



# **RTOS CE 开发指南**

**版本号: 1.0**  
**发布日期: 2020-11-03**

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.10.30	SWC	1. 初版



# 目 录

<b>1 前言</b>	<b>1</b>
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
<b>2 模块介绍</b>	<b>2</b>
2.1 模块功能介绍	2
2.2 相关术语介绍	2
2.3 模块配置介绍	2
2.4 CE 源码结构介绍	3
<b>3 CE 模块接口说明</b>	<b>4</b>
3.1 接口列表	4
3.2 接口使用说明	4
3.2.1 CE 初始化接口	4
3.2.2 AES 算法加解密接口	4
3.2.3 HASH 算法运算接口	5
3.2.4 RSA 算法运算接口	6
<b>4 hwcrypto 设备</b>	<b>7</b>
4.1 symmetric 设备管理接口	7
4.1.1 创建对称加解密上下文	7
4.1.2 设置加解密密钥	7
4.1.3 设置对称加解密初始化向量	8
4.1.4 加解密接口	8
<b>5 模块使用范例</b>	<b>9</b>
<b>6 FAQ</b>	<b>10</b>

# 1 前言

## 1.1 文档简介

本文将介绍 CE 驱动的设计结构、流程、API 接口。主要包含 CE 接口部分和算法处理部分。

## 1.2 目标读者

硬件底层设计人员，驱动编写、维护人员，应用开发人员。

## 1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
V833	Melis	drivers/hal/source/ce/*
R328	FreeRTOS	drivers/hal/source/ce/*

## 2 模块介绍

### 2.1 模块功能介绍

CE 模块主要支持对称算法、非对称算法、摘要算法进行数据的加密和解密功能。CE 模块主要支持的算法如下：

- AES 算法 ECB/CBC/CTR/CTS/OFB/CFB/CBC-MAC/XTS 等模式.
- HASH 算法 MD5/SHA1/SHA224/SHA256/SHA384/SHA512/HMAC-SHA1/HMAC-SHA256.
- 非对称算法 RSA512/1024/2048/3072/4096.

### 2.2 相关术语介绍

- CE: Crypto Engine, 算法引擎, 以前称作 SS
- AES: Advanced Encryption Standard, 高级加密标准
- 3DES: 3DES 基于 DES 的一种改进算法, 它使用 3 条 64 位的密钥对数据进行三次加密
- CBC: Cipher Block Chaining mode, 加密块链模式
- ECB: Electronic Code Book mode, 电子密码本模式

### 2.3 模块配置介绍

其 menuconfig 的配置如下：

```
Kernel Setup --->
  Drivers Setup --->
    SoC HAL Drivers --->
      CE devices --->
        [*] enable ce driver
        [*] enable ce hal APIs Test command
```

## 2.4 CE 源码结构介绍

驱动位于 source/drivers/hal/source/ce。

```
hal/
├── source
│   ├── ce
│   │   ├── ce_common.c           ;CE公用操作接口函数文件
│   │   ├── ce_common.h         ;CE操作接口函数头文件
│   │   ├── hal_ce.c            ;CE底层驱动文件
│   │   ├── hal_ce.h            ;CE底层驱动头文件
│   │   ├── Makefile
│   │   └── platform.h           ;平台配置头文件
│   ├── platform
│   │   ├── ce_sun8iw18.h        ;具体的平台配置头文件
│   │   └── ce_sun8iw19.h        ;具体的平台配置头文件
│   ├── include/hal
│   └── sunxi_hal_ce.h           ;CE公用操作接口函数头文件
```

## 3 CE 模块接口说明

### 3.1 接口列表

CE 提供的接口列表如下：

```
int sunxi_ce_init(void);  
int do_aes_crypto(crypto_aes_req_ctx_t *req_ctx);  
int do_hash_crypto(crypto_hash_req_ctx *req_ctx);  
int do_rsa_crypto(crypto_rsa_req_ctx *req_ctx);
```

### 3.2 接口使用说明

#### 3.2.1 CE 初始化接口

- 原型：int sunxi\_ce\_init(void)
- 功能：CE 模块初始化，主要申请中断、clk 初始化等
- 参数：
  - 无
- 返回值：
  - 0 代表成功
  - 负数代表失败

#### 3.2.2 AES 算法加解密接口

- 原型：int do\_aes\_crypto(crypto\_aes\_req\_ctx\_t \*req\_ctx)
- 功能：主要实现对 AES 算法进行加解密
- 参数：
  - 为 AES 算法上下文的结构体
- 返回值：
  - 0 代表成功

- 负数代表失败
- 上下文数据结构：

```
typedef struct {
    uint8_t *src_buffer;
    uint32_t src_length;
    uint8_t *dst_buffer;
    uint32_t dst_length;
    uint8_t *iv;
    uint8_t *key;
    uint32_t key_length;
    uint32_t dir; /*0--加密, 1--解密*/
    uint32_t mode; /*AES算法的模式*/
} crypto_aes_req_ctx_t;
```

