

32 位微控制器

HC32L15 系列的 AES

适用对象

系列	产品型号
HC32L15	HC32L150KATA
	HC32L150JATA
	HC32L150FAUA
	HC32L156KATA
	HC32L156JATA

目 录

1	摘要	3
2	AES 简介	3
3	HC32L15 系列的 AES	4
3.1	简介	4
3.2	说明	4
3.2.1	寄存器介绍	4
3.2.2	工作流程介绍	4
4	样例代码	7
4.1	代码介绍	7
4.2	代码运行	8
5	总结	11
6	版本信息 & 联系方式	12

1 摘要

本篇应用笔记主要介绍如何使用 HC32L15 系列的 AES 进行加解密运算。

2 AES 简介

什么是 AES?

高级加密标准（Advanced Encryption Standard，缩写：AES），在密码学中又称 Rijndael 加密法，是美国联邦政府采用的一种区块加密标准。

（引自‘百度百科’，‘互动百科’，‘维基百科’）

AES 的算法原理

AES 算法基于排列和置换运算。排列是对数据重新进行安排，置换是将一个数据单元替换为另一个。AES 使用几种不同的方法来执行排列和置换运算。

AES 是一个迭代的、对称密钥分组的密码，它可以使用 128、192 和 256 位密钥，并且用 128 位（16 字节）分组加密和解密数据。与公共密钥密码使用密钥对不同，对称密钥密码使用相同的密钥加密和解密数据。通过分组密码返回的加密数据的位数与输入数据相同。迭代加密使用一个循环结构，在该循环中重复置换和替换输入数据。

AES 的应用

随着计算机网络的不断发展，信息的安全性和保密性变得尤为重要。加密技术是对通信系统或者存储系统中的信息数据进行保护的一个很重要的方式。而 AES(Advanced Encryption Standard)算法是美国国家标准和技术研究所（NIST）在 21 世纪初正式推出的旨在取代 DES 算法的高级加密标准算法。用它对数据文件进行加密的运算，其优势在于 AES 是一种高效、安全的对称加密算法，具备很强的扩散性能，最终形成的密码有很高的随机性。数据文件经 AES 算法加密后，信息会得到有效保护。

3 HC32L15 系列的 AES

3.1 简介

华大 HC32L15 系列 MCU 的 AES 硬件加速器可以执行 AES 算法标准的加密流程和解密流程，其执行结果完全符合“FIPS PUB 197”对算法原理的描述。

可提供一个安全的硬件实现架构，能够有效抵御 Side Channel Attack 中的差异功耗分析（DPA）和差异错误分析（DFA）的攻击。

具有两个功能：加密、解密。

3.2 说明

本节介绍 HC32L15 系列的 AES，包括寄存器和工作流程。

3.2.1 寄存器介绍

对于 AES 模块的操作主要通过以下寄存器进行：

缩写	寄存器名称
AES_CON	控制寄存器
AES_DATA	数据寄存器
AES_KEY	密钥寄存器
AES_RNG	随机数寄存器
AES_DIV_EN	时钟分频寄存器

3.2.2 工作流程介绍

操作共同点

- 1) 上电后，首先必须通过 AHB 总线复位 hresetn 对本模块进行一次异步复位操作。时钟 HCLK 必须在复位脱离前稳定有效，并且在后续运行中持续稳定。
- 2) 在 AES 加解密过程中，数据寄存器会改变，如果下次运算操作的数据就是本次运算的结果，那么就无需重新写入数据。
- 3) 在用相同的密钥加解密大量数据的情况下，不需要重复写入密钥。

