加密引擎

目录

手册的本章包含以下主题：

注： 本系列参考手册章节旨在作为器件数据手册的补充资料。本文档适用于所有 dsPIC33E/PIC24 系列器件；但是，其中一些特性并非适用于所有器件。

请查询具体器件数据手册 **“加密引擎”** 章节开始处的注释，以查看本文档是否支持您当前使用的器件。

器件数据手册和系列参考手册的各章节均可从 Microchip 网站下载：http://www.microchip.com。

1.0 简介

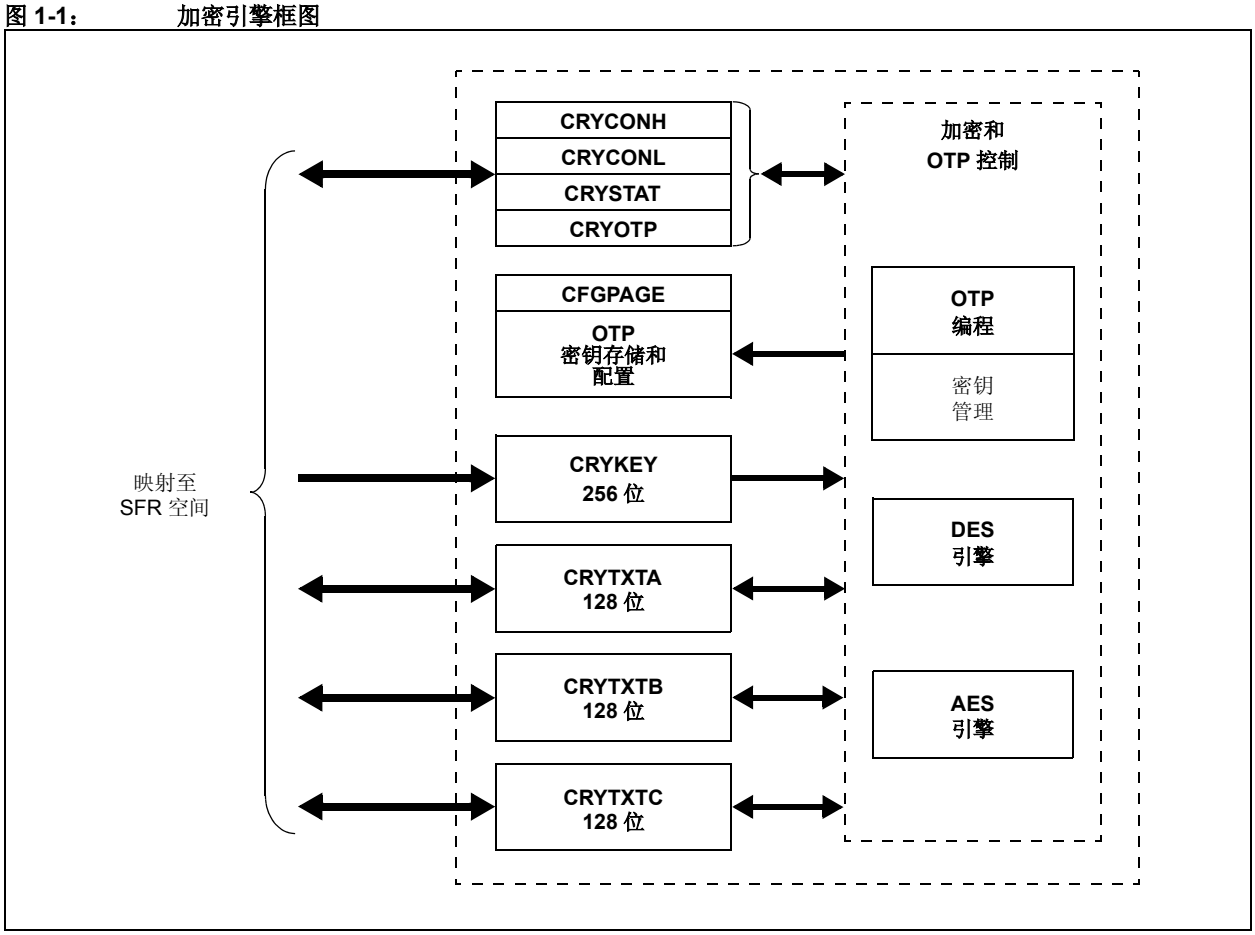
加密引擎为 dsPIC33/PIC24 器件提供一组新的数据安全选项。该引擎使用自己的独立状态机，可独立于 CPU 执行 NIST 标准数据加密和解密。从而无需担心在加强应用程序的安全性时，加密和解密额外需要的过多 CPU 或程序存储器开销。这样也就无需为新的应用程序开发相应的加密代码库。

主要特性包括：

- 存储器映射的 128 位和 256 位存储空间，用于加密 / 解密数据
- 多个密钥存储、选择和管理选项
- 支持内部现场保护
- 会话密钥加密和加载
- 半双工操作
- DES 和三重 DES （Triple DES， 3DES）加密和解密 （64 位块大小）：
  - 支持 64 位密钥和 2 密钥或 3 密钥三重 DES
- AES 加密和解密 （128 位块大小）：
  - 支持 128、 192 或 256 位密钥大小
- 支持 DES 和 AES 标准的 ECB、 CBC、 CFB、 OFB 和 CTR 模式
- 可编程安全密钥存储：
  - 512 字节 OTP 阵列用于密钥存储，不可从其他存储空间读取
  - 32 位配置页
  - 简单模块内编程接口
  - 支持密钥加密密钥 （Key Encryption Key， KEK）
- 支持伪随机数发生 （Pseudo-Random Number Generation， PRNG）（符合 NIST SP800-90）
- 支持随机数发生 （Random Number Generation， RNG）。

