



Hi3510 DES 和 AES 网络加解密实现

## Application Notes

文档版本 01

发布日期 2006-09-10

BOM编码 N/A

深圳市海思半导体有限公司为客户提供全方位的技术支持，用户可与就近的海思办事处联系，也可直接与公司总部联系。

## 深圳市海思半导体有限公司

地址： 深圳市龙岗区坂田华为基地华为电气生产中心 邮编： 518129

网址： <http://www.hisilicon.com>

客户服务电话： 0755-28788858

客户服务传真： 0755-28788838

客户服务邮箱： [support@hisilicon.com](mailto:support@hisilicon.com)

**版权所有 © 深圳市海思半导体有限公司 2006。 保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

### 商标声明



、Hisilicon、海思，均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

### 注意

由于产品版本升级或其它原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。



# 目 录

前言.....	1
1 概述.....	1-1
2 DES 和 3DES 加解密.....	2-1
2.1 支持特性.....	2-2
2.2 加密步骤.....	2-2
2.3 解密步骤.....	2-6
2.4 加解密注意事项.....	2-8
3 AES 加解密.....	3-1
3.1 支持特性.....	3-2
3.2 非 CTR 模式加解密步骤.....	3-2
3.2.1 非 CTR 模式加密步骤.....	3-2
3.2.2 非 CTR 模式解密步骤.....	3-5
3.3 CTR 模式加解密步骤.....	3-6
3.3.1 CTR 模式加密步骤.....	3-6
3.3.2 CTR 模式解密步骤.....	3-9
3.4 加解密注意事项.....	3-10
A 缩略语.....	A-1



# 前 言

## 概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

## 产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
Hi3510	V100

## 读者对象

本指南为软件开发人员编写。使用本书的程序员应该：

- 掌握相关硬件基本知识
- 熟练掌握 C 语言
- 掌握基本的 Linux 环境编程

## 内容简介

本文档介绍用 DES&AES 模块实现网络加解密。

章节	内容
1 概述	介绍 DES 和 AES 加解密模块。
2 DES 和 3DES 加解密	介绍 DES、3DES 加解密的步骤和注意事项。
3 AES 加解密	介绍 AES 加解密的步骤和注意事项。





章节	内容
附录 A 缩略语表	介绍本书中出现的缩略语。

## 约定

### 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	以本标志开始的文本表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
 说明	以本标志开始的文本是正文的附加信息，是对正文的强调和补充。

### 通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离。
“Terminal Display” 格式	“Terminal Display” 格式表示屏幕输出信息。此外，屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。



## 修改记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

### 文档版本 01 (2006-9-10)

第一次版本。



# 1 概述

DES 和 AES 加解密模块提供了 DES、3DES、AES 三种加解密算法，多种加解密模式。

通过 DES 和 AES 加解密模块，实现以下功能：

- 64bit 明密文的 DES 和 3DES 算法加解密
- 128bit 明密文的 AES 算法加解密

本文详细的介绍运用 DES 和 AES 加解密模块进行加解密的过程和注意事项。







# 2 DES 和 3DES 加解密

## 关于本章

本章描述内容如下表所示。

标题	内容
2.1 支持特性	描述 DES 和 3DES 加解密算法支持的特性。
2.2 加密步骤	描述 DES 和 3DES 加密步骤。
2.3 解密步骤	描述 DES 和 3DES 解密步骤。
2.4 加解密注意事项	描述 DES 和 3DES 加解密的注意事项。