- 4) 为了有效抵御差异功耗分析，在每次异步复位后必须输入新的随机数，并建议在每次启动本模块运算前都输入新的随机数，输入的 64 位随机数不能为全 0。当然，即使不输入新的随机数也不会影响模块本身运算的正确性，只是可能会降低抵御差异功耗分析的能力。
- 5) 密钥写入分三种情况：如果密钥是 128bits，将密钥写入地址 0x4006_B020-0x4006_B02C；如果密钥是 192bits，将密钥写入地址 0x4006_B020-0x4006_B034；如果密钥是 256bits，将密钥写入地址 0x4006_B020-0x4006_B03C。
- 6) 判断模块运算结束的方法：不断读取 AES_CON.START，如果其值变为 0，则表示运算结束。
- 7) AES 最多工作 20MHz，所以当 AHB 总线时钟速度高于 20MHz 时，请将 AES_DIV_EN.DIV_EN 设置为 1（HCLK 2 分频）以保证该模块可以正常工作。

加密流程

- 1) 将 64 位随机数写入 AES_RNG 中（在有抗 DPA 要求的应用场合下，每次本模块复位后的第一次操作必须写入新的随机数，其余场合可以选择省略本步骤）。
- 2) 根据应用需求将加密密钥写入密钥寄存器(AES_KEY)中，除非复位或者重新写入，否则 AES_KEY 中的值将被保持。如果多次运算的密钥相同，则在第一次写入后即可忽略本步骤。
- 3) 将 128 位明文写入数据寄存器(AES_DATA)中。
- 4) 设置 AES_CON 中的各位，包括：
 - a) 根据需要设置控制寄存器中的 AES_CON.KEY_SEL；
 - b) 将 AES_CON.FUNC_SEL 设置为 2'b00；
 - c) 根据需要设置 AES_CON.DUMMY_SEL；
 - d) 向控制寄存器中的 AES_CON.START 写入 1，启动模块进行运算；以上的步骤 a, b, c, d 可同时进行。
- 5) 判断模块运算是否结束。

不断读取 AES_CON.START，如果其值变为 0，则表示运算结束。
- 6) 读取数据寄存器(AES_DATA)，获得 128 位密文。
- 7) 否则读取 AES_CON.DFA_STA，若 AES_CON.DFA_STA=0，则转到 8)，

若 AES_CON.DFA_STA=1，则进行错误处理。

8) 如果要继续进行新的运算，回到步骤 1)，否则结束。

标准解密流程

- 1) 将 64bits 随机数写入 AES_RNG 中（在有抗 DPA 要求的应用场合下，每次本模块复位后的第一次操作必须写入新的随机数，其余场合可以选择省略本步骤）。
- 2) 根据应用需求将解密密钥写入密钥寄存器(AES_KEY)中，除非复位或者重新写入，否则 AES_KEY 中的值将被保持。如果多次运算的密钥相同，则在第一次写入后即可忽略本步骤。

3) 将 128bits 密文写入数据寄存器(AES_DATA)中。

4) 设置 AES_CON 中的各位，包括：

- a) 根据需要设置控制寄存器中的 AES_CON.KEY_SEL;
- b) 将 AES_CON.FUNC_SEL 设置为 2'b01;
- c) 根据需要设置 AES_CON.DUMMY_SEL;
- d) 向控制寄存器中的 AES_CON.START 写入 1，启动模块进行运算

以上的步骤 a, b, c, d 可同时进行。

5) 判断模块运算是否结束。

不断读取 AES_CON.START，如果其值变为 0，则表示运算结束。

6) 读取数据寄存器(AES_DATA)，获得 128 位运算结果。

7) 否则读取 AES_CON.DFA_STA，若 AES_CON.DFA_STA=0，则转到 8)，

若 AES_CON.DFA_STA=1，则进行错误处理。

8) 如果要继续进行新的运算，回到步骤 1)，否则结束。

4 样例代码

4.1 代码介绍

用户可以根据上述工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载 AES 的样例代码直接使用 AES 驱动库提供的 API 函数进行加解密的应用。