加密引擎的简化框图如图 1-1 所示。

图 1-1：　　加密引擎框图


CRYCONH
CRYCONL
CRYSTAT
CRYOTP
加密和
OTP 控制
CFGPAGE
OTP
密钥存储和
配置
OTP
编程
密钥
管理
映射至
SFR 空间
CRYKEY
256 位
DES
引擎
CRYTXTA
128 位
CRYTXTB
128 位
AES
引擎
CRYTXTC
128 位

2.0 寄存器

2.1 控制寄存器

加密引擎使用 4 个状态和控制寄存器：

- CRYCONH 和 CRYCONL
- CRYSTAT
- CRYOTP

CRYCON 寄存器（寄存器 2-1 和寄存器 2-2）为加密和解密操作设置所有参数，包括加密密钥的安全管理。通过 CRYCONL 寄存器使能该模块以及启动加密 / 解密操作。

CRYSTAT 寄存器 （寄存器 2-3）指示加密 / 解密操作的状态。它还包含了模块级中断标志。

CRYOTP 寄存器 （寄存器 2-4）用于控制安全 OTP 阵列的编程。**第 4.3.1 节 “存储的密钥 （安全 OTP 阵列）”** 对此进行了更详细地讨论。

尽管从技术层面来说，安全 OTP 阵列的 Page 0 不是一个寄存器，但它的作用就如同一个 OTP 配置熔丝寄存器。它既不是存储器映射到单片机的数据空间，也不可直接编程。寄存器 2-4 给出了其控制位的位置和功能。

2.1.1 复位状态和行为

与其他 dsPIC33/PIC24 外设相比，许多具有控制功能的位的行为方式可能与预期不同。除了常规系统复位外，大多数位还有额外的复位条件。其他位的值可能在初始化过程中发生改变；还有一些位在模块操作过程中被锁定，不会改变。

在寄存器说明和本章中对具体行为进行了注释说明。关于复位行为的具体信息也可以参见**第 6.0 节 “复位的影响”**。

2.2 数据寄存器空间

除了控制寄存器外，还有 4 个寄存器空间用于加密数据和密钥存储：

- CRYTXTA
- CRYTXTB
- CRYTXTC
- CRYKEY

虽然所有这些数据空间均映射至 SFR 空间，但实际实现为 128 位或 256 位宽阵列，而不是 16 位宽数据寄存器组。和 SFR 空间中的任何其他寄存器一样，对这些阵列的读和写操作均自动执行。

CRYTXTA 至 CRYTXTC 为 128 位宽空间，用于对加密引擎写入 / 读取数据。另外，它们还用于存储加密 / 解密操作的中间结果。当模块在执行操作 （CRYGO = 1）时，不能写入这些寄存器。

CRYTXTA 和 CRYTXTB 通常充当加密 / 解密过程的输入。 CRYTXTA 通常包含要加密的初始明文或要解密的初始密文。根据操作模式， CRYTXTB 可能包含密文输出或中间密文数据。在某些操作中它还可充当长度可编程的计数器。

CRYTXTC 主要用于存储加密 / 解密操作的最终输出。它还用作输入寄存器，包含要编程到安全 OTP 阵列中的数据。

CRYKEY 为 256 位宽的空间，用于存储所选操作的加密密钥。可从 SFR 空间和安全 OTP 阵列对其进行写操作。虽然 CRYKEY 映射到 SFR 空间，但为只写存储区域；此处存储的任何数据，无论其来源为何，均无法通过任何运行时操作读回。该特性有助于确保所有密钥数据的安全。

寄存器 2-1：　CRYCONH：加密控制寄存器高位字

U-0	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)
—	CTRSIZE6(2,3)	CTRSIZE5(2,3)	CTRSIZE4(2,3)	CTRSIZE3(2,3)	CTRSIZE2(2,3)	CTRSIZE1(2,3)	CTRSIZE0(2,3)
bit 15							bit 8

R/W-0(1)	R/W-0(1)	R/W-0(1)	U-0	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)
SKEYSEL	KEYMOD1(2)	KEYMOD0(2)	—	KEYSRC3(2)	KEYSRC2(2)	KEYSRC1(2)	KEYSRC0(2)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15　**未实现：**读为 0

bit 14-8　**CTRSIZE<6:0>：**计数器大小选择位 (1,2,3)

计数器定义为 CRYTXTB<n:0>，其中 n = CTRSIZE。每个操作后计数器递增 1，并在计数器从 （2^{n-1}-1）计满返回到 0 时发生计满返回事件。

1111111 = 128 位 （CRYTXTB<127:0>）
1111110 = 127 位 （CRYTXTB<126:0>）
•
•
0000010 = 3 位 （CRYTXTB<2:0>）
0000001 = 2 位 （CRYTXTB<1:0>）
0000000 = 1 位 （CRYTXTB<0>）；当 CRYTXTB<0> 从 1 切换到 0 时发生计满返回事件

bit 7　**SKEYSEL：**会话密钥选择位 (1)

