

AC784xx\_DFP CSE

9.1.0

Generated by Doxygen 1.8.13

# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>2</b>
2.1	File List . . . . .	2
<b>3</b>	<b>Class Documentation</b>	<b>3</b>
3.1	Cse_InputKeyInfoType Struct Reference . . . . .	3
3.1.1	Detailed Description . . . . .	3
3.1.2	Member Data Documentation . . . . .	3
3.1.2.1	AuthKey . . . . .	3
3.1.2.2	AuthKeyId . . . . .	4
3.1.2.3	Count . . . . .	4
3.1.2.4	NewKey . . . . .	4
3.1.2.5	NewKeyAttr . . . . .	4
3.1.2.6	NewKeyId . . . . .	4
3.2	Cse_InputOutputType Struct Reference . . . . .	5
3.2.1	Detailed Description . . . . .	5
3.2.2	Member Data Documentation . . . . .	5
3.2.2.1	Ciphertext . . . . .	5
3.2.2.2	CiphertextLength . . . . .	5
3.2.2.3	Iv . . . . .	6
3.2.2.4	Mac . . . . .	6
3.2.2.5	MacLength . . . . .	6
3.2.2.6	Message . . . . .	6

3.2.2.7	MessageLength	6
3.2.2.8	Plaintext	7
3.2.2.9	PlaintextLength	7
3.2.2.10	VerifyStatus	7
3.3	Cse_SheKeyInfoType Struct Reference	7
3.3.1	Detailed Description	7
3.3.2	Member Data Documentation	8
3.3.2.1	M1	8
3.3.2.2	M2	8
3.3.2.3	M3	8
3.3.2.4	M4	8
3.3.2.5	M5	8
<b>4</b>	<b>File Documentation</b>	<b>9</b>
4.1	AC784xx_API_Reference_Manual_CSE.pdf File Reference	9
4.2	AC784xx_Cse_Reg.c File Reference	9
4.2.1	Detailed Description	9
4.3	AC784xx_Cse_Reg.h File Reference	9
4.3.1	Detailed Description	9
4.4	Cse_Hal.c File Reference	9
4.4.1	Detailed Description	9
4.5	Cse_Hal.h File Reference	10
4.5.1	Detailed Description	12
4.5.2	Macro Definition Documentation	12
4.5.2.1	MATCH_KEYATTR	12
4.5.3	Enumeration Type Documentation	12
4.5.3.1	Cse_BootMode	12
4.5.3.2	Cse_BootStatus	13
4.5.3.3	Cse_KeyAttr	13
4.5.3.4	Cse_KeyId	13
4.5.4	Function Documentation	14
4.5.4.1	CSE_Hal_CalcM1ToM5()	14

4.5.4.2	CSE_Hal_CancelCommand()	14
4.5.4.3	CSE_Hal_DecryptCBC()	15
4.5.4.4	CSE_Hal_DecryptECB()	15
4.5.4.5	CSE_Hal_Deinit()	16
4.5.4.6	CSE_Hal_EncryptCBC()	16
4.5.4.7	CSE_Hal_EncryptECB()	17
4.5.4.8	CSE_Hal_ExtendRNGSeed()	17
4.5.4.9	CSE_Hal_GenerateMAC()	18
4.5.4.10	CSE_Hal_GenerateMACAddrMode()	18
4.5.4.11	CSE_Hal_GenerateRnd()	19
4.5.4.12	CSE_Hal_GetAsyncCmdStatus()	19
4.5.4.13	CSE_Hal_GetBFNStatus()	20
4.5.4.14	CSE_Hal_GetBINStatus()	20
4.5.4.15	CSE_Hal_GetBOKStatus()	20
4.5.4.16	CSE_Hal_GetBSYStatus()	20
4.5.4.17	CSE_Hal_GetID()	21
4.5.4.18	CSE_Hal_GetRamKey()	21
4.5.4.19	CSE_Hal_GetSBStatus()	22
4.5.4.20	CSE_Hal_GetStatus()	22
4.5.4.21	CSE_Hal_Init()	22
4.5.4.22	CSE_Hal_InstallCallback()	22
4.5.4.23	CSE_Hal_LoadKey()	23
4.5.4.24	CSE_Hal_LoadPlainKey()	23
4.5.4.25	CSE_Hal_MPCompress()	24
4.5.4.26	CSE_Hal_ResetKey()	24
4.5.4.27	CSE_Hal_SetSecureBootMode()	25
4.5.4.28	CSE_Hal_SetSecureBootStatus()	25
4.5.4.29	CSE_Hal_VerifyMAC()	26
4.5.4.30	CSE_Hal_VerifyMACAddrMode()	26

# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Cse_InputKeyInfoType</a>	
Input Key info content . . . . .	3
<a href="#">Cse_InputOutputType</a>	
Cse message input/output . . . . .	5
<a href="#">Cse_SheKeyInfoType</a>	
Key info content, eg. M1~M5 . . . . .	7

# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AC784xx_API_Reference_Manual_CSE.pdf</a>	9
<a href="#">AC784xx_Cse_Reg.c</a>	
This file provides cse hardware access functions	9
<a href="#">AC784xx_Cse_Reg.h</a>	
This file provides CSE hardware integration functions	9
<a href="#">Cse_Hal.c</a>	
This file provides CSE integration functions	9
<a href="#">Cse_Hal.h</a>	
This file provides CSE integration functions interface	10

## Chapter 3

# Class Documentation

### 3.1 Cse\_InputKeyInfoType Struct Reference

Input Key info content.

```
#include <Cse_Hal.h>
```

#### Public Attributes

- uint8 \* [AuthKey](#)
- [Cse\\_KeyId](#) AuthKeyId
- uint8 \* [NewKey](#)
- [Cse\\_KeyId](#) NewKeyId
- [Cse\\_KeyAttr](#) NewKeyAttr
- uint32 [Count](#)

#### 3.1.1 Detailed Description

Input Key info content.