以下部分简要介绍该代码的各个部分的功能：

1) AES 数据声明及初始化：

```
//AES TEST DATA INIT
const uint32_t u32AESTestData[4] = {0xAD6001EE, 0x82AB14FD, 0x97AB3599,
                                     0xFE5E2B5A};
const uint32_t u32AESTestKey[8] = {0x84448A5D, 0x3A53E5ED, 0x7D92105F,
                                   0xB2E52FD6, 0x84448A5D, 0x3A53E5ED,
                                   0x7D92105F, 0xB2E52FD6};
const uint32_t u32AESTestRand[4] = {0x12345678, 0x12345678};

static uint8_t pu8Ciphertext[16] = {0};
static uint8_t pu8Plaintext[16] = {0};

stc_aes_config_t stcAESConfig;
```

2) AES 结构体参数配置：

```
// AES Config
stcAESConfig.enDfaEn = AESDfaDisable;
stcAESConfig.enDiv = AESDiv2;
stcAESConfig.enDummySel = AESDummy7;
stcAESConfig.enKeySel = AESKey256;
stcAESConfig.pu8AESData = (uint8_t*)u32AESTestData;
stcAESConfig.pu8AESKey = (uint8_t*)u32AESTestKey;
stcAESConfig.pu8AESRand = (uint8_t*)u32AESTestRand;
```

3) AES 数据加密：

```
//encrypt
AES_Encrypt(&stcAESConfig, pu8Ciphertext);
```

4) AES 数据解密（此处使用上一步加密后的数据进行解密）：

```
//decrypt
stcAESConfig.pu8AESData = pu8Ciphertext;
AES_Decrypt(&stcAESConfig, pu8Plaintext);
```