1 = 使用 CRYKEY<255:128> 执行密钥生成 / 加密 / 加载
0 = 使用 CRYKEY<127:0> 执行密钥生成 / 加密 / 加载

bit 6-5　**KEYMOD<1:0>：**AES/DES 加密 / 解密密钥模式 / 密钥长度选择位 (1,2)

对于 DES 加密 / 解密操作 （CPHRSEL = 0）：
11 = 64 位， 3 密钥 3DES
10 = 保留
01 = 64 位，标准 2 密钥 3DES
00 = 64 位 DES
对于 AES 加密 / 解密操作 （CPHRSEL = 1）：
11 = 保留
10 = 256 位 AES
01 = 192 位 AES
00 = 128 位 AES

bit 4　**未实现：**读为 0

bit 3-0　**KEYSRC<3:0>：**加密密钥源位 (1,2)

请参见表 4-1 和表 4-2 获取 KEYSRC<3:0> 值。

注　1：这些位在系统复位或 CRYMD 位置 1 时复位。
2：只要某个操作正在进行中 （CRYGO 位置 1），对这些位域的写操作就会被锁定。
3：仅在 CRYTXTB 用作计数器时在 CTR 操作中使用；否则，这些位无任何作用。

寄存器 2-2：　CRYCONL：加密控制寄存器低位字

R/W-0	U-0	R/W-0	R/W-0(1)	R/W-0(1)	R/W-0(1)	U-0	R/W-0, HC(1)
CRYON	—	CRYSIDL	ROLLIE	DONEIE	FREEIE	—	CRYGO
bit 15							bit 8

R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)
OPMOD3(2)	OPMOD2(2)	OPMOD1(2)	OPMOD0(2)	CPHRSEL(2)	CPHRMOD2(2)	CPHRMOD1(2)	CPHRMOD0(2)
bit 7							bit 0

图注：	HC = 可由硬件清零的位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15　**CRYON**：加密引擎使能位
1 = 使能模块
0 = 禁止模块

bit 14　**未实现**：读为 0

bit 13　**CRYSIDL**：加密引擎空闲模式停止控制位
1 = 在空闲模式下停止模块操作
0 = 在空闲模式下继续模块操作

bit 12　**ROLLIE**：CRYTXTB 计满返回中断允许位 (1)
1 = 当 CRYTXTB 的计数器部分计满返回到 0 时发生中断事件
0 = 当 CRYTXTB 的计数器部分计满返回到 0 时不发生中断事件

bit 11　**DONEIE**：操作完成中断允许位 (1)
1 = 当前加密操作完成时发生中断事件
0 = 当前加密操作完成时不发生中断事件；软件必须查询 START 或 BUSY 位以确定当前加密操作何时完成

bit 10　**FREEIE**：输入文本中断允许位 (1)
1 = 当前加密操作期间使用输入文本 （明文或密文）时发生中断事件
0 = 使用输入文本时不发生中断事件

bit 9　**未实现**：读为 0

bit 8　**CRYGO**：加密引擎启动位 (1)
1 = 启动 OPMOD<3:0> 指定的操作 （操作完成时自动清零）
0 = 停止当前操作 （由软件清零时）；还指示当前操作已完成 （由硬件清零时）

注　1：这些位在系统复位或 CRYMD 位置 1 时复位。
2：只要某个操作正在进行中 （CRYGO 位置 1），对这些位域的写操作就会被锁定。

寄存器 2-2：　CRYCONL：加密控制寄存器低位字 （续）

bit 7-4　**OPMOD<3:0>：**操作模式选择位 (1,2)

1111 = 加载会话密钥 （使用密钥加密密钥解密 CRYTXTA/CRYTXTB 中的会话密钥并写入 CRYKEY）
1110 = 加密会话密钥 （使用密钥加密密钥加密 CRYKEY 中的会话密钥并写入 CRYTXTA/CRYTXTB）
1011 = 保留
1010 = 生成一个随机数
•
•
•
0100
0011
0010 = AES 解密密钥扩展
0001 = 解密
0000 = 加密

bit 3　**CPHRSEL：**加密引擎选择位 (1,2)

1 = AES 引擎
0 = DES 引擎

bit 2-0　**CPHRMOD<2:0>：**加密模式位 (1,2)

11x = 保留
101 = 保留
100 = 计数器 （CTR）模式
011 = 输出反馈 （OFB）模式
010 = 加密反馈 （CFB）模式
001 = 加密块链接 （CBC）模式
000 = 电子密码本 （ECB）模式

注　1： 这些位在系统复位或 CRYMD 位置 1 时复位。
2： 只要某个操作正在进行中 （CRYGO 位置 1），对这些位域的写操作就会被锁定。

寄存器 2-3：　CRYSTAT：加密状态寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/HSC-x(1)	R/HSC-0(1)	R/C-0,HS(2)	R/C-0, HS(2)	U-0	R/HSC-0(1)	R/HSC-x(1)	R/HSC-x(1)
CRYBSY(3)	TXTABSY	CRYABRT	ROLLOVR	—	MODFAIL(4)	KEYFAIL(3,4)	PGMFAIL(3,4)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
HS = 可由硬件置 1 的位	C = 可清零位	HSC = 硬件置 1/ 清零位	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 复位状态条件位