Definition at line 156 of file Cse\_Hal.h.

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 AuthKey

```
uint8* Cse_InputKeyInfoType::AuthKey
```

authorization key.

Definition at line 158 of file Cse\_Hal.h.

### 3.1.2.2 AuthKeyId

`Cse_KeyId Cse_InputKeyInfoType::AuthKeyId`

authorization key id.

Definition at line 159 of file Cse\_Hal.h.

### 3.1.2.3 Count

`uint32 Cse_InputKeyInfoType::Count`

the key to be update count value.

Definition at line 163 of file Cse\_Hal.h.

### 3.1.2.4 NewKey

`uint8* Cse_InputKeyInfoType::NewKey`

the new key will update to slot.

Definition at line 160 of file Cse\_Hal.h.

### 3.1.2.5 NewKeyAttr

`Cse_KeyAttr Cse_InputKeyInfoType::NewKeyAttr`

the new key Attr.

Definition at line 162 of file Cse\_Hal.h.

### 3.1.2.6 NewKeyId

`Cse_KeyId Cse_InputKeyInfoType::NewKeyId`

the new key id.

Definition at line 161 of file Cse\_Hal.h.

The documentation for this struct was generated from the following file:

- [Cse\\_Hal.h](#)



## 3.2 Cse\_InputOutputType Struct Reference

Cse message input/output.

```
#include <Cse_Hal.h>
```

### Public Attributes

- uint8 \* [Plaintext](#)
- uint32 [PlaintextLength](#)
- uint8 \* [Ciphertext](#)
- uint32 [CiphertextLength](#)
- uint8 \* [Iv](#)
- uint8 \* [Message](#)
- uint32 [MessageLength](#)
- uint8 \* [Mac](#)
- uint32 [MacLength](#)
- boolean \* [VerifyStatus](#)

### 3.2.1 Detailed Description

Cse message input/output.

Definition at line 127 of file Cse\_Hal.h.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 Ciphertext

```
uint8* Cse_InputOutputType::Ciphertext
```

pointer input or output ciphertext buf.

Definition at line 131 of file Cse\_Hal.h.

#### 3.2.2.2 CiphertextLength

```
uint32 Cse_InputOutputType::CiphertextLength
```

ciphertext length, align 16 bytes.

Definition at line 132 of file Cse\_Hal.h.

### 3.2.2.3 Iv

```
uint8* Cse_InputOutputType::Iv
```

the IV of the command in execution (using CBC mode)

Definition at line 133 of file Cse\_Hal.h.

### 3.2.2.4 Mac

```
uint8* Cse_InputOutputType::Mac
```

input or output Mac value buffer.

Definition at line 136 of file Cse\_Hal.h.

### 3.2.2.5 MacLength

```
uint32 Cse_InputOutputType::MacLength
```

input or output Mac value length.

Definition at line 137 of file Cse\_Hal.h.

### 3.2.2.6 Message

```
uint8* Cse_InputOutputType::Message
```

input Message buf for Generate/Verify CMAC.

Definition at line 134 of file Cse\_Hal.h.

### 3.2.2.7 MessageLength

```
uint32 Cse_InputOutputType::MessageLength
```

the Message length for Generate/Verify CMAC.

Definition at line 135 of file Cse\_Hal.h.

#### 3.2.2.8 Plaintext

```
uint8* Cse_InputOutputType::Plaintext
```

pointer input or output plaintext buf.

Definition at line 129 of file Cse\_Hal.h.

#### 3.2.2.9 PlaintextLength

```
uint32 Cse_InputOutputType::PlaintextLength
```

plaintext length, align 16 bytes.

Definition at line 130 of file Cse\_Hal.h.

#### 3.2.2.10 VerifyStatus

```
boolean* Cse_InputOutputType::VerifyStatus
```

the verify mac status result

Definition at line 138 of file Cse\_Hal.h.

The documentation for this struct was generated from the following file:

- [Cse\\_Hal.h](#)

### 3.3 Cse\_SheKeyInfoType Struct Reference

Key info content, eg. M1~M5.

```
#include <Cse_Hal.h>
```

#### Public Attributes

- uint8 \* [M1](#)
- uint8 \* [M2](#)
- uint8 \* [M3](#)
- uint8 \* [M4](#)
- uint8 \* [M5](#)

#### 3.3.1 Detailed Description

Key info content, eg. M1~M5.

Definition at line 144 of file Cse\_Hal.h.

### 3.3.2 Member Data Documentation

#### 3.3.2.1 M1

```
uint8* Cse_SheKeyInfoType::M1
```

M1: pointer to the 128-bit M1 message buffer.

Definition at line 146 of file Cse\_Hal.h.

#### 3.3.2.2 M2

```
uint8* Cse_SheKeyInfoType::M2
```

M2: pointer to the 256-bit M2 message buffer.

Definition at line 147 of file Cse\_Hal.h.

#### 3.3.2.3 M3

```
uint8* Cse_SheKeyInfoType::M3
```

M3: pointer to the 128-bit M3 message buffer.

Definition at line 148 of file Cse\_Hal.h.