通过以上代码即可完成一次 AES 数据的加密和解密。

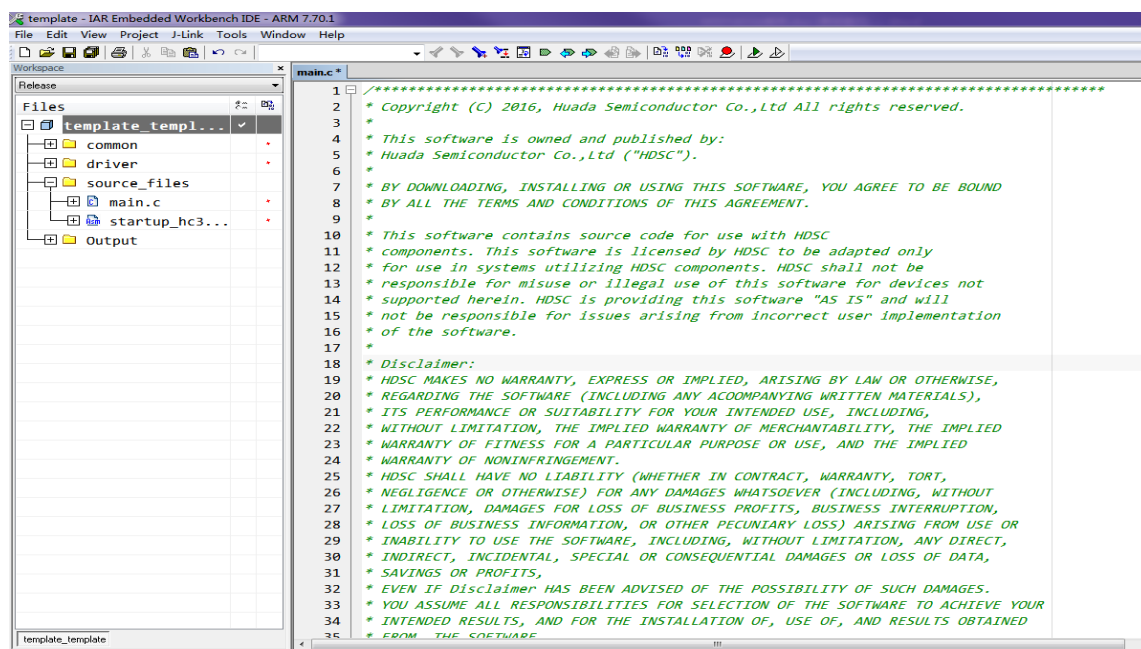
4.2 代码运行


用户可以通过华大半导体的网站下载到 AES 样例代码，并配合学习板运行相关代码学习使用 AES 模块。

以下部分主要介绍如何在学习板上运行 AES 样例代码并观察结果：

- 确认安装正确的 IAR（或 Keil，此处使用 IAR 做样例说明，Keil 中操作方法类似）工具（请从华大半导体完整下载相应的安装包，并参考用户手册进行安装）。
- 从华大半导体网站下载 AES 样例代码。
- 下载并运行样例代码：

