

# Secure Hash Algorithm (SHA) User Guide

Version 1.10 BETA

For use with SHA module versions 1.3 and above

**Date:** 02-Mar-2016 10:57

All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

---

# Table of Contents

---

System Overview	3
Introduction	3
Feature Check	4
Packages and Documents	4
Packages	4
Documents	4
Change History	5
Source File List	6
API Header File	6
System File	6
Platform Support Package (PSP) File	6
Version File	6
Application Programming Interface	7
sha1_init_fn	7
sha256_init_fn	8
sha384_init_fn	9
sha512_init_fn	10
Hash Output Sizes	11
Error Codes	12
Integration	13
PSP Porting	13

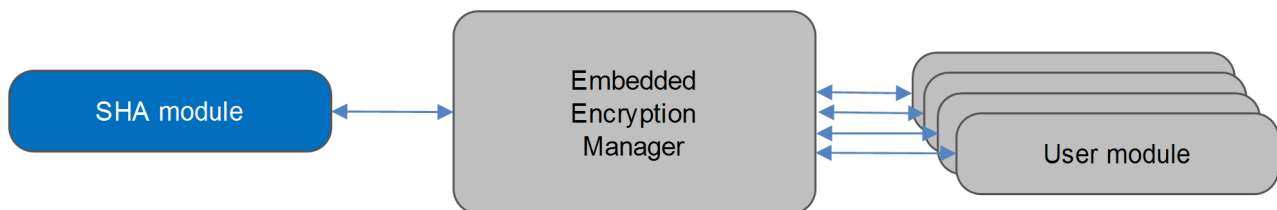
# 1 System Overview

## 1.1 Introduction

This guide is for those who want to generate hash codes for data by using any of the following Secure Hash Algorithms (SHAs): SHA-1, SHA-256, SHA-384 or SHA-512. The SHA module implements all these hash algorithms.

You register the SHA module with HCC's Embedded Encryption Manager (EEM), making it usable by other applications (for example, HCC's TLS/DTLS) through a standard interface. The EEM is the core component of HCC's encryption system.

The system structure is shown below:

**Note:**

- Although every attempt has been made to simplify the system's use, to get the best results you must understand clearly the requirements of the systems you design.
- HCC Embedded offers hardware and firmware development consultancy to help you implement your system; contact [sales@hcc-embedded.com](mailto:sales@hcc-embedded.com).

## 1.2 Feature Check

The main features of the SHA module are the following:

- It conforms to the HCC Advanced Embedded Framework.
- It conforms to the HCC Coding Standard including full MISRA compliance.
- It can be used with or without an RTOS.
- It conforms to the HCC Embedded Encryption Manager (EEM) standard and is compatible with the EEM.
- It can be verified by using the HCC Encryption Test Suite.
- The SHA-1 implementation conforms to RFC 3174.
- The SHA-256, SHA-384, and SHA-512 implementations conform to RFC 4634.

## 1.3 Packages and Documents

### Packages

The table below lists the packages that you need in order to use this module.

Package	Description
hcc_base_docs	This contains the two guides that will help you get started.
enc_base	The EEM base package.
enc_sha	The SHA package described in this document.

### Documents

For an overview of HCC verifiable embedded network encryption, see [Product Information](#) on the main HCC website.

Readers should note the points in the [HCC Documentation Guidelines](#) on the HCC documentation website.

#### HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the [Quick Start Guide](#) when HCC provides package updates.

#### HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

#### HCC Embedded Encryption Manager User Guide

This document describes the EEM.

## HCC Secure Hash Algorithm (SHA) User Guide

This is this document.

### 1.4 Change History

---

This section includes recent changes to this product. For a list of all changes, refer to the file **src/history/enc/enc\_sha.txt** in the distribution package.

Version	Changes
1.03	Added support for SHA-512 and SHA-384. Corrected output length check in SHA-256.
1.02	Added SHA1_BLOCK_LEN, SHA256_BLOCK_LEN defines to API. Simplified SHA-1 and SHA-256 calculation algorithm.
1.01	Changed version number to match new <b>enc_base</b> package.

## 2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

**Note:** Do not modify any of these files.

### 2.1 API Header File

---

The file `src/api/api_enc_sw_sha.h` is the only file that should be included by an application using this module. For details of the functions, see [Application Programming Interface](#).