#### 3.3.2.4 M4

```
uint8* Cse_SheKeyInfoType::M4
```

M4: pointer to the 256-bit M4 message buffer.

Definition at line 149 of file Cse\_Hal.h.

#### 3.3.2.5 M5

```
uint8* Cse_SheKeyInfoType::M5
```

M5: pointer to the 128-bit M5 message buffer.

Definition at line 150 of file Cse\_Hal.h.

The documentation for this struct was generated from the following file:

- [Cse\\_Hal.h](#)

## Chapter 4

# File Documentation

### 4.1 AC784xx\_API\_Reference\_Manual\_CSE.pdf File Reference

### 4.2 AC784xx\_Cse\_Reg.c File Reference

This file provides cse hardware access functions.

#### 4.2.1 Detailed Description

This file provides cse hardware access functions.

### 4.3 AC784xx\_Cse\_Reg.h File Reference

This file provides CSE hardware integration functions.

#### 4.3.1 Detailed Description

This file provides CSE hardware integration functions.

### 4.4 Cse\_Hal.c File Reference

This file provides CSE integration functions.

#### 4.4.1 Detailed Description

This file provides CSE integration functions.

## 4.5 Cse\_Hal.h File Reference

This file provides CSE integration functions interface.

```
#include "Device_Register.h"
#include "System_AC784xx.h"
```

### Classes

- struct [Cse\\_InputOutputType](#)  
*Cse message input/output.*
- struct [Cse\\_SheKeyInfoType](#)  
*Key info content, eg. M1~M5.*
- struct [Cse\\_InputKeyInfoType](#)  
*Input Key info content.*

### Macros

- #define [MATCH\\_KEYATTR\(x\)](#) (x)

### Enumerations

- enum [Cse\\_KeyId](#) {  
[CSE\\_SECRET\\_KEY](#) = 0x0U, [CSE\\_MASTER\\_ECU](#), [CSE\\_BOOT\\_MAC\\_KEY](#), [CSE\\_BOOT\\_MAC](#),  
[CSE\\_KEY\\_1](#), [CSE\\_KEY\\_2](#), [CSE\\_KEY\\_3](#), [CSE\\_KEY\\_4](#),  
[CSE\\_KEY\\_5](#), [CSE\\_KEY\\_6](#), [CSE\\_KEY\\_7](#), [CSE\\_KEY\\_8](#),  
[CSE\\_KEY\\_9](#), [CSE\\_KEY\\_10](#), [CSE\\_RAM\\_KEY](#) = 0xFU, [CSE\\_KEY\\_11](#) = 0x14U,  
[CSE\\_KEY\\_12](#), [CSE\\_KEY\\_13](#), [CSE\\_KEY\\_14](#), [CSE\\_KEY\\_15](#),  
[CSE\\_KEY\\_16](#), [CSE\\_KEY\\_17](#) }  
*Specify the KeyID to be used to implement the requested cryptographic operation.*
- enum [Cse\\_BootMode](#) { [CSE\\_BOOT\\_STRICT](#), [CSE\\_BOOT\\_SERIAL](#), [CSE\\_BOOT\\_PARALLEL](#), [CSE\\_BOOT\\_NO↵  
DEFINE](#) }  
*Specifies the boot type for the BOOT\_DEFINE command.*
- enum [Cse\\_BootStatus](#) { [CSE\\_BOOT\\_OK](#), [CSE\\_BOOT\\_FAILURE](#) }  
*secure boot status.*
- enum [Cse\\_KeyAttr](#) {  
[KEY\\_ATTR\\_NONE](#) = 0x00U, [KEY\\_ATTR\\_VERIFY\\_ONLY](#) = 0x01U, [KEY\\_ATTR\\_WILDCARD](#) = 0x02U, [KEY\\_ATTR↵  
TR\\_KEY\\_USAGE](#) = 0x04U,  
[KEY\\_ATTR\\_DEBUG\\_PROTECT](#) = 0x08U, [KEY\\_ATTR\\_BOOT\\_PROTECT](#) = 0x10U, [KEY\\_ATTR\\_WRITE\\_PROT↵  
ECT](#) = 0x20U }  
*the attributes for Keys.*

