Application Notes
ZHCA816 – July 2018

使用 Unique ID 实现 C2000 代码加密的方法

Young Hu *North West China OEM Team*

摘要

在 TMS320F28x7x 和 F28004x 芯片内部的 UID_REGS 寄存器里面包含了一个 256 位的全球唯一 ID。全球唯一的 ID 可以用来做终端产品序列号、组网通信的节点地址或者芯片代码加密等功能。同时 TI 提供了基于 MSP430 的 AES-128 算法参考代码，本文将 AES-128 移植到 C2000 平台并且测试。最后本文给出了一种使用 Unique ID 和 AES-128 加密算法对程序加密及校验的方法，同时结合芯片内部 DCSM 加密模块，可以有效的对程序进行保护。

目录

1 C2000 Unique ID 介绍 2
2 代码加密及校验流程 2
2.1 无 Unique ID 加密的程序校验方法 2
2.2 基于高级加密算法的加密及校验方法 3
3 高级加密算法的选择 5
4 加密算法实现及加密流程测试 5
4.1 AES-128 算法移植 5
4.2 加密和解密工程实现及测试 6
5 小结 10
参考文献 11

图例

图 1 UID_REGS 寄存器描述 2
图 2 简单加密方式流程 3
图 3 用于加密的工程流程图 4
图 4 解密及校验的流程图 4
图 5 AES-128 对 UID_REGS 进行加密操作 5
图 6 TI AES-128 算法文件 6
图 7 读取出的 UID_REGS 内容 7
图 8 读取 UID_REGS 到数组变量中 8
图 9 AES-128 加密后的数据 9
图 10 AES-128 解密后的数据 10

1 C2000 Unique ID 介绍

在 TMS320F28x7x 和 F28004x 系列芯片内部的 OTP 中有 UID_REGS 寄存器作为器件唯一 ID。256 位的 UID_REGS 寄存器的前 192 位是伪随机数，后 32 位是全球唯一码，最后 32 位是前 224 位的 Fletcher's checksum。由于中间 32 位的 UID_UNIQUE 寄存器是全球唯一的，所以 256 位值也是唯一的。因此，用户可以利用该 ID 码，对芯片内部算法进行更严密更有效的保护。寄存器的描述和介绍如下所示。

Offset	Acronym	Register Name	Write Protection	Section
0h	UID_PSRAND0	UID Psuedo-random 192 bit number		Go
2h	UID_PSRAND1	UID Psuedo-random 192 bit number		Go
4h	UID_PSRAND2	UID Psuedo-random 192 bit number		Go
6h	UID_PSRAND3	UID Psuedo-random 192 bit number		Go
8h	UID_PSRAND4	UID Psuedo-random 192 bit number		Go
Ah	UID_PSRAND5	UID Psuedo-random 192 bit number		Go
Ch	UID_UNIQUE	UID Unique 32 bit number		Go
Eh	UID_CHECKSUM	UID Checksum		Go

图 1 UID_REGS 寄存器描述

2 代码加密及校验流程

2.1 Unique ID 加密的简单实现方法

使用 Unique ID 做程序加密最简单的方式就是如下图 1 所示，主要分两个阶段：（1）首次运行：将 UID_REGS 寄存器的值写入 Flash 执行区域，通过改写运行标志位，标记已经执行过首次运行操作；（2）非首次运行：读取存储到指定位置的 UID 的值与 UID_REGS 寄存器进行比较，如果一致，那么程序正常执行，如果不一致，那么进入错误处理程序，比如芯片自锁死、错误逻辑损坏外部电路等措施。