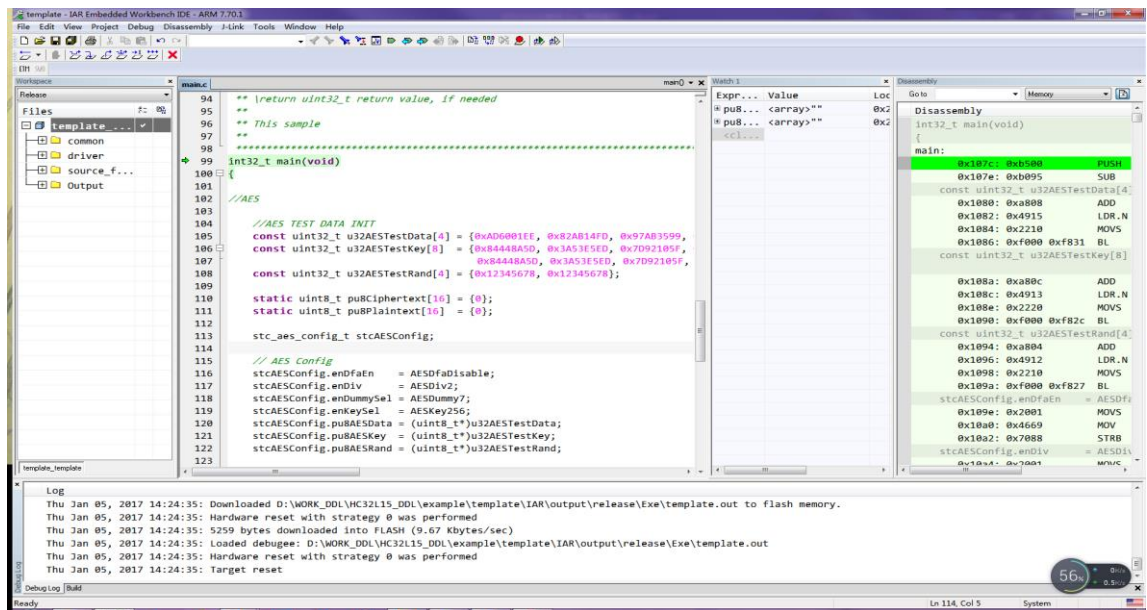
1) 打开 AES 项目，并打开‘main.c’如下视图：



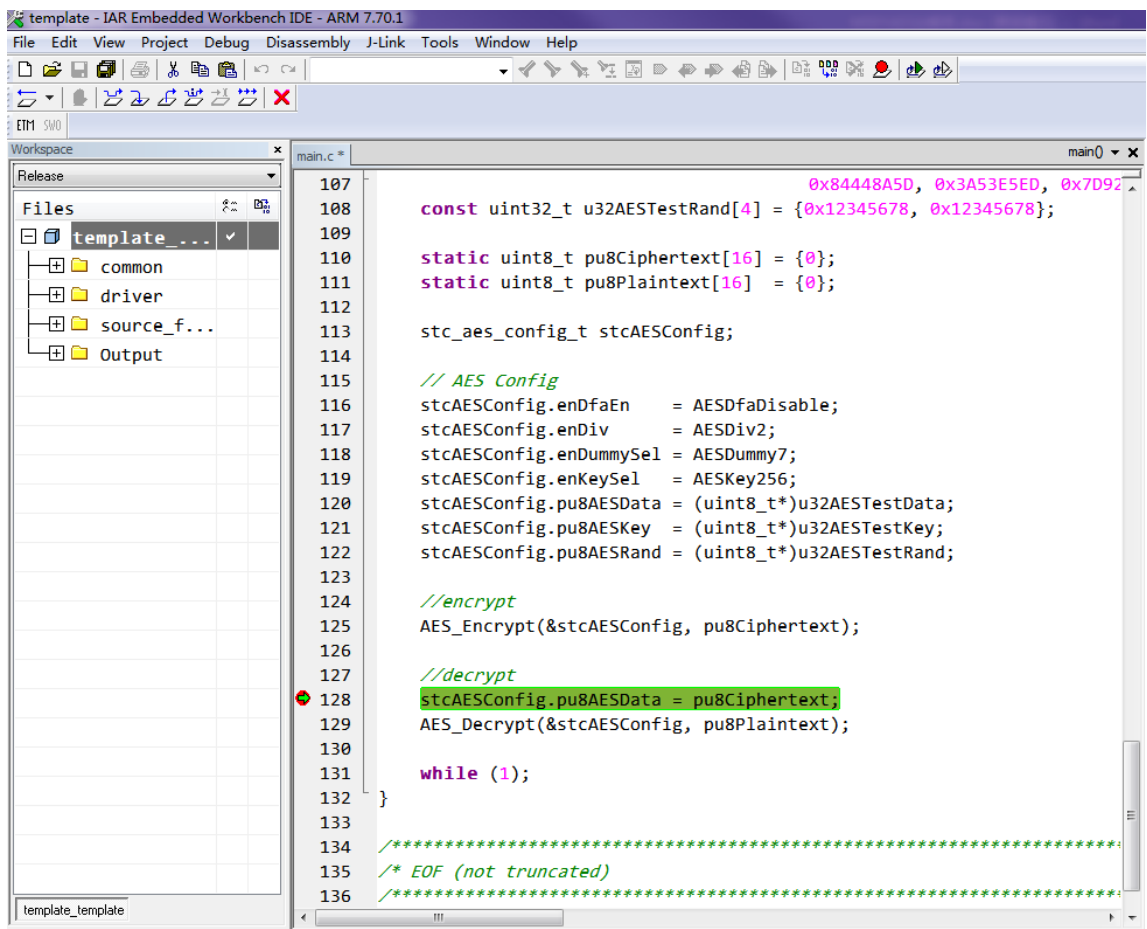
2) 点击  重新编译链接整个项目。

3) 点击  将代码下载到学习板上。