## Functions

- Hal\_StatusType [CSE\\_Hal\\_Init](#) (void)  
*Initialize CSE module.*
- Hal\_StatusType [CSE\\_Hal\\_Deinit](#) (void)  
*Uninitialize CSE module.*
- Hal\_StatusType [CSE\\_Hal\\_EncryptECB](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId, uint32 TimeoutUs)  
*Performs the AES-128 encryption in ECB mode of the input plain text with the Key ID and returns the cipher text.*
- Hal\_StatusType [CSE\\_Hal\\_DecryptECB](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId, uint32 TimeoutUs)  
*Performs the AES-128 decryption in ECB mode of the input cipher text with the Key ID and returns the plain text.*
- Hal\_StatusType [CSE\\_Hal\\_EncryptCBC](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId, uint32 TimeoutUs)  
*Performs the AES-128 encryption in CBC mode of the input plain text with the Key ID and returns the cipher text.*
- Hal\_StatusType [CSE\\_Hal\\_DecryptCBC](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId, uint32 TimeoutUs)  
*Performs the AES-128 decryption in CBC mode of the input cipher text with the Key ID and returns the plain text.*
- Hal\_StatusType [CSE\\_Hal\\_GenerateMAC](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId, uint32 TimeoutUs)  
*Generate the MAC of a given message using CMAC with AES-128.*
- Hal\_StatusType [CSE\\_Hal\\_VerifyMAC](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId, uint32 TimeoutUs)  
*Verifies the MAC of a given message using CMAC with AES-128.*
- Hal\_StatusType [CSE\\_Hal\\_GetRamKey](#) (const [Cse\\_SheKeyInfoType](#) \*Ptr)  
*Exports the RAM\_KEY into a format protected by SECRET\_KEY.*
- Hal\_StatusType [CSE\\_Hal\\_GenerateRnd](#) (uint8 \*Rnd)  
*Extends the seed of the PRNG.*
- Hal\_StatusType [CSE\\_Hal\\_GenerateMACAddrMode](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId)  
*Generate the MAC of a given message(locate in FLASH) using CMAC with AES-128.*
- Hal\_StatusType [CSE\\_Hal\\_VerifyMACAddrMode](#) (const [Cse\\_InputOutputType](#) \*Ptr, [Cse\\_KeyId](#) KeyId)  
*Verifies the MAC of a given message(locate in FLASH) using CMAC with AES-128.*
- Hal\_StatusType [CSE\\_Hal\\_ExtendRNGSeed](#) (const uint8 \*Entropy)  
*Extends the seed of the PRNG.*
- Hal\_StatusType [CSE\\_Hal\\_GetID](#) (const uint8 \*Challenge, uint8 \*Uid, uint8 \*Sreg, uint8 \*Mac)  
*Get the UID from CSE module.*
- Hal\_StatusType [CSE\\_Hal\\_MPCompress](#) (const uint8 \*Msg, uint16 MsgLen, uint8 \*MpCompress, uint32 TimeoutUs)  
*Compresses the given messages.*
- Hal\_StatusType [CSE\\_Hal\\_SetSecureBootStatus](#) ([Cse\\_BootStatus](#) Status)  
*Mark failure/success boot verification.*
- Hal\_StatusType [CSE\\_Hal\\_SetSecureBootMode](#) (uint32 BootSize, [Cse\\_BootMode](#) BootMode)  
*Define bootcode size and secure boot type.*
- void [CSE\\_Hal\\_CancelCommand](#) (void)  
*Cancels a previously launched asynchronous command.*
- void [CSE\\_Hal\\_InstallCallback](#) (const Hal\_CallbackType Func, void \*Args)  
*Installs a callback function for CSE driver.*
- Hal\_StatusType [CSE\\_Hal\\_GetAsyncCmdStatus](#) (void)  
*Asynchronously get operation status.*
- Hal\_StatusType [CSE\\_Hal\\_ResetKey](#) (const uint8 \*MasterKey)  
*Erase all keys stored in the CSE.*
- Hal\_StatusType [CSE\\_Hal\\_LoadKey](#) ([Cse\\_KeyId](#) KeyId, const [Cse\\_SheKeyInfoType](#) \*InPtr)  
*Updates an internal key to CSE module.*
- Hal\_StatusType [CSE\\_Hal\\_LoadPlainKey](#) (const uint8 \*PlainKey)  
*Updates the RAM key memory slot with a 128-bit plaintext.*
- Hal\_StatusType [CSE\\_Hal\\_CalcM1ToM5](#) (const [Cse\\_InputKeyInfoType](#) \*InPtr, const [Cse\\_SheKeyInfoType](#) \*OutPtr)  
*Calculate M1 To M5.*
- uint32 [CSE\\_Hal\\_GetStatus](#) (void)  
*Returns the content of the status register.*
- uint32 [CSE\\_Hal\\_GetBOKStatus](#) (void)

- Returns the content of the CSESTAT BOK bit value.*
  - uint32 [CSE\\_Hal\\_GetBFNStatus](#) (void)
- Returns the content of the CSESTAT BFN bit value.*
  - uint32 [CSE\\_Hal\\_GetBINStatus](#) (void)
- Returns the content of the CSESTAT BIN bit value.*
  - uint32 [CSE\\_Hal\\_GetSBStatus](#) (void)
- Returns the content of the CSESTAT SB bit value.*
  - uint32 [CSE\\_Hal\\_GetBSYStatus](#) (void)
- Returns the content of the CSESTAT BSY bit value.*

### 4.5.1 Detailed Description

This file provides CSE integration functions interface.

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 MATCH\_KEYATTR

```
#define MATCH_KEYATTR(  
    x ) (x)
```

Definition at line 122 of file Cse\_Hal.h.

### 4.5.3 Enumeration Type Documentation

#### 4.5.3.1 Cse\_BootMode

```
enum Cse\_BootMode
```

Specifies the boot type for the BOOT\_DEFINE command.

Enumerator

CSE_BOOT_STRICT	strict secure boot mode
CSE_BOOT_SERIAL	serial secure boot mode
CSE_BOOT_PARALLEL	parallel secure boot mode
CSE_BOOT_NODEFINE	not boot define master key not init yet or non-cse enable part

Definition at line 88 of file Cse\_Hal.h.



#### 4.5.3.2 Cse\_BootStatus

enum `Cse_BootStatus`

secure boot status.

##### Enumerator

CSE_BOOT_OK	secure boot success status
CSE_BOOT_FAILURE	secure boot fail status

Definition at line 99 of file Cse\_Hal.h.

#### 4.5.3.3 Cse\_KeyAttr

enum `Cse_KeyAttr`

the attributes for Keys.