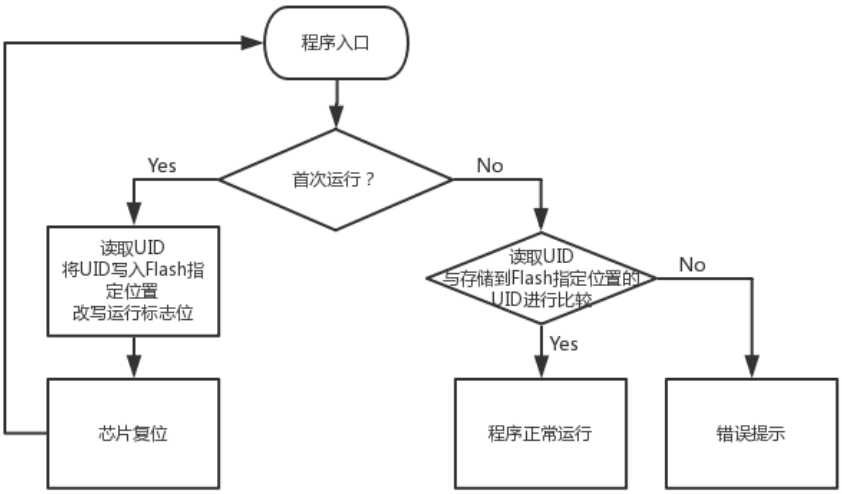


图 2 简单加密方式流程

但是这种方式存在较大的安全隐患，破解者可以通过仿真的方式（如果芯片没有使用 DCSM 加密的话）获得芯片的 UID_REGS 以及找到存储 UID 的 Flash 区域，进而修改烧写用于批量的二进制文件，即可实现破解。

2.2 基于高级加密算法的加密及校验方法

为了提高安全保密性能，用户可以选择采用高级加密算法对 Unique ID 进行加密，提高破解难度。同时为了避免破解者将加密算法从 Flash 中读取出来，所以本文采用加密和解密分离的方式。一个用于加密的工程将 UID_REGS 读出，运行加密算法，将加密后的数据写入到 Flash 中。一个正常量产的工程包含了正常的应用程序和解密算法。加密方式也不宜采用常见简单的加密规则，应尽可能采用特殊复杂的加密方式，使破解者不能迅速确定加密算法。。

用于加密的程序流程如图 3 所示，加密程序运行以后在 Flash 指定的位置会被写入加密的 UID 数据。当然，用于加密的程序也可以放在上位机中，通过修改二进制烧写文件，或者独立 bootloader 的方式将加密后的 UID 写入 Flash 中。

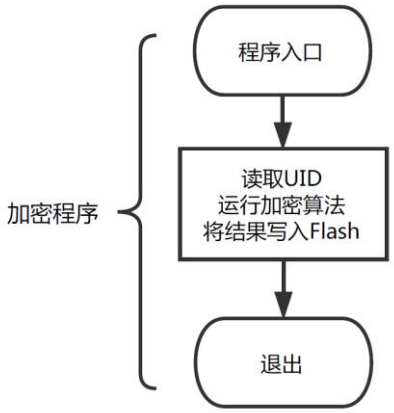


图 3 用于加密的工程流程图

正常量产的程序流程图如图 4 所示，读取出 Flash 中的加密后的 UID 数据，运行解密算法，得到解密的 UID 与当前芯片内部的 UID_REGS 进行比较校验，如果内容一致，那么执行正常应用程序。如果内容不一致，那么采取相应的措施。