## 2.1 支持特性

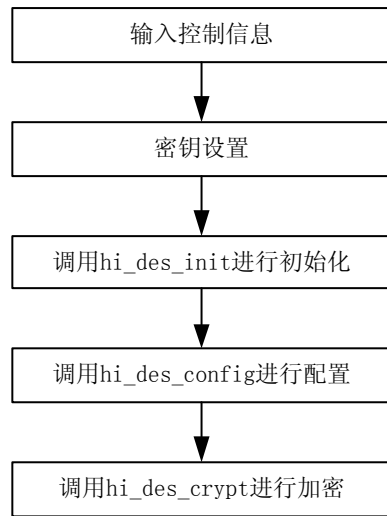
DES 和 3DES 加解密算法支持特性如下：

- 支持对 64bit 数据加解密。
- 支持 DES、3DES 两种加解密算法，DES 加解密算法需要 1 个 64bit 密钥，3DES 加解密算法需要 2 个或 3 个 64bit 密钥。
- 支持密钥的大小端存放方式。
- 支持 ECB、CBC、1-CFB、8-CFB、64-CFB、1-OFB、8-OFB、64-OFB 等 8 种加解密模式。
- 明密文长度必须为 64bit 的倍数，如果长度不够，需要补齐。

## 2.2 加密步骤

DES 和 3DES 加密步骤如图 2-1 所示。

图2-1 DES 和 3DES 加密步骤



### 步骤 1 输入控制信息

结构体 `des_encrypt_ctrl` 接受用户输入的控制信息，结构体的定义如下：

```
struct des_encrypt_ctrl
{
    int des_endian;
    int des_shift;
    int des_type;
    int des_alg;
    unsigned char iv[8];
};
```



```
}des_encrypt_ctrl;
```

结构体各个域含义如表 2-1 所示。

表2-1 结构体各个域含义

域名称	描述
des_endian	表示密钥的大小端存放方式。赋值范围：{0,1}。 0 表示小端方式，低字节在前面。 1 表示大端方式，高字节在前面。
des_shift	模块内部一次加解密的位宽。赋值范围：{0,1,2,3}。 0 表示 64bit，1 表示 8bit，2 表示 1bit，3 表示 64bit。 在 CFB 和 OFB 模式下，赋值范围是{0,1,2,3}。其他模式下只能将此值赋为 0 或 3。
des_type	表示加解密的模式。赋值范围：{0,1,2,3}。 0 表示 ECB，1 表示 CBC，2 表示 CFB，3 表示 OFB。
des_alg	表示选用的加解密算法。赋值范围：{0,1}。 0 表示 DES，1 表示 3DES。
iv	表示开始加解密时输入的初始向量，长度为 64bit。

控制信息赋值示例如下：

```
void set_value ()
{
    unsigned char iv_tmp[8];
                                     //其他变量定义或声明

    struct des_encrypt_ctrl ctrl;
    ctrl.des_endian = 0;           //key存放方式为小端存放方式
    ctrl.des_shift = 2;           //一次加解密的位宽为1bit
    ctrl.des_type = 2;            //CFB模式
    ctrl.des_alg = 1;             //3DES加解密

    for(i=0;i++;i<8)
    {
        ctrl.iv[i] = iv_tmp[i];
    }
                                     //其他操作
}
```



## 说明

ECB 模式下无需给 iv 向量赋值。在其他模式下，加解密时 iv 向量必须相同。

## 步骤 2 密钥设置

通过结构体 `keys` 对密钥赋值，结构体定义如下：

```
struct keys{
    unsigned char keys1[8];
    unsigned char keys2[8];
    unsigned char keys3[8];
};
```

加密过程中可以通过调用 `hi_des_crypt` 更换密钥，在解密时需要使用相对应的密钥进行解密。赋值规则如下：

- DES 加解密  
只需给结构体中的第一个域 `keys1` 赋值。
- 3DES 加解密
  - 如果选用 3 个密钥加密  
需要给结构体 `keys` 的三个域 `keys1`、`keys2`、`keys3` 赋值。
  - 如果选用 2 个密钥加密  
需要给结构体 `keys` 的域 `keys1`、`keys2` 赋值，给 `keys3` 的赋值与 `keys1` 值相等。