##### Enumerator

KEY_ATTR_NONE	the key empty attribute
KEY_ATTR_VERIFY_ONLY	the key verify only attribute
KEY_ATTR_WILDCARD	the key wildcard attribute
KEY_ATTR_KEY_USAGE	the key key usage attribute
KEY_ATTR_DEBUG_PROTECT	the key debug protect attribute
KEY_ATTR_BOOT_PROTECT	the key boot protect attribute
KEY_ATTR_WRITE_PROTECT	the key write protect attribute

Definition at line 108 of file Cse\_Hal.h.

#### 4.5.3.4 Cse\_KeyId

enum `Cse_KeyId`

Specify the KeyID to be used to implement the requested cryptographic operation.

##### Enumerator

CSE_SECRET_KEY	secret key
CSE_MASTER_ECU	master key for authorization key to use
CSE_BOOT_MAC_KEY	boot mac key for calculate boot mac value
CSE_BOOT_MAC	store boot mac value
CSE_KEY_1	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_2	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_3	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_4	user key for encryption/decryption/generateMac/verifyMAC etc

## Enumerator

CSE_KEY_5	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_6	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_7	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_8	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_9	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_10	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_RAM_KEY	ram key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_11	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_12	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_13	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_14	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_15	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_16	user key for encryption/decryption/generateMac/verifyMAC etc
CSE_KEY_17	user key for encryption/decryption/generateMac/verifyMAC etc

Definition at line 59 of file Cse\_Hal.h.

## 4.5.4 Function Documentation

### 4.5.4.1 CSE\_Hal\_CalcM1ToM5()

```
Hal_StatusType CSE_Hal_CalcM1ToM5 (
    const Cse_InputKeyInfoType * InPtr,
    const Cse_SheKeyInfoType * OutPtr )
```

Calculate M1 To M5.

#### Note

Function ID: DES\_CSE\_API\_044

#### Parameters

in	<i>InPtr</i>	the key content to be updated.
out	<i>OutPtr</i>	store M1~M5 value.

#### Returns

Hal\_StatusType

### 4.5.4.2 CSE\_Hal\_CancelCommand()

```
void CSE_Hal_CancelCommand (
    void )
```

Cancels a previously launched asynchronous command.

**Note**

Function ID: DES\_CSE\_API\_019

**Returns**

void

**4.5.4.3 CSE\_Hal\_DecryptCBC()**

```
Hal_StatusType CSE_Hal_DecryptCBC (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId,
    uint32 TimeoutUs )
```

Performs the AES-128 decryption in CBC mode of the input cipher text with the Key ID and returns the plain text.

**Note**

Function ID: DES\_CSE\_API\_006

**Parameters**

in	<i>Ptr</i>	a structure containing input and output. eg.plaintext/ciphertext.
in	<i>KeyId</i>	the index of the key used in cse command.
in	<i>TimeoutUs</i>	timeout value for sync flow.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Value: operation was fail

**4.5.4.4 CSE\_Hal\_DecryptECB()**

```
Hal_StatusType CSE_Hal_DecryptECB (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId,
    uint32 TimeoutUs )
```

Performs the AES-128 decryption in ECB mode of the input cipher text with the Key ID and returns the plain text.

**Note**

Function ID: DES\_CSE\_API\_004

**Parameters**

in	<i>Ptr</i>	a structure containing input and output. eg.plaintext/ciphertext.
in	<i>KeyId</i>	the index of the key used in cse command.
in	<i>TimeoutUs</i>	timeout value for sync flow.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Value: operation was fail

**4.5.4.5 CSE\_Hal\_Deinit()**

```
Hal_StatusType CSE_Hal_Deinit (  
    void )
```

Uninitialize CSE module.

**Note**

Function ID: DES\_CSE\_API\_002

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Value: operation was fail

**4.5.4.6 CSE\_Hal\_EncryptCBC()**

```
Hal_StatusType CSE_Hal_EncryptCBC (  
    const Cse_InputOutputType * Ptr,  
    Cse_KeyId KeyId,  
    uint32 TimeoutUs )
```

Performs the AES-128 encryption in CBC mode of the input plain text with the Key ID and returns the cipher text.

**Note**

Function ID: DES\_CSE\_API\_005

**Parameters**

in	<i>Ptr</i>	a structure containing input and output. eg.plaintext/ciphertext.
in	<i>KeyId</i>	the index of the key used in cse command.
in	<i>TimeoutUs</i>	timeout value for sync flow.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.7 CSE\_Hal\_EncryptECB()**

```
Hal_StatusType CSE_Hal_EncryptECB (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId,
    uint32 TimeoutUs )
```

Performs the AES-128 encryption in ECB mode of the input plain text with the Key ID and returns the cipher text.

**Note**

Function ID: DES\_CSE\_API\_003

**Parameters**

in	<i>Ptr</i>	a structure contatining input and output. eg.plaintext/ciphertext.
in	<i>KeyId</i>	the index of the key used in cse command.
in	<i>TimeoutUs</i>	timeout value for sync flow.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.8 CSE\_Hal\_ExtendRNGSeed()**

```
Hal_StatusType CSE_Hal_ExtendRNGSeed (
    const uint8 * Entropy )
```

Extends the seed of the PRNG.

**Note**

Function ID: DES\_CSE\_API\_014

**Parameters**

in	<i>Entropy</i>	pointer to a 128-bit buffer containing the entropy
----	----------------	--

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.9 CSE\_Hal\_GenerateMAC()**

```
Hal_StatusType CSE_Hal_GenerateMAC (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId,
    uint32 TimeoutUs )
```

Generate the MAC of a given message using CMAC with AES-128.