bit 15-8　**未实现：**读为 0

bit 7　**CRYBSY：**加密引擎忙状态位 (1,3)
1 = 正在进行加密操作
0 = 未在进行加密操作

bit 6　**TXTABSY：**CRYTXTA 忙状态位 (1)
1 = CRYTXTA 寄存器忙，且不可写入
0 = CRYTXTA 空闲且可写入

bit 5　**CRYABRT：**加密操作中止状态位 (2)
1 = 用软件清零 CRYGO 位中止上一个操作 （在该位被清零之前，无法执行其他操作）
0 = 上一个操作正常完成 （CRYGO 由硬件清零）

bit 4　**ROLLOVR：**计数器计满返回状态位 (2)
1 = 执行上一个 CTR 模式操作时 CRYTXTB 计数器计满返回
0 = 未发生计满返回事件

bit 3　**未实现：**读为 0

bit 2　**MODFAIL：**模式配置故障标志位 (1,4)
1 = 当前所选操作和密码模式配置无效；在选择有效模式前无法将 PGM 位置 1（使用任何有效配置时，由硬件自动清零）
0 = 当前所选操作和密码模式配置有效

bit 1　**KEYFAIL：**密钥配置故障状态位 (1,3,4)
请参见表 4-1 和表 4-2 了解无效密钥配置。
1 = 当前所选密钥和模式配置无效；在选择有效模式前无法将 PGM 位置 1（使用任何有效配置时，由硬件自动清零）
0 = 当前所选配置有效

bit 0　**PGMFAIL：**密钥存储 / 配置编程配置故障标志位 (1,3,4)
1 = KEYPG<3:0> 指示的页保留或锁定；无法将 PGW 位置 1 且不可启动任何编程操作
0 = 可对 KEYPG<3:0> 指示的页编程

注　1： 这些位在系统复位或 CRYMD 位置 1 时复位。
2： 这些位在系统复位、 CRYMD 位置 1 或 CRYGO 清零时复位。
3： 这些位在所有 OTP 读操作期间自动置 1，包括 POR 时的初始读操作。读操作完成后，该位呈现适当的状态，反映当前配置。
4： 即使模块禁止 （CRYON = 0），这些位仍起作用。这允许在使能模块前验证模式配置的兼容性。

寄存器 2-4：　CRYOTP：加密 OTP 页编程控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/HSC-x(1)	R/W-0(1)	R/S-1, HC(2)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/S-0, HC(3)
PGMTST	OTPIE	CRYREAD(4,5)	KEYPG3	KEYPG2	KEYPG1	KEYPG0	CRYWR(4,5)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
S = 只可置 1 位	HC = 可由硬件清零的位	HSC = 硬件置 1/ 清零位	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-8　**未实现：**读为 0

bit 7　**PGMTST：**密钥存储 / 配置编程测试位 (1)

该位反映 TSTPGM 位的状态，并在编程后用于测试对安全 OTP 阵列的编程。
1 = TSTPGM （CFGPAGE<30>）已编程 （1）
0 = TSTPGM 未编程 （0）

bit 6　**OTPIE：**密钥存储 / 配置编程中断允许位 (1)

1 = 当前编程或读操作完成时产生中断
0 = 当前编程或读操作完成时不产生中断；软件必须查询PGM、READ或CRYBSY位以确定当前编程操作何时完成

bit 5　**CRYREAD：**加密密钥存储 / 配置读取位 (2,4,5)

1 = 正在进行读操作 （CRYGO = 1 时；当操作完成时由硬件自动清零）
0 = 读操作已完成

bit 4-1　**KEYPG<3:0>：**密钥存储 / 配置编程页选择位 (1)

1111= 保留
•
•
•
1001
1000 = OTP Page 8
0111 = OTP Page 7
0110 = OTP Page 6
0101 = OTP Page 5
0100 = OTP Page 4
0011 = OTP Page 3
0010 = OTP Page 2
0001 = OTP Page 1
0000 = 配置页 （CFGPAGE）； OTP Page 0

bit 0　**CRYWR：**加密密钥存储 / 配置编程位 (3,4,5)

1 = 使用 CRYTXTC<63:0> 中的值编程密钥存储 / 配置位
0 = 编程操作已完成

注　1：这些位在系统复位或 CRYMD 位置 1 时复位。
2：此位仅在系统复位时复位。
3：这些位在系统复位、 CRYMD 位置 1 或 CRYGO 清零时复位。
4：仅当 CRYON = 1 且 CRYGO = 0 时，该位才置 1。不要在任何给定时间将 CRYREAD 和 CRYWR 同时置 1。
5：当 CRYON 或这些位置 1 时，请勿将其清零；始终允许硬件操作完成并自动清零该位。

寄存器 2-5：　CFGPAGE：加密安全阵列配置寄存器 （OTP Page 0）

R-x	R/P-x	U-x	U-x	R/P-x	R/P-x	R/P-x	R/P-x
r	TSTPGM(1)	—	—	KEY7TYPE1	KEY7TYPE0	KEY5TYPE1	KEY5TYPE0
bit 31							bit 24

R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x
KEY3TYPE1	KEY3TYPE0	KEY1TYPE1	KEY1TYPE0	SKEYEN	LKYSRC7	LKYSRC6	LKYSRC5
bit 23							bit 16

R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x
LKYSRC4	LKYSRC3	LKYSRC2	LKYSRC1	LKYSRC0	SRCLCK	WRLOCK8	WRLOCK7
bit 15							bit 8