如选用 2 个密钥加密，赋值示例如下：

```
void set_key()
{
    unsigned char key_1_input[8];
    unsigned char key_2_input[8];
    Struct keys key_tmp;
    int i = 0;

    for(i = 0;i++;i<8)
    {
        Key_tmp.keys1[i] = key_1_input[i];
        Key_tmp.keys3[i] = key_1_input[i];
        Key_tmp.keys2[i] = key_2_input[i];
    }
}
```

如选用 3 个密钥加密，赋值示例如下：

```
void set_key()
{
    unsigned char key_1_input[8];
    unsigned char key_2_input[8];
    unsigned char key_3_input[8];
```



```
Struct keys key_tmp;  
int i = 0;  
  
for(i = 0;i++;i<8)  
{  
    Key_tmp.keys1[i] = key_1_input[i];  
    Key_tmp.keys2[i] = key_2_input[i];  
    Key_tmp.keys3[i] = key_3_input[i];  
}  
}
```

### 步骤 3 调用 hi\_des\_init 进行初始化

在加解密之前，必须调用 hi\_des\_init 函数进行初始化。示例如下：

```
void main()  
{  
  
    //输入控制信息，给密钥赋值  
  
    hi_des_init();  
  
}
```

### 步骤 4 调用 hi\_des\_config 进行配置

将赋值后的 des\_encrypt\_ctrl 作为 hi\_des\_config 的参数，然后将控制信息保存到寄存器。即步骤 1 中设置的数据需要通过调用这个函数生效。

示例如下：

```
void test()  
{  
  
    //其他变量声明及定义  
  
    struct des_encrypt_ctrl ctrlstruct;  
    //给ctrlstruct的域赋值，给密钥赋值  
  
    hi_des_init();  
    hi_des_config(&ctrlstruct);  
    //其他操作  
  
}
```

### 步骤 5 调用 hi\_des\_crypt 进行加密

函数原型如下：

```
int hi_des_crypt(unsigned char* src, unsigned char* dest,  
                unsigned int byte_length, struct keys *pcipher);
```

各个参数的说明如下：

- src                      unsigned char 类型指针，指向待加密明文。



- dest unsigned char 类型指针，指向加密后的密文。
- byte\_length 明文的长度，以 byte 为单位，长度必须为 8 的倍数。
- pcipher 结构体指针，指向 64bit 密钥。

采用 3DES 加解密，选用 3 个密钥加密，示例如下：

```
void main()
{
    unsigned char plaintext[8] = "1234567";
    unsigned char cipher_out[8];
    struct keys cipher_key;
    struct des_encrypt_ctrl ctrlstruct;
    unsigned int i = 0;
    unsigned char key_1_input[8];
    unsigned char key_2_input[8];
    unsigned char key_3_input[8];

                                //控制信息输入
    for(i = 0;i++;i<8)
    {
        cipher_key.keys1[i] = key_1_input[i];
        cipher_key.keys2[i] = key_2_input[i];
        cipher_key.keys3[i] = key_3_input[i];
    }

    hi_des_init();
    hi_des_config(&ctrlstruct);
    hi_des_crypt(plaintext, cipher_out, 8, &cipher_key);
                                //其他操作
}
```