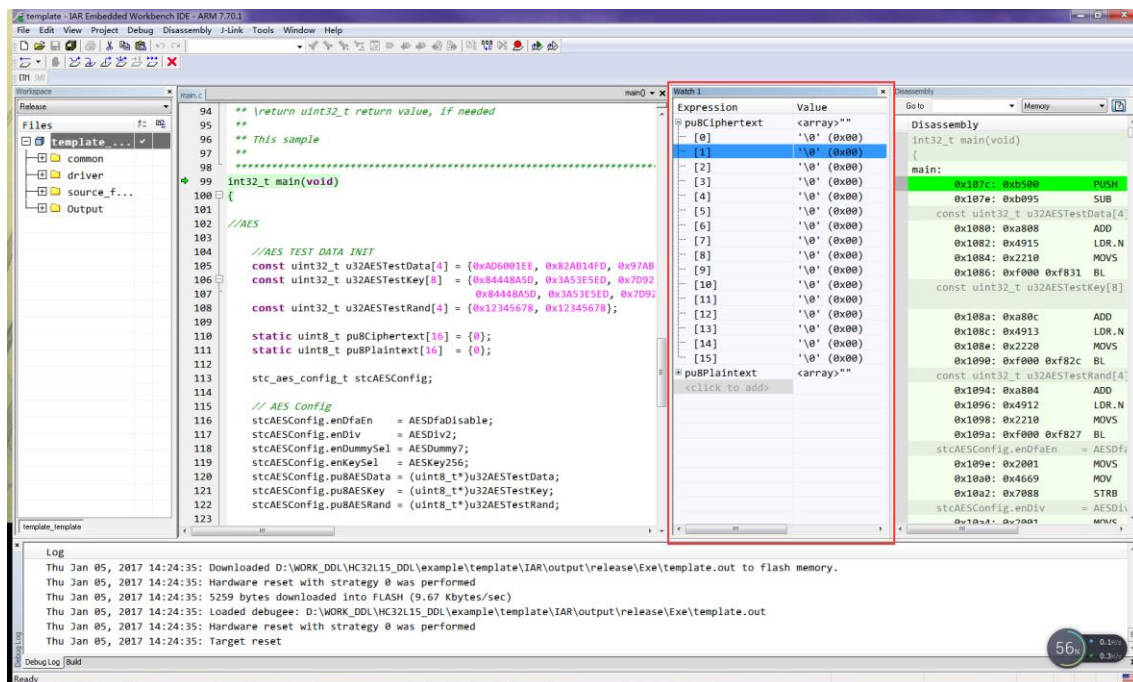
4) 可以看见类似如下的视图:




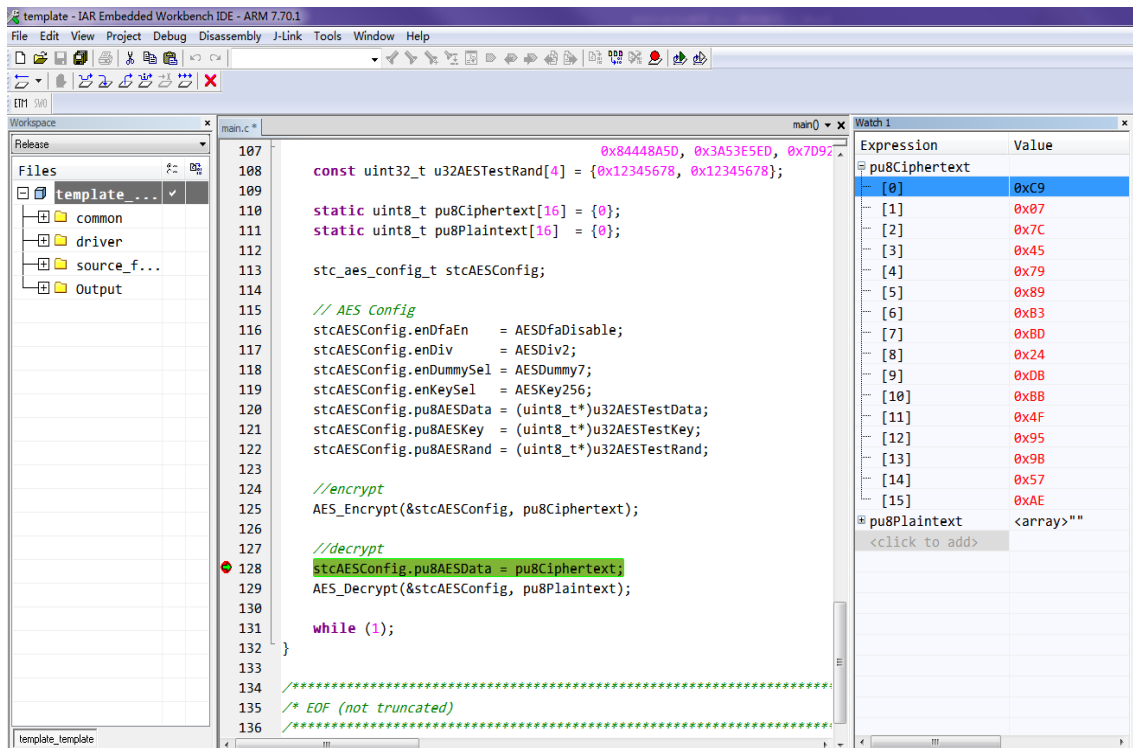
5) 在‘main.c’的加密函数后一行设置断点，如下图所示:




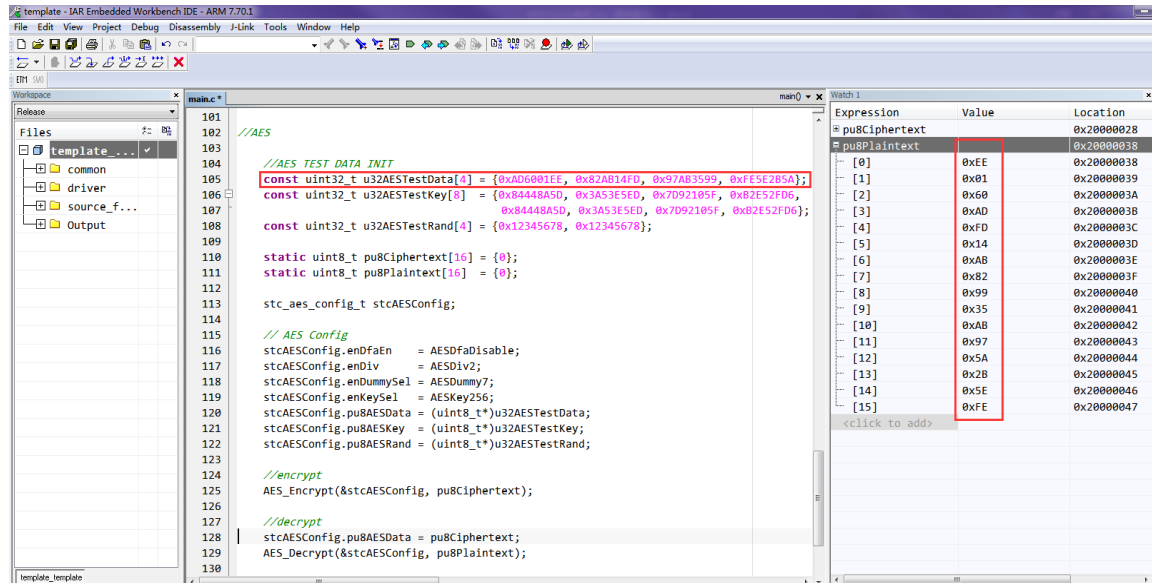
- 6) 点击“View -> Watch -> Watch1”打开一个‘watch1’窗口,并添加需要观测的变量来观测其数值, 如下图:



- 7) 点击  运行。
- 8) 代码运行并会停止在‘main.c’的断点处, 观察并记录密文‘pu8Ciphertext’的值, (此处可下载和使用其它同类 AES 加解密软件, 来验证该数据的正确性)。



- 9) 继续点击  运行，进行解密功能运算。
- 10) 此时可观察并记录明文‘pu8Plaintext’的值，（使用同一密钥解密的值应与解密前的数据即“u32AESTestData”一致），表示解密正确，如下图：



- 11) 运行完毕后可以关闭项目文件。
- 12) 用户亦可通过修改代码中 AES 的相关配置参数或初始化数据来进一步学习 AES 模块的功能。

5 总结

以上章节简要介绍了 HC32L15 系列的 AES，详细说明了 AES 模块的寄存器及操作流程，演示了如何使用相关的样例代码进行加解密，在实际开发中用户可以根据自己的需要配置和使用 AES 加解密功能。

6 版本信息 & 联系方式

日期	版本	修改记录
2018/8/9	Rev1.0	初版发布。



如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: mcu@hdsc.com.cn

网址: www.hdsc.com.cn

通信地址: 上海市张江高科园区碧波路 572 弄 39 号

邮编: 201203