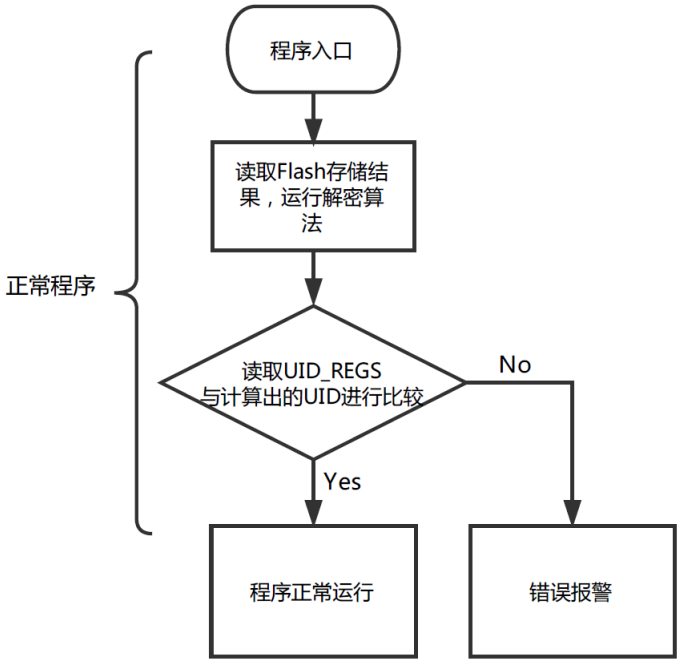


图 4 解密及校验的流程图

3 高级加密算法的选择

开发人员可以根据具体需求来选择加密算法，比如采用简单的移位取反的算法，也可以选择高级加密算法，如 DES、AES、RSA、TEA 等。本文中选择 AES-128 加密算法。AES 加密算法是密码学中的高级加密标准，该加密算法采用对称分组密码体制，密钥长度的最少支持为 128、192、256，分组长度 128 位，算法应易于各种硬件和软件实现。这种加密算法是美国联邦政府采用的区块加密标准，这个标准用来替代原先的 DES，已经被多方分析且广为全世界所使用。一个针对 AES 128 位密钥的攻击若“只”需要 2^120 计算复杂度（少于穷举法 2^128），128 位密钥的 AES 就算被破解了；而且该方法在目前还不实用。从应用的角度来看，这种程度的破解依然太不切实际。

由于 AES-128 是对 16 bytes（128bit）数据进行加密的过程。所以我们将 256 位的 UID_REGS 分成两部分：

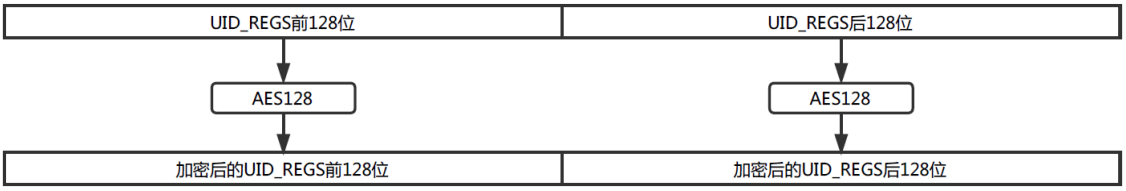


图 5 AES-128 对 UID_REGS 进行加密操作

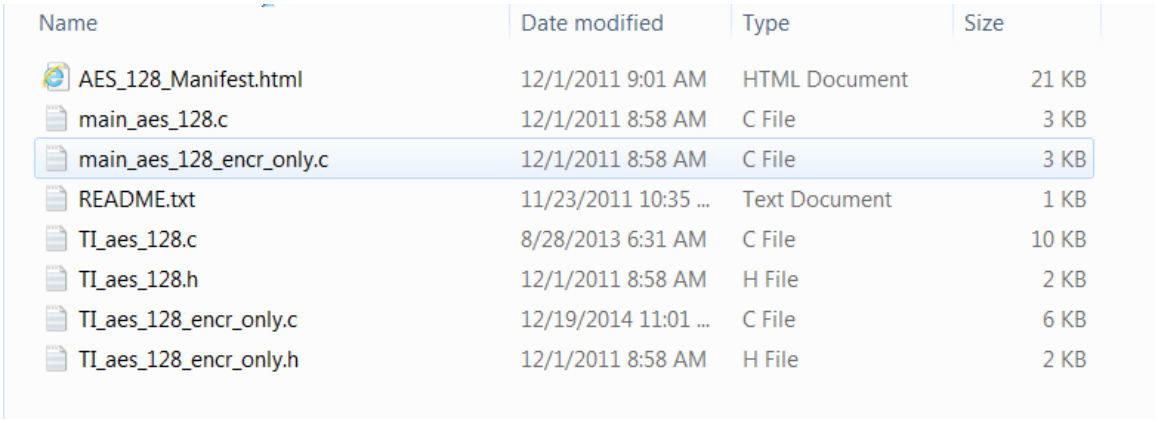
4 加密算法实现及加密流程测试

4.1 AES-128 算法移植

下述章节所介绍的算法开发均基于如下开发测试环境进行：

CCS Version	7.2.0.00013
Compiler Version	TI v16.9.3.LTS
测试 EVM 型号	F28377S LaunchPad