示例中，cipher\_out 为加密后的密文，如选用 2 个密钥加密，需要给 keys1、keys2 赋值，给 keys3 的赋值与 keys1 值相等。

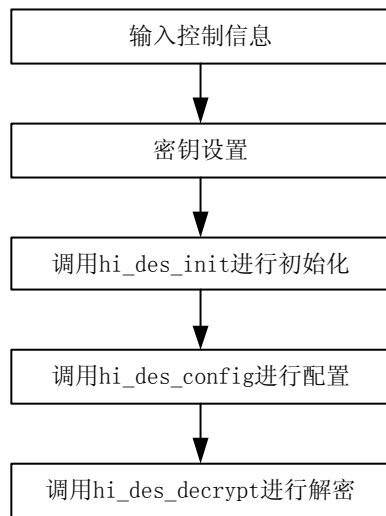
如果选用 DES 加解密算法，只需给 keys1 赋值即可。

----结束

## 2.3 解密步骤

解密步骤 1~4 与加密步骤中 1~4 步骤相同，具体请参见“2.2 加密步骤”。DES 和 3DES 解密步骤如图 2-2 所示。

图2-2 DES 和 3DES 解密步骤



步骤 1 输入控制信息

步骤 2 密钥设置

步骤 3 调用 hi\_des\_init 进行初始化

步骤 4 调用 hi\_des\_config 进行配置

步骤 5 调用 hi\_des\_decrypt 进行解密

调用 hi\_des\_decrypt 进行解密。解密函数定义如下：

```
int hi_des_decrypt(unsigned char * src, unsigned char * dest,  
                  unsigned int byte_length, struct keys *pcipher);
```

各个参数的说明如下：

- src                      unsigned char 类型指针，指向待解密密文。
- dest                     unsigned char 类型指针，指向解密后的明文。
- byte\_length             密文的长度，以 byte 为单位，长度必须为 8 的倍数。
- pcipher                 结构体指针，指向 64bit 密钥。

采用 3DES 加解密算法，选用 3 个密钥解密，示例如下：

```
void main()  
{  
    unsigned char cipher_in[8] = "1234567";  
    unsigned char plaintext_out[8];  
    struct keys cipher_key;  
    struct des_encrypt_ctrl ctrlstruct;  
    unsigned int i = 0;  
    unsigned char key_1_input[8];  
    unsigned char key_2_input[8];
```



```
unsigned char key_3_input[8];

//控制信息输入

for(i = 0;i++;i<8)
{
    cipher_key.keys1[i] = key_1_input[i];
    cipher_key.keys2[i] = key_2_input[i];
    cipher_key.keys3[i] = key_3_input[i];
}
hi_des_init();
hi_des_config(&ctrlstruct);
hi_des_decrypt (cipher_in, plaintext_out,8,&cipher_key);
//其他操作
}
```

示例中，plaintext\_out 为解密后的明文。如选用 2 个密钥解密，需要给 keys1、keys2 赋值，keys3 的赋值与 keys1 相等。

如果选用 DES 加解密算法，只需给 keys1 赋值即可。

----结束

## 2.4 加解密注意事项

DES 和 3DES 模块是独立模块，可以和其他模块同时运行，占用总线带宽。加解密注意事项如下：

- 加密密钥与解密密钥相同。  
将密文解密为明文，解密密钥必须与加密密钥相同。  
例如：明文 a 通过密钥 key c 加密后生成密文 b，要解密密文 b 到明文 a，解密密钥必须也为 key c。
- 解密时输入的初始向量 iv 必须与加密时输入的初始向量一致。
- 明密文长度  
明密文长度必须为 64bit 的倍数。如果长度不够，需要补齐。低字节放置有效数据，高字节放置填充数据。
- iv 赋值  
ECB 模式与其他模式不同，不需要给 iv 向量赋值。





# 3 AES 加解密

## 关于本章

本章描述内容如下表所示。

标题	内容
<a href="#">3.1 支持特性</a>	描述 AES 加解密算法支持的特性。
<a href="#">3.2 非 CTR 模式加解密步骤</a>	描述 AES 加解密算法下非 CTR 模式的加解密步骤。
<a href="#">3.3 CTR 模式加解密步骤</a>	描述 AES 加解密算法下 CTR 模式的加解密步骤。
<a href="#">3.4 加解密注意事项</a>	描述 AES 加解密的注意事项。



## 3.1 支持特性

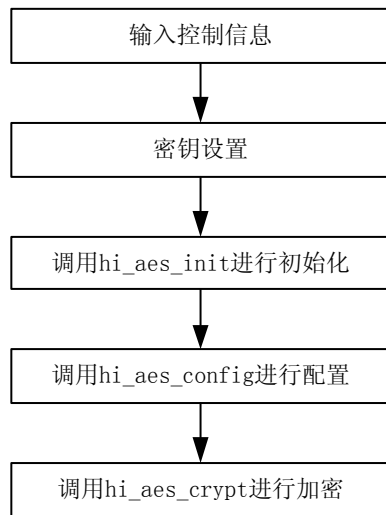
- 支持对 128bit 数据加解密。
- 密钥长度只支持 128bit。
- 支持 ECB、CBC、1-CFB、8-CFB、128-CFB、OFB、CTR 模式。
- 明密文长度为 128bit 的倍数，如果长度不够，需要补齐。