R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x	R/P-x
WRLOCK6	WRLOCK5	WRLOCK74	WRLOCK3	WRLOCK2	WRLOCK1	WRLOCK0	SWKYDIS
bit 7							bit 0

图注：	r = 保留位		
R = 可读位	P = 一次编程位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31　**保留：**不要修改

bit 30　**TSTPGM：**客户编程测试位 (1)
1 = 已编程 CFGPAGE
0 = 未编程 CFGPAGE

bit 29-28　**未实现：**读为 0

bit 27-26　**KEY7TYPE<1:0>：**密钥类型选择位 （OTP Page 7 和 8）
11 = 仅用于 192 位 /256 位 AES 操作
10 = 仅用于 128 位 AES 操作
01 = 仅用于 DES3 操作
00 = 仅用于 DES/DES2 操作

bit 25-24　**KEY5TYPE<1:0>：**密钥类型选择位 （OTP Page 5 和 6）
11 = 仅用于 192 位 /256 位 AES 操作
10 = 仅用于 128 位 AES 操作
01 = 仅用于 DES3 操作
00 = 仅用于 DES/DES2 操作

bit 23-22　**KEY3TYPE<1:0>：**密钥类型选择位 （OTP Page 3 和 4）
11 = 仅用于 192 位 /256 位 AES 操作
10 = 仅用于 128 位 AES 操作
01 = 仅用于 DES3 操作
00 = 仅用于 DES/DES2 操作

bit 21-20　**KEY1TYPE<1:0>：**密钥类型选择位 （OTP Page 1 和 2）
11 = 仅用于 192 位 /256 位 AES 操作
10 = 仅用于 128 位 AES 操作
01 = 仅用于 DES3 操作
00 = 仅用于 DES/DES2 操作

bit 19　**SKEYEN：**会话密钥使能位
1 = 存储的 Key #1 仅可用作密钥加密密钥
0 = 存储的 Key #1 可用于任何操作

注　1： 该位的状态通过 PGMTST 位 （CRYOTP<7>）反映。

寄存器 2-5： CFGPAGE：加密安全阵列配置寄存器 （OTP Page 0）（续）

bit 18-11 **LKYSRC<7:0>：**锁定密钥源配置位

如果 SRCLCK = 1：

1xxxxxxx = 密钥源和 KEYSRC<3:0> = 1111 时一样
01xxxxxx = 密钥源和 KEYSRC<3:0> = 0111 时一样
001xxxxx = 密钥源和 KEYSRC<3:0> = 0110 时一样
0001xxxx = 密钥源和 KEYSRC<3:0> = 0101 时一样
00001xxx = 密钥源和 KEYSRC<3:0> = 0100 时一样
000001xx = 密钥源和 KEYSRC<3:0> = 0011 时一样
0000001x = 密钥源和 KEYSRC<3:0> = 0010 时一样
00000001 = 密钥源和 KEYSRC<3:0> = 0001 时一样
00000000 = 密钥源和 KEYSRC<3:0> = 0000 时一样

如果 SRCLCK = 0：

忽略这些位。

bit 10 **SRCLCK：**密钥源锁定位

1 = 密钥源由 LKYSRC<7:0> 位决定 （禁止软件密钥选择）
0 = 密钥源由 KEYSRC<3:0> 位 （CRYCONH<3:0>）决定 （禁止锁定密钥选择）

bit 9-1 **WRLOCK<8:0>：**写锁定页使能位

对于 OTP Page 0 （CFGPAGE）至 8：

1 = OTP 页永久锁定且不可编程
0 = OTP 页未锁定且可编程

bit 0 **SWKYDIS：**软件密钥禁止位

1 = 禁止软件密钥（CRYKEY 寄存器）；当 KEYSRC<3:0> = 0000 时，KEYFAIL 状态位将置 1 且不启动任何加密 / 解密 / 会话密钥操作直至 KEYSRC<3:0> 位更改为 0000 以外的某个值
0 = 当 KEYSRC<3:0> = 0000 时软件密钥 （CRYKEY 寄存器）可用作密钥源

注 1： 该位的状态通过 PGMTST 位 （CRYOTP<7>）反映。

3.0 工作原理

本章不再对加密和加密技术进行简要介绍。而是简要概述通过加密引擎及其许多变体所实现的具体加密和解密方法。此处对 AES 和 DES 加密标准的内部细节不作讨论。

对于想要了解更完整的背景信息和其他信息的用户，请参见**第 8.0 节 “相关参考资料”**中列出的其他参考资料。

3.1 对称密码

需要了解的现代密码中的最简单一种是**对称密码**。此类密码将明文消息与**密钥**（有时也被称为**私钥**）结合起来，产生加密密文。需要知道密钥才能解密密文，这种加密方式称为对称加密。从技术层面来说，对称加密算法不要求加密和解密使用相同的密钥，不过即使使用不同的密钥，也可从一个密钥简单推导出另一个密钥 （即，将一个固定数字加到加密密钥上来获得解密密钥）。