### 3.2.3 HASH 算法运算接口

- 原型：int do\_hash\_crypto(crypto\_hash\_req\_ctx \*req\_ctx)
- 功能：主要实现对 HASH 算法进行运算
- 参数：
  - 为 HASH 算法上下文的结构体
- 返回值：
  - 0 代表成功
  - 负数代表失败
- 上下文数据结构：

```
typedef struct {
    uint8_t *src_buffer;
    uint32_t src_length;
    uint8_t *dst_buffer;
    uint32_t dst_length;
    uint8_t md[SHA_MAX_DIGEST_SIZE];
    uint32_t md_size;
    uint32_t type; /*hash算法的模式*/
    uint32_t dir;
    uint32_t padding_mode; /*hash算法的填充模式*/
} crypto_hash_req_ctx;
```

备注：具体底层尚未实现，只是预留接口给未来开发使用。



### 3.2.4 RSA 算法运算接口

- 原型：int do\_rsa\_crypto(crypto\_rsa\_req\_ctx \*req\_ctx)
- 功能：主要实现对 RSA 算法进行加解密
- 参数：
  - 为 RSA 算法上下文的结构体
- 返回值：
  - 0 代表成功
  - 负数代表失败
- 上下文数据结构：

```
typedef struct {  
    uint8_t *key_n; /*公钥模数*/  
    uint32_t n_len;  
    uint8_t *key_e; /*公钥指数*/  
    uint32_t e_len;  
    uint8_t *key_d; /*私钥*/  
    uint32_t d_len;  
    uint8_t *src_buffer;  
    uint32_t src_length;  
    uint8_t *dst_buffer;  
    uint32_t dst_length;  
    uint32_t dir; /*0--加密, 1--解密*/  
    uint32_t type; /*RSA算法的模式*/  
    uint32_t bitwidth; /*RSA算法位宽*/  
} crypto_rsa_req_ctx;
```

备注：具体底层尚未实现，只是预留接口给未来开发使用。

## 4 hwcrypto 设备

由于硬件接口不一，种类繁多，软件对接起来比较困难，RT-Thread 推出了 hwcrypto 硬件加解密驱动框架，并对接了常见的安全传输套件。sunxi 已经适配 RT-Thread 的 hwcrypto 硬件加解密驱动框架（目前只适配 AES 算法），因此可以直接使用 hwcrypto 的 API 接口即可。由于 hwcrypto 的框架的接口众多，这里只选择常用 AES 算法的接口介绍，其他接口可以参考 RT-Thread 官网。

### 4.1 symmetric 设备管理接口

#### 4.1.1 创建对称加解密上下文

- 原型：struct rt\_hwcrypto\_ctx rt\_hwcrypto\_symmetric\_create(struct rt\_hwcrypto\_device device, hwcrypto\_type type);
- 功能：对称加解密设备的句柄，创建对称加解密上下文
- 参数：
  - 加解密设备句柄，选择 rt\_hwcrypto\_dev\_default() 即可
  - 对称加解密算法模式
- 返回值：
  - 其他代表成功
  - NULL 代表失败

#### 4.1.2 设置加解密密钥

- 原型：rt\_hwcrypto\_symmetric\_setkey(ctx, key, keybits);
- 功能：根据对称加解密的上下文、密钥与密钥长度，设置加解密密钥
- 参数：
  - 上下文
  - 输入密钥
  - 密钥长度
- 返回值：
  - RT\_EOK 代表成功
  - 其他代表失败

### 4.1.3 设置对称加解密初始化向量

- 原型：rt\_err\_t rt\_hwcrypto\_symmetric\_setiv(struct rt\_hwcrypto\_ctx ctx, const rt\_uint8\_t iv, rt\_size\_t len);
- 功能：根据对称加解密的上下文、初始化向量与向量长度，设置对称加解密初始化向量
- 参数：
  - 上下文
  - 初始化向量
  - 初始化向量长度
- 返回值：
  - RT\_EOK 代表成功
  - 其他代表失败

### 4.1.4 加解密接口

- 原型：rt\_err\_t rt\_hwcrypto\_symmetric\_crypt(struct rt\_hwcrypto\_ctx ctx, hwcrypto\_mode mode, rt\_size\_t length, const rt\_uint8\_t in, rt\_uint8\_t \*out);
- 功能：对数据进行加解密
- 参数：
  - 上下文
  - 加密或解密
  - 数据长度
  - 输入数据
  - 输出数据
- 返回值：
  - RT\_EOK 代表成功
  - 其他代表失败

## 5 模块使用范例

---

可参考驱动 APIs 测试代码（`/source/ekernel/drivers/test/test_ce.c`）。



## 6 FAQ

---






## 著作权声明

版权所有 © 2020 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

## 商标声明

、 全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

## 免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。