## 3.2 非 CTR 模式加解密步骤

### 3.2.1 非 CTR 模式加密步骤

非 CTR 模式加密步骤如图 3-1 所示。

图3-1 非 CTR 模式加密步骤



#### 步骤 1 输入控制信息

结构体 `aes_encrypt_ctrl` 接受用户输入的控制信息，结构体的定义如下：

```
struct aes_encrypt_ctrl
{
    int aes_type;
    unsigned char iv[16];
};
```

结构体各个域含义如表 3-1 所示。



表3-1 结构体各个域含义

域名称	描述
aes_type	表示加解密的模式。赋值范围：{0,1,2,3,4,5,6,7}。 0 表示 ECB，1 表示 CBC，2 表示 1-CFB，3 表示 8-CFB，4 表示 128-CFB，5 表示 OFB，6 表示 CTR，7 表示 ECB。
iv	表示开始加解密时，输入的初始向量，长度为 128bit。 此域对 ECB，CTR 模式无效，这两种模式下无需赋值。

```
void set_value ()
{
    unsigned char iv_tmp[16];
    struct aes_encrypt_ctrl ctrl;
    //其他变量定义或声明

    ctrl.aes_type = 2; //选取1-CFB模式
    for(i = 0;i++;i<16)
    {
        ctrl.iv[i] = iv_tmp[i];
    }
    //其他操作
}
```

说明

加密过程中可以更换 iv 向量。但是在解密时需要使用相对应的 iv 向量进行解密。

**步骤 2 密钥设置**

AES 算法与 DES 下不同，没有定义结构体来接受密钥的输入，密钥设置示例如下：

```
void set_key()
{
    unsigned char key_input[16];
    int i = 0;

    for(i = 0;i++;i<16)
    {
        Key_input[i] = 0xab+i;
    }
}
```

在调用加解密函数时输入赋值后的密钥。

**步骤 3 调用 hi\_aes\_init 进行初始化**

在加解密之前，必须调用 hi\_aes\_init 函数进行初始化。示例如下：



```
void main()
{
    //输入控制信息，密钥赋值
    hi_aes_init();
    //其他操作
}
```

#### 步骤 4 调用 hi\_aes\_config 进行配置

将赋值后的 struct aes\_encrypt\_ctrl 变量作为 hi\_aes\_config 的参数，然后将控制信息保存到寄存器。即步骤 1 中设置的数据需要通过这个函数生效。

示例如下：

```
void test()
{
    struct aes_encrypt_ctrl ctrlstruct;
    //结构体ctrlstruct各个域的赋值
    hi_aes_init();
    hi_aes_config(&ctrlstruct);
    //其他操作
}
```

#### 步骤 5 调用 hi\_aes\_crypt 进行加密

调用 hi\_aes\_crypt 函数进行加密。函数原型如下：

```
int hi_aes_crypt(unsigned char * src, unsigned char* dest,
                unsigned int byte_length, unsigned char *pcipher);
```

各个参数的说明如下：

- src unsigned char 类型指针，指向需加密明文。
- dest unsigned char 类型指针，指向加密后的密文。
- byte\_length 明文的长度，以 byte 为单位，长度必须为 16 的倍数。
- pcipher unsigned char 类型指针，指向 128bit 密钥。