TI 官方提供了一个的基于 MSP430 的 AES-128 算法，链接地址参考本文参考文献链接。由于本设计中采用的是 F28377S，最小寻址是 16 位的，而 MSP430 的最小寻址为 8 位的，所以我们需要将算法移植到 C2000 平台上来。代码文件如下图所示：



Name	Date modified	Type	Size
AES_128_Manifest.html	12/1/2011 9:01 AM	HTML Document	21 KB
main_aes_128.c	12/1/2011 8:58 AM	C File	3 KB
main_aes_128_encr_only.c	12/1/2011 8:58 AM	C File	3 KB
README.txt	11/23/2011 10:35 ...	Text Document	1 KB
TI_aes_128.c	8/28/2013 6:31 AM	C File	10 KB
TI_aes_128.h	12/1/2011 8:58 AM	H File	2 KB
TI_aes_128_encr_only.c	12/19/2014 11:01 ...	C File	6 KB
TI_aes_128_encr_only.h	12/1/2011 8:58 AM	H File	2 KB

图 6 TI AES-128 算法文件

我们将其中的 TI_aes_128.c 和 TI_aes_128.h 复制到工程文件下面，对 **galois_mul2** 函数做如下修改，即可实现程序移植。

```
unsigned char galois_mul2(unsigned char value)
{
    if (value>>7)
    {
        return (((value << 1)^0x1b) & 0xFF);
    }
    else
    {
        return ((value << 1) & 0xFF);
    }
}
```

4.2 加密和解密工程实现及测试

加密程序核心代码如下所示：

```
    unsigned char key0[]    = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                               0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};

    unsigned char key1[]    = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                               0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};

    Uid_GetRegs(UID_Regs);          // 从OTP中读取UID_REGS寄存器的值
    AES_Enc(UID_Regs,key0);         // 对前128bit进行AES-128加密运算
    AES_Enc(&UID_Regs[16],key1); // 对后 128bit 进行 AES-128 加密运算
```

计算出来的结果储存在 UID_Regs[32]的数组中。随后调用 F021 Flash API 函数对预设的地址进行编程。

解密程序核心代码如下所示：

```
    unsigned char key0[]    = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                               0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};

    unsigned char key1[]    = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                               0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};

    Uid_GetEncRegs(UID_Regs);     //从Flash中读取加密后的UID_REGS寄存器值
    AES_Dec(UID_Regs,key0);       //对前128bit进行AES-128解密运算
    AES_Dec(&UID_Regs[16],key1); //对后 128bit 进行 AES-128 解密运算
    Uid_Compare(UID_Regs);        //与 OTP 中的 UID_REGS 进行比较
```

计算出来的结果储存在 UID_Regs[32]的数组中，随后读取 OTP 中的 UID_REGS 寄存器进行比较。随后进行其他处理。

另外可以根据根据需求选择 AES-128 的密钥（key0/key1）。

1. 将程序下载到 F28377S 的 Launchpad 中， UID_REGS 起始地址是 0x0703C0，使用 CCS 的 Memory Browser 功能进行查看。