对称密码可进一步分为**块密码** （在**第 3.2 节 “块密码”**中讨论）和**流密码** （在**第 3.3 节 “流密码”**中讨论）。

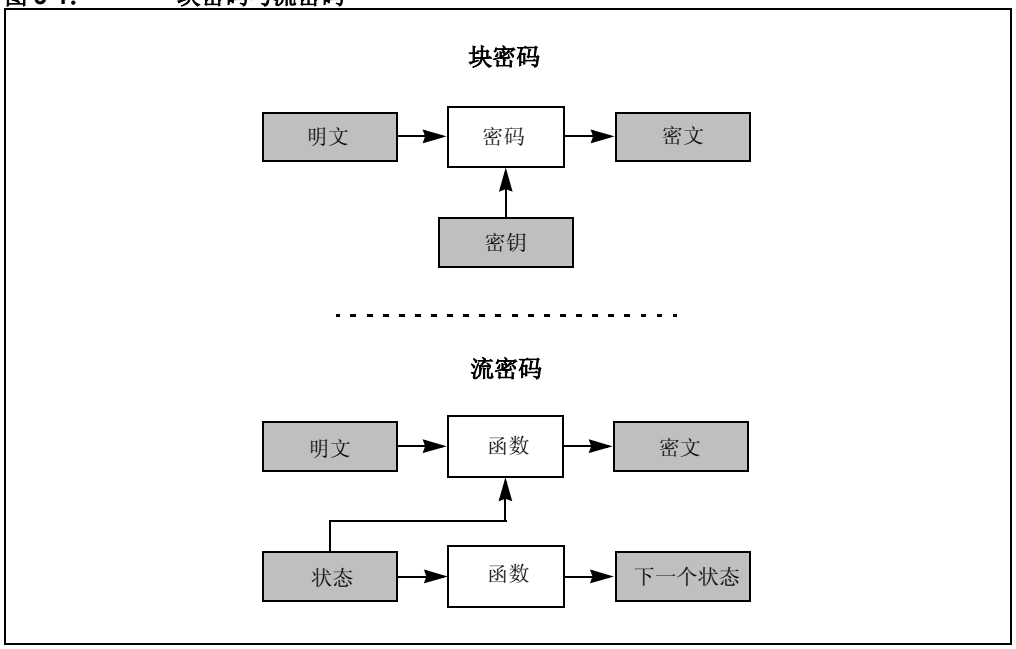
3.1.1 块密码与流密码

当同时使用块密码和流密码（也称为状态密码）创建一个明文消息的密文消息时，两者有一些主要的不同。块密码将相同的加密函数应用于每个明文块，如图 3-1 所示。每一块明文（即，一个“块”）与流密码中的对应明文相比较大。只要有明文消息、一个密钥和一个加密函数就可以创建一条密文消息。

另一方面，流密码则需要明文消息，当前的流 “状态”，一个用于创建下一个 “状态”的函数，以及一个根据明文消息和当前状态创建密文消息的函数。被变换的明文碎片通常很小，甚至小到 1 个比特，“流密码”因此得名。

长期以来，流密码的数学算法演变得更加简单，因此可用硬件更快速轻松地实现。著名的 ARCFOUR（RC4）密码就是流密码的一个例子。流密码的安全性依赖于其中一个或两个函数的加密安全。

图 3-1： 块密码与流密码



3.2 块密码

块密码对固定长度的明文块进行加密以产生相同长度的密文块。给定一个具体的密钥，明文块和密文块之间则存在 1 对 1 的对应关系。因此，块密码有时称为**密码本**，因为从理论上讲，针对每个密钥值，有可能将密码实现为一个非常大的查找表。对于现代块加密算法而言，这当然是不现实的，因此将其实现为非常适合在计算机上实现的数学运算序列。著名的块密码包括**数据加密标准（Data Encryption Standard，DES）**、**高级加密标准（Advanced Encryption Standard，AES）**和 **Blowfish**。