**Note**

Function ID: DES\_CSE\_API\_007

**Parameters**

in	<i>Ptr</i>	a structure contatining input and output. eg.message/cmac.
in	<i>KeyId</i>	the index of the key used in cse command.
in	<i>TimeoutUs</i>	timeout value for sync flow.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.10 CSE\_Hal\_GenerateMACAddrMode()**

```
Hal_StatusType CSE_Hal_GenerateMACAddrMode (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId )
```

Generate the MAC of a given message(locate in FLASH) using CMAC with AES-128.

**Note**

Function ID: DES\_CSE\_API\_012

**Parameters**

in	<i>Ptr</i>	a structure contatining input and output, Message and Addr must be 16bytes align. eg.message/cmac.
in	<i>Key↔ Id</i>	the index of the key used in cse command.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.11 CSE\_Hal\_GenerateRnd()**

```
Hal_StatusType CSE_Hal_GenerateRnd (
    uint8 * Rnd )
```

Extends the seed of the PRNG.

**Note**

Function ID: DES\_CSE\_API\_011

**Parameters**

out	<i>Rnd</i>	pointer to a 128-bit buffer containing the PRNG value
-----	------------	---

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.12 CSE\_Hal\_GetAsyncCmdStatus()**

```
Hal_StatusType CSE_Hal_GetAsyncCmdStatus (
    void )
```

Asynchronously get operation status.

**Note**

Function ID: DES\_CSE\_API\_021

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

#### 4.5.4.13 CSE\_Hal\_GetBFNStatus()

```
uint32 CSE_Hal_GetBFNStatus (
    void )
```

Returns the content of the CSESTAT BFN bit value.

##### Returns

Value of the status register

#### 4.5.4.14 CSE\_Hal\_GetBINStatus()

```
uint32 CSE_Hal_GetBINStatus (
    void )
```

Returns the content of the CSESTAT BIN bit value.

##### Returns

Value of the status register

#### 4.5.4.15 CSE\_Hal\_GetBOKStatus()

```
uint32 CSE_Hal_GetBOKStatus (
    void )
```

Returns the content of the CSESTAT BOK bit value.

##### Returns

Value of the status register

#### 4.5.4.16 CSE\_Hal\_GetBSYStatus()

```
uint32 CSE_Hal_GetBSYStatus (
    void )
```

Returns the content of the CSESTAT BSY bit value.

##### Parameters

in	none	
----	------	--



**Returns**

Value of the status register

**4.5.4.17 CSE\_Hal\_GetID()**

```
Hal_StatusType CSE_Hal_GetID (
    const uint8 * Challenge,
    uint8 * Uid,
    uint8 * Sreg,
    uint8 * Mac )
```

Get the UID from CSE module.

**Note**

Function ID: DES\_CSE\_API\_015

**Parameters**

in	<i>Challenge</i>	pointer to the 128-bit buffer containing challenge data
out	<i>Uid</i>	pointer to 120 bit buffer containing UID data
out	<i>Sreg</i>	Value of the status register.
out	<i>Mac</i>	pointer to the 128 bit buffer containing MAC data

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.18 CSE\_Hal\_GetRamKey()**

```
Hal_StatusType CSE_Hal_GetRamKey (
    const Cse_SheKeyInfoType * Ptr )
```

Exports the RAM\_KEY into a format protected by SECRET\_KEY.

**Note**

Function ID: DES\_CSE\_API\_010

**Parameters**

out	<i>Ptr</i>	store M1~M5 value.
-----	------------	--------------------

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.19 CSE\_Hal\_GetSBStatus()**

```
uint32 CSE_Hal_GetSBStatus (
    void )
```

Returns the content of the CSESTAT SB bit value.

**Returns**

Value of the status register

**4.5.4.20 CSE\_Hal\_GetStatus()**

```
uint32 CSE_Hal_GetStatus (
    void )
```

Returns the content of the status register.

**Returns**

Value of the status register

**4.5.4.21 CSE\_Hal\_Init()**

```
Hal_StatusType CSE_Hal_Init (
    void )
```

Initialize CSE module.

**Note**

Function ID: DES\_CSE\_API\_001

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.22 CSE\_Hal\_InstallCallback()**

```
void CSE_Hal_InstallCallback (
    const Hal_CallbackType Func,
    void * Args )
```

Installs a callback function for CSE driver.

**Note**

Function ID: DES\_CSE\_API\_020

## Parameters

in	<i>Func</i>	The function to be invoked
in	<i>Args</i>	The parameter to be passed to the callback function

## Returns

none

## 4.5.4.23 CSE\_Hal\_LoadKey()

```
Hal_StatusType CSE_Hal_LoadKey (
    Cse_KeyId KeyId,
    const Cse_SheKeyInfoType * InPtr )
```

Updates an internal key to CSE module.

## Note

Function ID: DES\_CSE\_API\_023

## Parameters

in	<i>Key↔ Id</i>	the key to be updated
in	<i>InPtr</i>	store M1~M5 value.

## Returns

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

## 4.5.4.24 CSE\_Hal\_LoadPlainKey()

```
Hal_StatusType CSE_Hal_LoadPlainKey (
    const uint8 * PlainKey )
```

Updates the RAM key memory slot with a 128-bit plaintext.

## Note

Function ID: DES\_CSE\_API\_024

## Parameters

in	<i>PlainKey</i>	pointer to the 128-bit plain text
----	-----------------	-----------------------------------

## Returns

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

## 4.5.4.25 CSE\_Hal\_MPCompress()

```
Hal_StatusType CSE_Hal_MPCompress (  
    const uint8 * Msg,  
    uint16 MsgLen,  
    uint8 * MpCompress,  
    uint32 TimeoutUs )
```

Compresses the given messages.