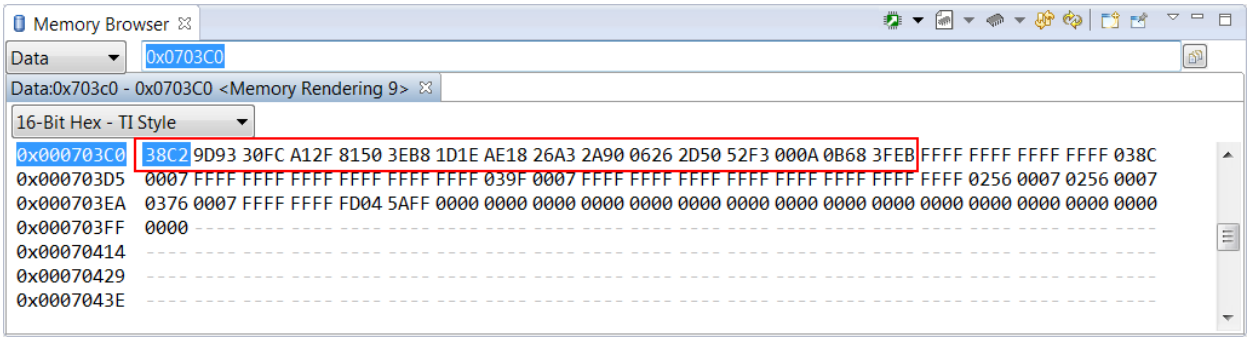


图 7 读取出的 UID_REGS 内容

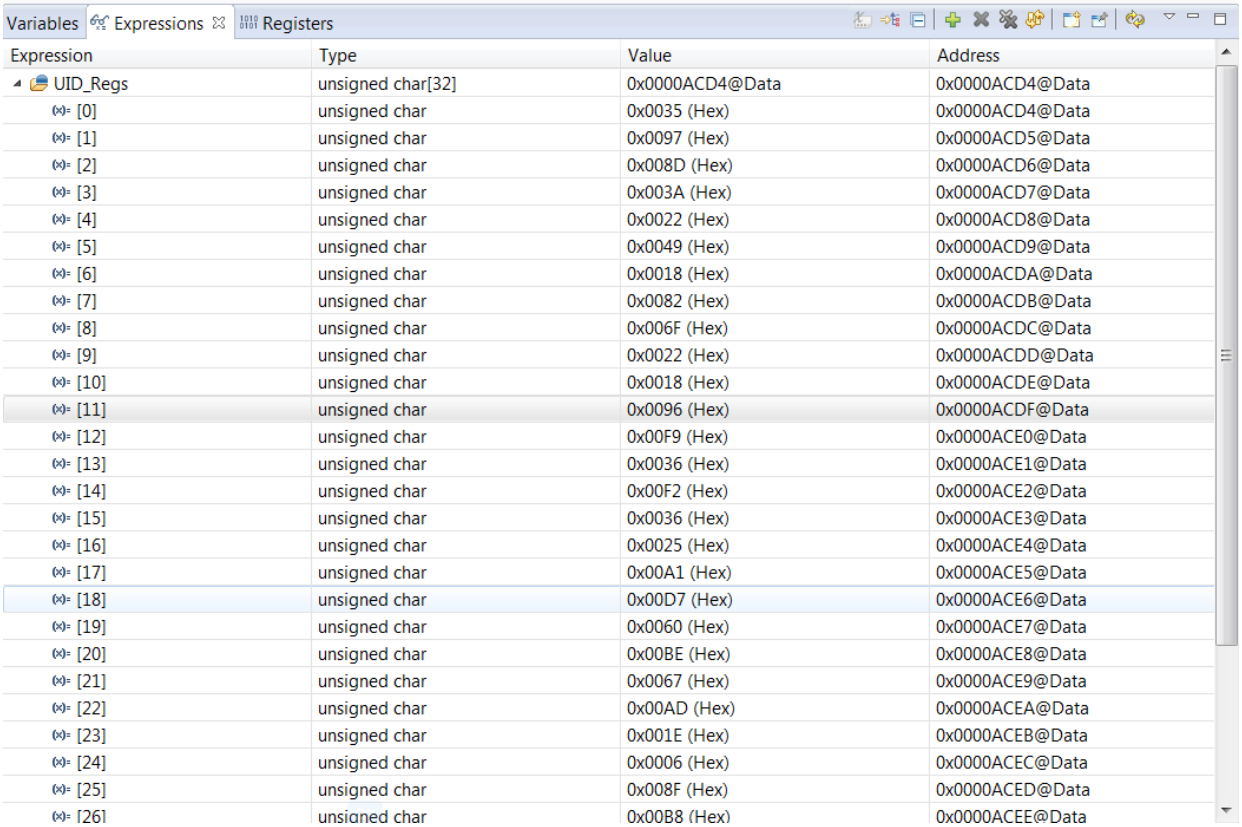
2. 将 UID_REGS 存放在 unsigned char 型的 UID_Regs 中，如下所示：