3.2.1 电子密码本模式

到目前为止我们所讨论的块密码的简单用法称为电子密码本（ECB）模式，如图 3-2 和图 3-3 所示。公式 3-1 给出了该运算的正式表述。

公式 3-1： ECB 加密和解密

加密：

$C_i = E_K(P_i)$

（使用密钥 K 加密明文块 i 从而产生密文块 i）。

解密：

$P_i = D_K(C_i)$

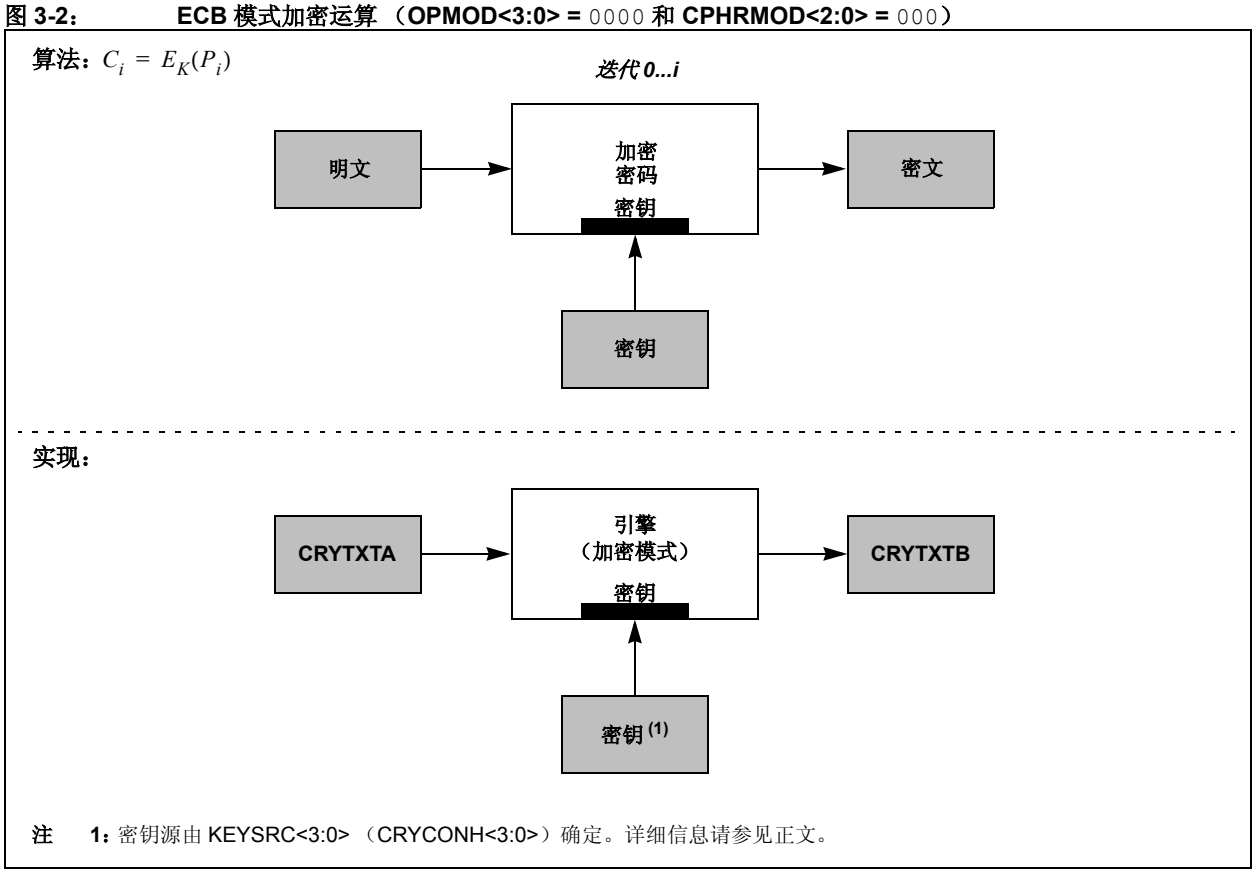
（使用密钥 K 解密密文块 i 从而产生明文块 i）。

在 ECB 模式下单纯使用块密码来加密数据存在许多问题。首先，由于每个明文块每次都会加密为相同的密文块，这可能导致在不知道明文块的内容的情况下，也可将密文块与某个事件关联起来。当有人想要触发该事件时，他们只需简单地重新发送对应的密文块；这个过程称为**重放攻击**。

此外，大多数 ECB 模式下的块加密算法不会对重复数据进行扰码处理，使得可从密文块部分逆推出明文块。使用 ECB 模式加密任何一种简单的图形图像都将导致加密图形图像仍然可识别。

