_t

  *The gpEncryption_SecLevel_t type defines the IEEE Std 802.15.4 (2006) security level.*

## gpEncryption_Result_t

- #define gpEncryption_ResultSuccess 0x0

  *The function returned successful.*
- #define gpEncryption_ResultInvalidParameter 0x5

  *An invalid parameter was given as a parameter to this function.*
- #define gpEncryption_ResultBusy 0x7

  *The GP chip is busy.*
- #define gpEncryption_KeyIdUserKey0 0x00

  *User key identifiers.*
- #define **gpEncryption_KeyIdUserKey1** 0x01
- #define **gpEncryption_KeyIdUserKey2** 0x02
- #define **gpEncryption_KeyIdUserKey3** 0x03
- #define **gpEncryption_KeyIdUserKey4** 0x04
- #define **gpEncryption_KeyIdUserKey5** 0x05

- #define **gpEncryption_KeyIdUserKey6** 0x06
- #define **gpEncryption_KeyIdUserKey7** 0x07
- #define gpEncryption_KeyIdProductKey0 0x50

  *Product key identifiers.*
- #define **gpEncryption_KeyIdProductKey1** 0x51
- #define gpEncryption_KeyIdKeyPtr 0x7E

  *Unspecified key identifier.*
- #define gpEncryption_KeyIdUnspecified 0x7F

  *Unspecified key identifier.*
- typedef UInt8 gpEncryption_Result_t

  *The gpEncryption_Result_t type defines the result of various encryption functions.*
- typedef UInt8 **gpEncryption_KeyId_t**