示例说明如下：

```
void main()
{
    unsigned char plaintext_in[16] = "123456788765432";
    unsigned char cipher_out[16];
    struct aes_encrypt_ctrl aes_encrypt_ctrl;
    unsigned char key[16];
    //控制信息输入，密钥赋值
    hi_aes_init();
    hi_aes_config(&ctrlstruct);
```



```
    hi_aes_crypt (plaintext_in, cipher_out, 16, key);  
                //其他操作  
}
```

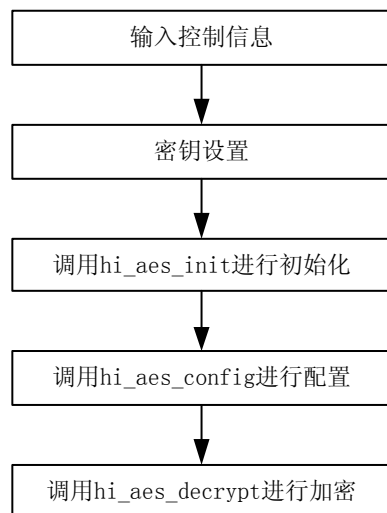
示例中，cipher\_out 为加密后的密文。

----结束

## 3.2.2 非 CTR 模式解密步骤

非 CTR 模式解密步骤 1~4 与非 CTR 模式加密步骤中 1~4 步骤相同，具体请参见“3.2.1 非 CTR 模式加密步骤”。非 CTR 模式解密步骤如图 3-2 所示。

图3-2 非 CTR 模式解密步骤



步骤 1 输入控制信息

步骤 2 密钥设置

步骤 3 调用 hi\_aes\_init 进行初始化

步骤 4 调用 hi\_aes\_config 进行配置

步骤 5 调用 hi\_aes\_decrypt 进行解密

调用 hi\_aes\_decrypt 进行解密。函数原型如下：

```
int hi_aes_decrypt(unsigned char * src, unsigned char * aest,  
                  unsigned int byte_length, unsigned char *pcipher);
```

其中各个参数的说明如下：

- src unsigned char 类型指针，指向需解密的密文。
- dest unsigned char 类型指针，指向解密后的明文。
- byte\_length 密文长度，以 byte 为单位，长度必须为 16 的倍数。



- `pcipher`                      `unsigned char` 类型指针，指向 128bit 密钥。

示例说明如下：

```
void main()
{
    unsigned char cipher_in[16] = "123456788765432";
    unsigned char plaintext_out[16];
    struct aes_encrypt_ctrl aes_encrypt_ctrl;
    unsigned char key[16];

    //控制信息输入，密钥赋值

    hi_aes_init();
    hi_aes_config(&ctrlstruct);
    hi_aes_decrypt (cipher_in, plaintext_out, 16, key);
    //其他操作
}
```

示例中，`plaintext_out` 为解密后的明文。

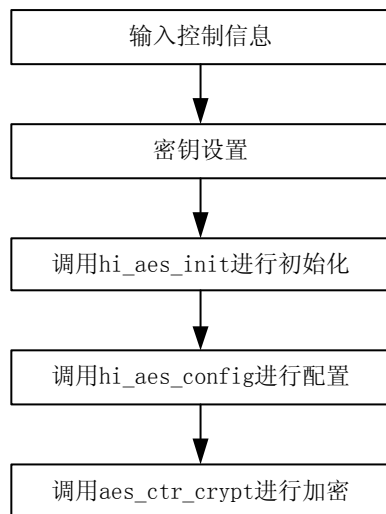
----结束

## 3.3 CTR 模式加解密步骤

### 3.3.1 CTR 模式加密步骤

CTR 模式下，加解密模块在加密时不会由初始向量生成下一次加密所需的向量。因此加密时，都需要用户提供和明文的长度相等的 `iv` 向量作为加密函数参数。故 CTR 模式下，结构体 `aes_encrypt_ctrl` 中的 `iv` 域无需赋值。CTR 模式加密步骤如图 3-3 所示。

图3-3 CTR 模式加密步骤





### 步骤 1 输入控制信息

CTR 模式加密步骤中输入控制信息描述与 3.2.1 中步骤 1，非 CTR 模式加密步骤中输入控制信息描述相同，具体请参见“3.2.1 非 CTR 模式加密步骤”。