### 2.2 System File

---

The file `src/enc/software/sha/sha.c` contains the source code. **This file should only be modified by HCC**

### 2.3 Platform Support Package (PSP) File

---

The file `psp/include/psp_endianness.h` provides macros for reading and writing little- and big-Endian values.

**Note:** You must modify this PSP implementation for your specific microcontroller and development board; see [PSP Porting](#) for details.

### 2.4 Version File

---

The file `src/version/ver_enc_sw_sha.h` contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

## 3 Application Programming Interface

This section describes the Application Programming Interface (API) functions, the hash output sizes, and the error codes.

### 3.1 sha1\_init\_fn

Call this function from the EEM to forward the structure containing SHA-1 functions to it.

#### Format

```
t_enc_ret sha1_init_fn ( t_enc_driver_fn const * * const pp_encdriver )
```

#### Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing SHA-1 functions.	t_enc_driver_fn **

#### Return Values

Return value	Description
ENC_SUCCESS	Successful execution.
ENC_INVALID_ERR	The module has already been initialized.

## 3.2 sha256\_init\_fn

Call this function from the EEM to forward the structure containing SHA-256 functions to it.

### Format

```
t_enc_ret sha256_init_fn ( t_enc_driver_fn const * * const pp_encdriver )
```

### Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing SHA-256 functions.	t_enc_driver_fn **

### Return Values

Return value	Description
ENC_SUCCESS	Successful execution.
ENC_INVALID_ERR	The module has already been initialized.



### 3.3 sha384\_init\_fn

Call this function from the EEM to forward the structure containing SHA-384 functions to it.

#### Format

```
t_enc_ret sha384_init_fn ( t_enc_driver_fn const * * const pp_encdriver )
```

#### Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing SHA-384 functions.	t_enc_driver_fn **

#### Return Values

Return value	Description
ENC_SUCCESS	Successful execution.
ENC_INVALID_ERR	The module has already been initialized.

### 3.4 sha512\_init\_fn

Call this function from the EEM to forward the structure containing SHA-512 functions to it.

#### Format

```
t_enc_ret sha512_init_fn ( t_enc_driver_fn const * * const pp_encdriver )
```

#### Arguments

Parameter	Description	Type
pp_encdriver	A pointer to a structure containing SHA-512 functions.	t_enc_driver_fn **

#### Return Values

Return value	Description
ENC_SUCCESS	Successful execution.
ENC_INVALID_ERR	The module has already been initialized.

## 3.5 Hash Output Sizes

---

The hash output sizes are defined in the file `src/api/api_enc_sw_sha.h`.

Name	Value	Description
SHA1_OUT_LEN	20	The size of the SHA-1 hash output.
SHA256_OUT_LEN	32	The size of the SHA-256 hash output.
SHA384_OUT_LEN	48	The size of the SHA-384 hash output.
SHA512_OUT_LEN	64	The size of the SHA-512 hash output.

## 3.6 Error Codes

---

The table below lists the error codes that may be generated by the API calls.

Error code	Value	Meaning
ENC_SUCCESS	0	Successful execution.
ENC_INVALID_ERR	1	The module has already been initialized.

## 4 Integration

The SHA module is designed to be as open and as portable as possible. No assumptions are made about the functionality, the behavior, or even the existence, of the underlying operating system. For the system to work at its best, perform the porting outlined below. This is a straightforward task for an experienced engineer.

### 4.1 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer. For full details of these functions, see the *Platform Support Package (PSP) Base User Guide*.

The module makes use of the following standard PSP functions:

Function	Package	Element	Description
<b>psp_memcpy()</b>	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
<b>psp_memset()</b>	psp_base	psp_string	Sets the specified area of memory to the defined value.

The module makes use of the following standard PSP macros. These are defined in the file **psp/include/psp\_endianness.h**.

Macro	Package	Element	Description
PSP_RD_BE32	psp_base	psp_endianness	Reads a 32 bit value stored as big-endian from a memory location.
PSP_WR_BE32	psp_base	psp_endianness	Writes a 32 bit value stored as big-endian to a memory location.
PSP_RD_BE64	psp_base	psp_endianness	Reads a 64 bit value stored as big-endian from a memory location.
PSP_WR_BE64	psp_base	psp_endianness	Writes a 64 bit value stored as big-endian to a memory location.

**Note:** You must modify this PSP implementation for your specific microcontroller and development board.