Variables | Expressions | Registers

Expression	Type	Value	Address
UID_Regs	unsigned char[32]	0x0000ACD4@Data	0x0000ACD4@Data
[0]	unsigned char	0x00C2 (Hex)	0x0000ACD4@Data
[1]	unsigned char	0x0038 (Hex)	0x0000ACD5@Data
[2]	unsigned char	0x0093 (Hex)	0x0000ACD6@Data
[3]	unsigned char	0x009D (Hex)	0x0000ACD7@Data
[4]	unsigned char	0x00FC (Hex)	0x0000ACD8@Data
[5]	unsigned char	0x0030 (Hex)	0x0000ACD9@Data
[6]	unsigned char	0x002F (Hex)	0x0000ACDA@Data
[7]	unsigned char	0x00A1 (Hex)	0x0000ACDB@Data
[8]	unsigned char	0x0050 (Hex)	0x0000ACDC@Data
[9]	unsigned char	0x0081 (Hex)	0x0000ACDD@Data
[10]	unsigned char	0x00B8 (Hex)	0x0000ACDE@Data
[11]	unsigned char	0x003E (Hex)	0x0000ACDF@Data
[12]	unsigned char	0x001E (Hex)	0x0000ACE0@Data
[13]	unsigned char	0x001D (Hex)	0x0000ACE1@Data
[14]	unsigned char	0x0018 (Hex)	0x0000ACE2@Data
[15]	unsigned char	0x00AE (Hex)	0x0000ACE3@Data
[16]	unsigned char	0x00A3 (Hex)	0x0000ACE4@Data
[17]	unsigned char	0x0026 (Hex)	0x0000ACE5@Data
[18]	unsigned char	0x0090 (Hex)	0x0000ACE6@Data
[19]	unsigned char	0x002A (Hex)	0x0000ACE7@Data
[20]	unsigned char	0x0026 (Hex)	0x0000ACE8@Data
[21]	unsigned char	0x0006 (Hex)	0x0000ACE9@Data
[22]	unsigned char	0x0050 (Hex)	0x0000ACEA@Data
[23]	unsigned char	0x002D (Hex)	0x0000ACEB@Data

图 8 读取 UID_REGS 到数组变量中

3. AES-128 加密后的数据如下所示：



Variables | Expressions | Registers

Expression	Type	Value	Address
UID_Regs	unsigned char[32]	0x0000ACD4@Data	0x0000ACD4@Data
[0]	unsigned char	0x0035 (Hex)	0x0000ACD4@Data
[1]	unsigned char	0x0097 (Hex)	0x0000ACD5@Data
[2]	unsigned char	0x008D (Hex)	0x0000ACD6@Data
[3]	unsigned char	0x003A (Hex)	0x0000ACD7@Data
[4]	unsigned char	0x0022 (Hex)	0x0000ACD8@Data
[5]	unsigned char	0x0049 (Hex)	0x0000ACD9@Data
[6]	unsigned char	0x0018 (Hex)	0x0000ACDA@Data
[7]	unsigned char	0x0082 (Hex)	0x0000ACDB@Data
[8]	unsigned char	0x006F (Hex)	0x0000ACDC@Data
[9]	unsigned char	0x0022 (Hex)	0x0000ACDD@Data
[10]	unsigned char	0x0018 (Hex)	0x0000ACDE@Data
[11]	unsigned char	0x0096 (Hex)	0x0000ACDF@Data
[12]	unsigned char	0x00F9 (Hex)	0x0000ACE0@Data
[13]	unsigned char	0x0036 (Hex)	0x0000ACE1@Data
[14]	unsigned char	0x00F2 (Hex)	0x0000ACE2@Data
[15]	unsigned char	0x0036 (Hex)	0x0000ACE3@Data
[16]	unsigned char	0x0025 (Hex)	0x0000ACE4@Data
[17]	unsigned char	0x00A1 (Hex)	0x0000ACE5@Data
[18]	unsigned char	0x00D7 (Hex)	0x0000ACE6@Data
[19]	unsigned char	0x0060 (Hex)	0x0000ACE7@Data
[20]	unsigned char	0x00BE (Hex)	0x0000ACE8@Data
[21]	unsigned char	0x0067 (Hex)	0x0000ACE9@Data
[22]	unsigned char	0x00AD (Hex)	0x0000ACEA@Data
[23]	unsigned char	0x001E (Hex)	0x0000ACEB@Data
[24]	unsigned char	0x0006 (Hex)	0x0000ACEC@Data
[25]	unsigned char	0x008F (Hex)	0x0000ACED@Data
[26]	unsigned char	0x00B8 (Hex)	0x0000ACEE@Data