控制信息的输入如下所示：

```
void set_value ()
{
    unsigned char iv_tmp[16];
                                //其他变量定义或声明

    struct aes_encrypt_ctrl ctrl;
    ctrl.aes_type = 6;           //选取CTR模式

                                //其他操作
}
```

CTR 模式下，初始向量的赋值如下所示：

```
void set_iv()
{
    unsigned char plaintext[64]; //明文
    unsigned char p_ivin[64];   //初始向量iv
    unsigned char aest[64];     //密文
    unsigned char pciher[64];   //密钥

    for(i=0;i<64;i++)
    {
        Plaintext[i] = 'a';
        p_ivin[i] = 'b';       //iv初始向量赋值
    }

                                //调用hi_aes_init,hi_aes_config
    aes_ctr_crypt(plaintext, aest, 64, pciher, p_ivin); //加密函数
                                //其他操作
}
```

由示例程序可知，初始向量作为 aes\_ctr\_crypt 函数的参数输入。

### 步骤 2 密钥设置

CTR 模式加密步骤中密钥的赋值与和 3.2.1 步骤 2，非 CTR 模式加密步骤密钥的赋值描述相同，具体请参见“3.2.1 非 CTR 模式加密步骤”。

### 步骤 3 调用 hi\_aes\_init 进行初始化

在加解密之前，必须调用 hi\_aes\_init 函数进行初始化。初始化函数示例如下：



```
void main()
{
    //输入控制信息，给密钥赋值
    hi_aes_init();
    //其他操作
}
```

#### 步骤 4 调用 hi\_aes\_config 进行配置

将赋值后的 struct aes\_encrypt\_ctrl 变量作为 hi\_aes\_config 的参数，然后将控制信息保存到寄存器。即步骤 1 中设置的数据需要通过调用这个函数生效。

示例如下：

```
void test()
{
    struct aes_encrypt_ctrl ctrlstruct;
    //结构体ctrlstruct各个域的赋值
    Hi_aes_init();
    hi_aes_config(&ctrlstruct);
    //其他操作
}
```

#### 步骤 5 调用 aes\_ctr\_crypt 进行加密

调用 aes\_ctr\_crypt 函数进行加密。加密函数定义如下：

```
int aes_ctr_crypt(unsigned char * src, unsigned char * aest,
                 unsigned int byte_length, unsigned char *pcipher,
                 unsigned char * ctr_iv);
```

其中各个参数的说明：

- src unsigned char 类型指针，指向需加密明文。
- dest unsigned char 类型指针，指向加密后的密文。
- byte\_length 明文长度，以 byte 为单位，长度必须为 16 的倍数。
- pcipher unsigned char 类型指针，指向 128bit 密钥。
- ctr\_iv unsigned char 类型指针，指向 CTR 模式下的 iv 输入向量。

示例说明如下：

```
void main()
{
    struct aes_encrypt_ctrl aes_encrypt_ctrl;
    unsigned char plaintext[64]; //明文
    unsigned char p_ivin[64]; //初始向量iv
    unsigned char cipher_out[64]; //密文
    unsigned char pciher[64]; //密钥
}
```





```
                                //输入控制信息，密钥设置  
  
    hi_aes_init();  
    hi_aes_config(&ctrlstruct);  
    aes_ctr_crypt( plaintext, cipher_out, 64, pciher, p_ivin); //加密函数  
                                //其他操作  
}
```