## Note

Function ID: DES\_CSE\_API\_016

## Parameters

in	<i>Msg</i>	pointer to the messages which will be compressed
in	<i>MsgLen</i>	number in bits of message
out	<i>MpCompress</i>	pointer to the 128 bit buffer storing the compressed data
in	<i>TimeoutUs</i>	timeout value in milliseconds

## Returns

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

## 4.5.4.26 CSE\_Hal\_ResetKey()

```
Hal_StatusType CSE_Hal_ResetKey (  
    const uint8 * MasterKey )
```

Erase all keys stored in the CSE.

## Note

Function ID: DES\_CSE\_API\_022

## Parameters

in	<i>MasterKey</i>	Master Key in chip.
----	------------------	---------------------

## Returns

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

## 4.5.4.27 CSE\_Hal\_SetSecureBootMode()

```
Hal_StatusType CSE_Hal_SetSecureBootMode (
    uint32 BootSize,
    Cse_BootMode BootMode )
```

Define bootcode size and secure boot type.

## Note

Function ID: DES\_CSE\_API\_018

## Parameters

in	<i>BootSize</i>	number of blocks of 128-bit data to check on boot
in	<i>BootMode</i>	secure boot type

## Returns

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

## 4.5.4.28 CSE\_Hal\_SetSecureBootStatus()

```
Hal_StatusType CSE_Hal_SetSecureBootStatus (
    Cse_BootStatus Status )
```

Mark failure/success boot verification.

## Note

Function ID: DES\_CSE\_API\_017

**Parameters**

in	<i>Status</i>	secure boot flow status
----	---------------	-------------------------

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.29 CSE\_Hal\_VerifyMAC()**

```
Hal_StatusType CSE_Hal_VerifyMAC (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId,
    uint32 TimeoutUs )
```

Verifies the MAC of a given message using CMAC with AES-128.

**Note**

Function ID: DES\_CSE\_API\_008

**Parameters**

in	<i>Ptr</i>	a structure contatining input and output. eg.message/cmac.
in	<i>KeyId</i>	the index of the key used in cse command.
in	<i>TimeoutUs</i>	timeout value for sync flow.

**Returns**

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

**4.5.4.30 CSE\_Hal\_VerifyMACAddrMode()**

```
Hal_StatusType CSE_Hal_VerifyMACAddrMode (
    const Cse_InputOutputType * Ptr,
    Cse_KeyId KeyId )
```

Verifies the MAC of a given message(locate in FLASH) using CMAC with AES-128.

**Note**

Function ID: DES\_CSE\_API\_013

## Parameters

in	<i>Ptr</i>	a structure contatining input and output. eg.message/cmac.
in	<i>Key↔ Id</i>	the index of the key used in cse command.

## Returns

operation status

- STATUS\_SUCCESS: operation was successful
- Other Valule: operation was fail

# Index

AC784xx\_API\_Reference\_Manual\_CSE.pdf, [9](#)  
AC784xx\_Cse\_Reg.c, [9](#)  
AC784xx\_Cse\_Reg.h, [9](#)  
AuthKey  
    Cse\_InputKeyInfoType, [3](#)  
AuthKeyId  
    Cse\_InputKeyInfoType, [3](#)  
  
CSE\_Hal\_CalcM1ToM5  
    Cse\_Hal.h, [14](#)  
CSE\_Hal\_CancelCommand  
    Cse\_Hal.h, [14](#)  
CSE\_Hal\_DecryptCBC  
    Cse\_Hal.h, [15](#)  
CSE\_Hal\_DecryptECB  
    Cse\_Hal.h, [15](#)  
CSE\_Hal\_Deinit  
    Cse\_Hal.h, [16](#)  
CSE\_Hal\_EncryptCBC  
    Cse\_Hal.h, [16](#)  
CSE\_Hal\_EncryptECB  
    Cse\_Hal.h, [17](#)  
CSE\_Hal\_ExtendRNGSeed  
    Cse\_Hal.h, [17](#)  
CSE\_Hal\_GenerateMACAddrMode  
    Cse\_Hal.h, [18](#)  
CSE\_Hal\_GenerateMAC  
    Cse\_Hal.h, [18](#)  
CSE\_Hal\_GenerateRnd  
    Cse\_Hal.h, [19](#)  
CSE\_Hal\_GetAsyncCmdStatus  
    Cse\_Hal.h, [19](#)  
CSE\_Hal\_GetBFNStatus  
    Cse\_Hal.h, [19](#)  
CSE\_Hal\_GetBINStatus  
    Cse\_Hal.h, [20](#)  
CSE\_Hal\_GetBOKStatus  
    Cse\_Hal.h, [20](#)  
CSE\_Hal\_GetBSYStatus  
    Cse\_Hal.h, [20](#)  
CSE\_Hal\_GetID  
    Cse\_Hal.h, [21](#)  
CSE\_Hal\_GetRamKey  
    Cse\_Hal.h, [21](#)  
CSE\_Hal\_GetSBStatus  
    Cse\_Hal.h, [22](#)  
CSE\_Hal\_GetStatus  
    Cse\_Hal.h, [22](#)  
CSE\_Hal\_Init  
    Cse\_Hal.h, [22](#)  
CSE\_Hal\_InstallCallback  
    Cse\_Hal.h, [22](#)  
  