图 9 AES-128 加密后的数据

4. AES-128 解密后的数据如下所示，解密后的结果与原始 UID_REGS 内容是一致的。

Variables | Expressions | Registers

Expression	Type	Value	Address
UID_Regs	unsigned char[32]	0x0000ACD4@Data	0x0000ACD4@Data
[0]	unsigned char	0x00C2 (Hex)	0x0000ACD4@Data
[1]	unsigned char	0x0038 (Hex)	0x0000ACD5@Data
[2]	unsigned char	0x0093 (Hex)	0x0000ACD6@Data
[3]	unsigned char	0x009D (Hex)	0x0000ACD7@Data
[4]	unsigned char	0x00FC (Hex)	0x0000ACD8@Data
[5]	unsigned char	0x0030 (Hex)	0x0000ACD9@Data
[6]	unsigned char	0x002F (Hex)	0x0000ACDA@Data
[7]	unsigned char	0x00A1 (Hex)	0x0000ACDB@Data
[8]	unsigned char	0x0050 (Hex)	0x0000ACDC@Data
[9]	unsigned char	0x0081 (Hex)	0x0000ACDD@Data
[10]	unsigned char	0x00B8 (Hex)	0x0000ACDE@Data
[11]	unsigned char	0x003E (Hex)	0x0000ACDF@Data
[12]	unsigned char	0x001E (Hex)	0x0000ACE0@Data
[13]	unsigned char	0x001D (Hex)	0x0000ACE1@Data
[14]	unsigned char	0x0018 (Hex)	0x0000ACE2@Data
[15]	unsigned char	0x00AE (Hex)	0x0000ACE3@Data
[16]	unsigned char	0x00A3 (Hex)	0x0000ACE4@Data
[17]	unsigned char	0x0026 (Hex)	0x0000ACE5@Data
[18]	unsigned char	0x0090 (Hex)	0x0000ACE6@Data
[19]	unsigned char	0x002A (Hex)	0x0000ACE7@Data
[20]	unsigned char	0x0026 (Hex)	0x0000ACE8@Data
[21]	unsigned char	0x0006 (Hex)	0x0000ACE9@Data
[22]	unsigned char	0x0050 (Hex)	0x0000ACEA@Data
[23]	unsigned char	0x002D (Hex)	0x0000ACEB@Data
[24]	unsigned char	0x00F3 (Hex)	0x0000ACEC@Data
[25]	unsigned char	0x0052 (Hex)	0x0000ACED@Data
[26]	unsigned char	0x000A (Hex)	0x0000ACEE@Data

图 10 AES-128 解密后的数据

5 小结

通过上述使用 C2000 的 Unique ID 和 AES-128 算法可以实现基于唯一 ID 的程序加密的方式。同时，C2000 内部还提供了 DCSM 模块，通过两个 128 位的 CSM 密码，对不同的存储区进行访问属性的设置，采用基于 Unique ID 的加密方式，可以将芯片破解变为几乎不可能的事情。

另外，还可以采取以下措施进行进一步的加密措施：（1）将解密算法进行多重加密，上电后将 Flash 中的解密算法进行解密后拷贝到 RAM 中执行；（2）密码不匹配的处理，可以让程序正确运行一段时间后再做失效处理，加大破解的时间成本；（3）在有组网通信的应用中，比如 MODBUS、CAN、Ethernet 等，可以将定期更新密钥的方式，通过通信主机下发密钥。这样可以进一步加大程序破解难度。