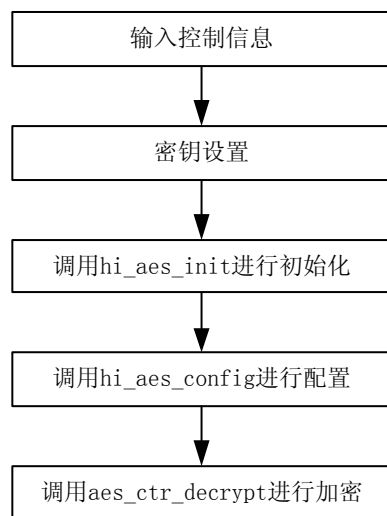
示例中，cipher\_out 为加密后的密文。

----结束

### 3.3.2 CTR 模式解密步骤

CTR 模式解密步骤 1~4 与 CTR 模式加密步骤中 1~4 步骤相同，具体请参见“3.3.1 CTR 模式加密步骤”。CTR 模式下，结构体 aes\_encrypt\_ctrl 中的 iv 域无需赋值，在解密函数中输入。输入密文的长度和输入的初始向量 iv 长度必须相等。CTR 模式解密步骤如图 3-4 所示。

图3-4 CTR 模式解密步骤



步骤 1 输入控制信息

步骤 2 密钥设置

步骤 3 调用 hi\_aes\_init 进行初始化

步骤 4 调用 hi\_aes\_config 进行配置

步骤 5 调用 aes\_ctr\_decrypt 进行解密

调用 aes\_ctr\_decrypt 进行解密。解密函数定义如下：

```
int aes_ctr_decrypt(unsigned char* src, unsigned char* aest,
```



```
unsigned int byte_length, unsigned char *pcipher,  
unsigned char* ctr_iv);
```

其中各个参数的说明:

- src unsigned char 类型指针, 指向需解密密文。
- dest unsigned char 类型指针, 指向解密后的明文。
- byte\_length 密文长度, 以 byte 为单位, 长度必须为 16 的倍数。
- pcipher unsigned char 指针, 指向 128bit 密钥。
- ctr\_iv unsigned char 指针, 指向 CTR 模式下的 iv 输入向量。

示例说明如下:

```
void main()  
{  
    struct aes_encrypt_ctrl aes_encrypt_ctrl;  
    unsigned char plaintext[64]; //明文  
    unsigned char p_ivin[64]; //初始向量iv  
    unsigned char cipher_out[64]; //密文  
    unsigned char pciher[64]; //密钥  
  
    //输入控制信息, 密钥设置  
  
    hi_aes_init();  
    hi_aes_config(&ctrlstruct);  
    aes_ctr_decrypt(cipher_out, plaintext, 64, pciher, p_ivin); //加密函数  
    //其他操作  
}
```

示例中, plaintext 为解密后的明文。

----结束

## 3.4 加解密注意事项

AES 模块是独立模块, 可以和其他模块同时运行, 占用总线带宽。加解密注意事项如下:

- 加密密钥与解密密钥相同。  
密文解密为明文, 解密密钥必须与加密密钥相同。  
例如: 明文 a 通过密钥 key c 加密后生成密文 b, 要解密 b 到明文 a, 解密密钥必须也为 key c。
- 解密时输入的初始向量 iv 必须与加密时输入的初始向量一致。
- 明密文长度



明密文长度必须为 128bit 的倍数。如果长度不够，需要补齐。低字节放置有效数据，高字节放置填充数据。

- 初始向量 iv 赋值情况

ECB 模式与其他模式不同，不需要给 iv 赋值。

CTR 模式下，输入明密文的长度和输入的初始向量 iv 长度必须相等。即：用户明密文长度为  $128 \times n$  bit，则输入的 iv 长度也必须为  $128 \times n$  bit。



# A 缩略语

## A

**AES**                      Advanced Encryption Standard                      先进的加密标准

## C

**CBC**                      Cipher Block Chaining                      密码分组链接

**CFB**                      Cipher FeedBack                      密码反馈

**CTR**                      Counter                      AES 的一种计数工作模式

## D

**DES**                      Data Encryption Standard                      数据加密标准

## E

**ECB**                      Electronic Code Book                      电子密码本

## N

**NIST**                      National Institute of Standards and Technology                      美国标准与技术研究所

## O

**OFB**                      Output FeedBack                      输出反馈