CSE\_Hal\_LoadKey  
    Cse\_Hal.h, [23](#)  
CSE\_Hal\_LoadPlainKey  
    Cse\_Hal.h, [23](#)  
CSE\_Hal\_MPCompress  
    Cse\_Hal.h, [24](#)  
CSE\_Hal\_ResetKey  
    Cse\_Hal.h, [24](#)  
CSE\_Hal\_SetSecureBootMode  
    Cse\_Hal.h, [25](#)  
CSE\_Hal\_SetSecureBootStatus  
    Cse\_Hal.h, [25](#)  
CSE\_Hal\_VerifyMACAddrMode  
    Cse\_Hal.h, [26](#)  
CSE\_Hal\_VerifyMAC  
    Cse\_Hal.h, [26](#)  
Ciphertext  
    Cse\_InputOutputType, [5](#)  
CiphertextLength  
    Cse\_InputOutputType, [5](#)  
Count  
    Cse\_InputKeyInfoType, [4](#)  
Cse\_BootMode  
    Cse\_Hal.h, [12](#)  
Cse\_BootStatus  
    Cse\_Hal.h, [12](#)  
Cse\_Hal.c, [9](#)  
Cse\_Hal.h, [10](#)  
    CSE\_Hal\_CalcM1ToM5, [14](#)  
    CSE\_Hal\_CancelCommand, [14](#)  
    CSE\_Hal\_DecryptCBC, [15](#)  
    CSE\_Hal\_DecryptECB, [15](#)  
    CSE\_Hal\_Deinit, [16](#)  
    CSE\_Hal\_EncryptCBC, [16](#)  
    CSE\_Hal\_EncryptECB, [17](#)  
    CSE\_Hal\_ExtendRNGSeed, [17](#)  
    CSE\_Hal\_GenerateMACAddrMode, [18](#)  
    CSE\_Hal\_GenerateMAC, [18](#)  
    CSE\_Hal\_GenerateRnd, [19](#)  
    CSE\_Hal\_GetAsyncCmdStatus, [19](#)  
    CSE\_Hal\_GetBFNStatus, [19](#)  
    CSE\_Hal\_GetBINStatus, [20](#)  
    CSE\_Hal\_GetBOKStatus, [20](#)  
    CSE\_Hal\_GetBSYStatus, [20](#)  
    CSE\_Hal\_GetID, [21](#)  
    CSE\_Hal\_GetRamKey, [21](#)  
    CSE\_Hal\_GetSBStatus, [22](#)  
    CSE\_Hal\_GetStatus, [22](#)  
    CSE\_Hal\_Init, [22](#)  
    CSE\_Hal\_InstallCallback, [22](#)  
    CSE\_Hal\_LoadKey, [23](#)  
    CSE\_Hal\_LoadPlainKey, [23](#)



- CSE\_Hal\_MPCompress, [24](#)
- CSE\_Hal\_ResetKey, [24](#)
- CSE\_Hal\_SetSecureBootMode, [25](#)
- CSE\_Hal\_SetSecureBootStatus, [25](#)
- CSE\_Hal\_VerifyMACAddrMode, [26](#)
- CSE\_Hal\_VerifyMAC, [26](#)
- Cse\_BootMode, [12](#)
- Cse\_BootStatus, [12](#)
- Cse\_KeyAttr, [13](#)
- Cse\_KeyId, [13](#)
- MATCH\_KEYATTR, [12](#)
- Cse\_InputKeyInfoType, [3](#)
  - AuthKey, [3](#)
  - AuthKeyId, [3](#)
  - Count, [4](#)
  - NewKey, [4](#)
  - NewKeyAttr, [4](#)
  - NewKeyId, [4](#)
- Cse\_InputOutputType, [5](#)
  - Ciphertext, [5](#)
  - CiphertextLength, [5](#)
  - Iv, [5](#)
  - Mac, [6](#)
  - MacLength, [6](#)
  - Message, [6](#)
  - MessageLength, [6](#)
  - Plaintext, [6](#)
  - PlaintextLength, [7](#)
  - VerifyStatus, [7](#)
- Cse\_KeyAttr
  - Cse\_Hal.h, [13](#)
- Cse\_KeyId
  - Cse\_Hal.h, [13](#)
- Cse\_SheKeyInfoType, [7](#)
  - M1, [8](#)
  - M2, [8](#)
  - M3, [8](#)
  - M4, [8](#)
  - M5, [8](#)
- Iv
  - Cse\_InputOutputType, [5](#)
- M1
  - Cse\_SheKeyInfoType, [8](#)
- M2
  - Cse\_SheKeyInfoType, [8](#)
- M3
  - Cse\_SheKeyInfoType, [8](#)
- M4
  - Cse\_SheKeyInfoType, [8](#)
- M5
  - Cse\_SheKeyInfoType, [8](#)
- MATCH\_KEYATTR
  - Cse\_Hal.h, [12](#)
- Mac
  - Cse\_InputOutputType, [6](#)
- MacLength
  - Cse\_InputOutputType, [6](#)
- Message
  - Cse\_InputOutputType, [6](#)
- MessageLength
  - Cse\_InputOutputType, [6](#)
- NewKey
  - Cse\_InputKeyInfoType, [4](#)
- NewKeyAttr
  - Cse\_InputKeyInfoType, [4](#)
- NewKeyId
  - Cse\_InputKeyInfoType, [4](#)
- Plaintext
  - Cse\_InputOutputType, [6](#)
- PlaintextLength
  - Cse\_InputOutputType, [7](#)
- VerifyStatus
  - Cse\_InputOutputType, [7](#)