参考文献

[1]. TMS320F2837xS Delfino™ Microcontrollers datasheet. (SPRS881E)
[2]. TMS320F2837xS Delfino Microcontrollers Technical Reference Manual. (SPRUHX5E)
[3]. TMS320F2837xS Flash API Version 1.55 Reference Guide. (SPNU630A)
[4]. C2000™ Unique Device Number. (SPRACD0)
[5]. TI Advanced Encryption Standard. http://www.ti.com/tool/AES-128
[6]. 常用加密算法比较https://www.jianshu.com/p/3eae41356a24
[7]. 高级加密标准. 维基百科 https://zh.wikipedia.org/wiki/%E9%AB%98%E7%BA%A7%E5%8A%A0%E5%AF%86%E6%A0%87%E5%87%86

有关 TI 设计信息和资源的重要通知

德州仪器 (TI) 公司提供的技术、应用或其他设计建议、服务或信息，包括但不限于与评估模块有关的参考设计和材料（总称“TI 资源”），旨在帮助设计人员开发整合了 TI 产品的 应用； 如果您（个人，或如果是代表贵公司，则为贵公司）以任何方式下载、访问或使用了任何特定的 TI 资源，即表示贵方同意仅为该等目标，按照本通知的条款进行使用。

TI 所提供的 TI 资源，并未扩大或以其他方式修改 TI 对 TI 产品的公开适用的质保及质保免责声明；也未导致 TI 承担任何额外的义务或责任。TI 有权对其 TI 资源进行纠正、增强、改进和其他修改。

您理解并同意，在设计应用时应自行实施独立的分析、评价和 判断， 且应全权负责并确保 应用的安全性， 以及您的 应用 （包括应用中使用的所有 TI 产品）） 应符合所有适用的法律法规及其他相关要求。你就您的 应用声明，您具备制订和实施下列保障措施所需的一切必要专业知识，能够 (1) 预见故障的危险后果，(2) 监视故障及其后果，以及 (3) 降低可能导致危险的故障几率并采取适当措施。您同意，在使用或分发包含 TI 产品的任何 应用前， 您将彻底测试该等 应用 和该等应用所用 TI 产品的 功能而设计。除特定 TI 资源的公开文档中明确列出的测试外，TI 未进行任何其他测试。

您只有在为开发包含该等 TI 资源所列 TI 产品的 应用时， 才被授权使用、复制和修改任何相关单项 TI 资源。但并未依据禁止反言原则或其他法理授予您任何TI知识产权的任何其他明示或默示的许可，也未授予您 TI 或第三方的任何技术或知识产权的许可，该等产权包括但不限于任何专利权、版权、屏蔽作品权或与使用TI产品或服务的任何整合、机器制作、流程相关的其他知识产权。涉及或参考了第三方产品或服务的信息不构成使用此类产品或服务的许可或与其相关的保证或认可。使用 TI 资源可能需要您向第三方获得对该等第三方专利或其他知识产权的许可。

TI 资源系“按原样”提供。TI 兹免除对 TI 资源及其使用作出所有其他明确或默认的保证或陈述，包括但不限于对准确性或完整性、产权保证、无屡发故障保证，以及适销性、适合特定用途和不侵犯任何第三方知识产权的任何默认保证。

TI 不负责任何申索，包括但不限于因组合产品所致或与之有关的申索，也不为您辩护或赔偿，即使该等产品组合已列于 TI 资源或其他地方。对因 TI 资源或其使用引起或与之有关的任何实际的、直接的、特殊的、附带的、间接的、惩罚性的、偶发的、从属或惩戒性损害赔偿，不管 TI 是否获悉可能会产生上述损害赔偿，TI 概不负责。

您同意向 TI 及其代表全额赔偿因您不遵守本通知条款和条件而引起的任何损害、费用、损失和/或责任。

本通知适用于 TI 资源。另有其他条款适用于某些类型的材料、TI 产品和服务的使用和采购。这些条款包括但不限于适用于 TI 的半导体产品 (http://www.ti.com/sc/docs/stdterms.htm)、评估模块和样品 (http://www.ti.com/sc/docs/sampterms.htm) 的标准条款。

邮寄地址：上海市浦东新区世纪大道 1568 号中建大厦 32 楼，邮政编码：200122