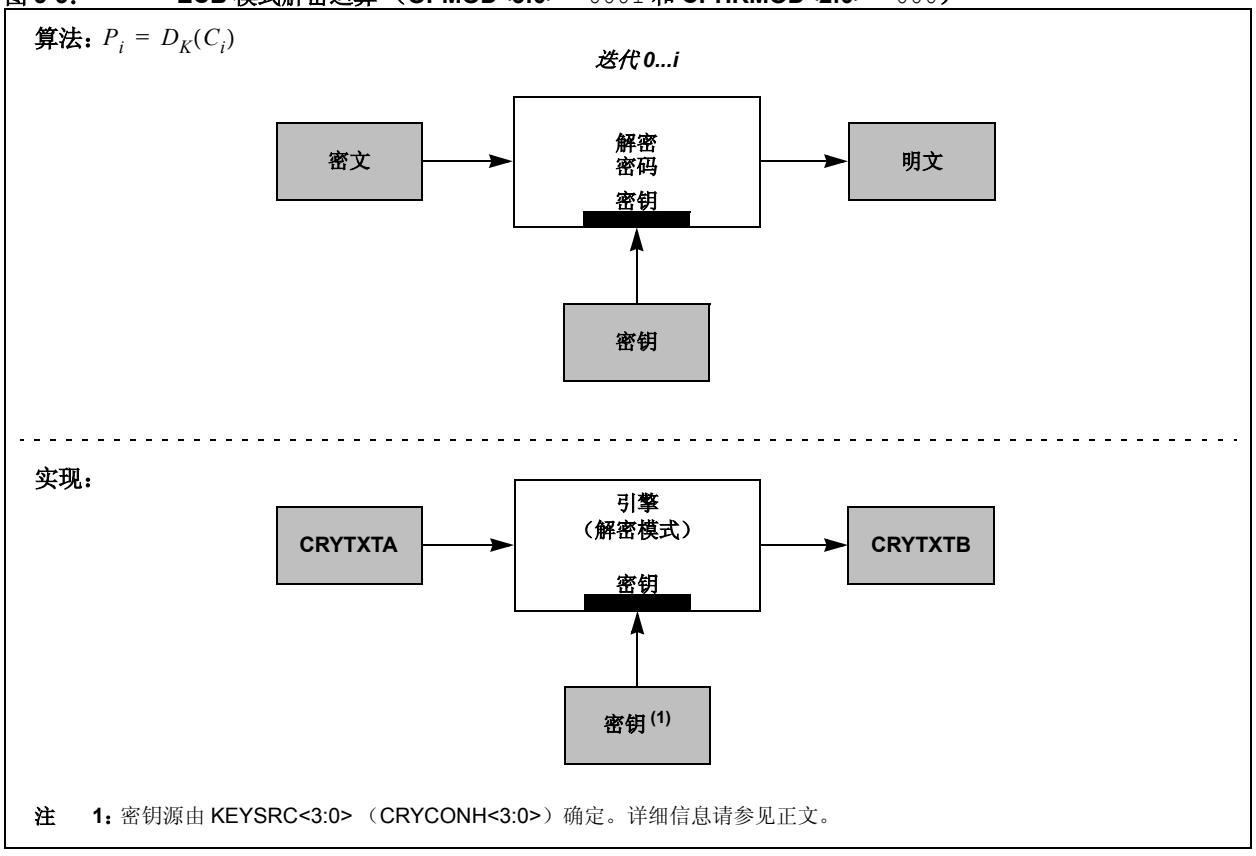
因此，块密码通常用在下面章节所描述的各种块密码模式中。

图 3-2：　ECB 模式加密运算 （OPMOD<3:0> = 0000 和 CPHRMOD<2:0> = 000）



注　1：密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

图 3-3：　ECB 模式解密运算 （OPMOD<3:0> = 0001 和 CPHRMOD<2:0> = 000）



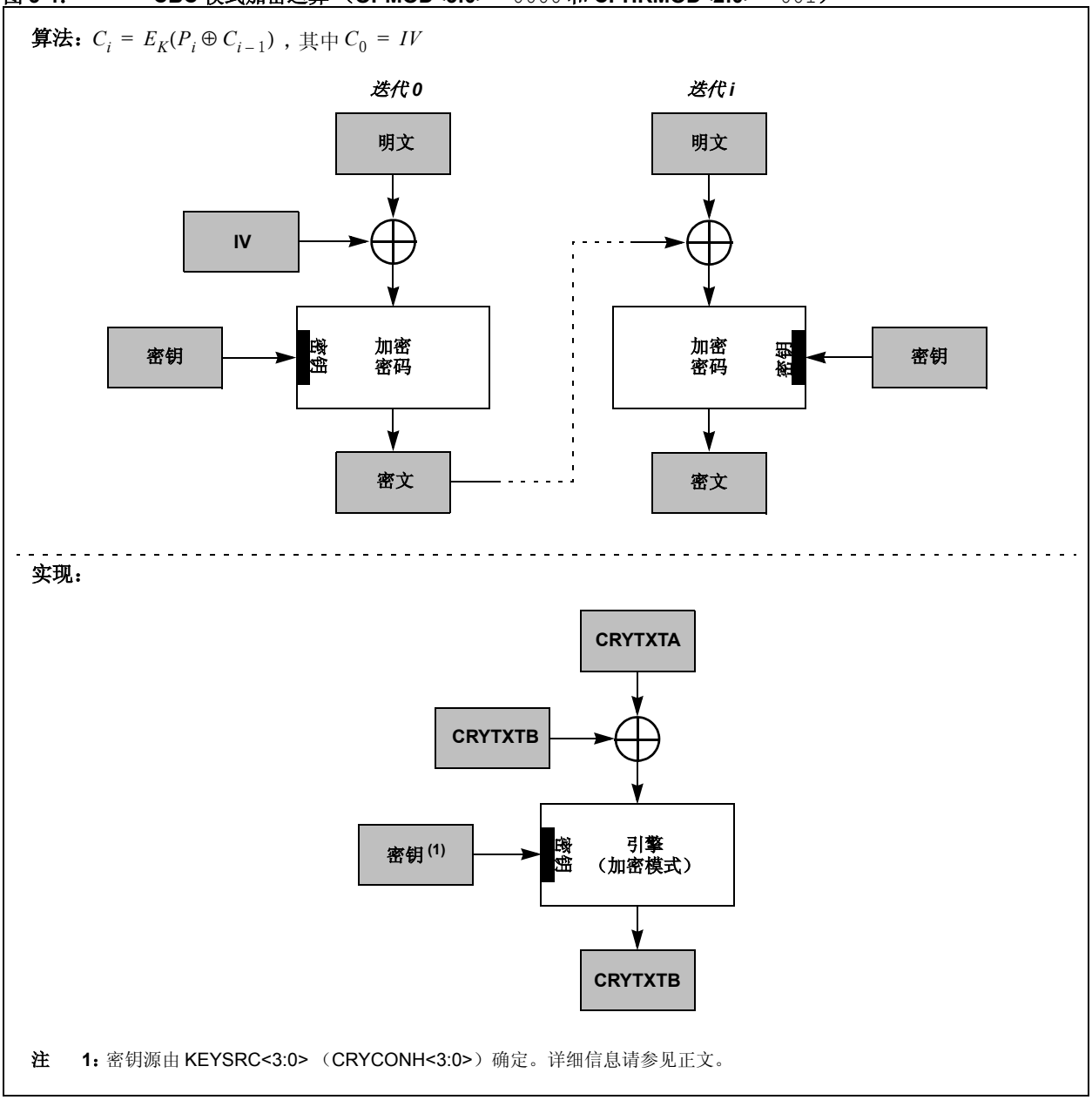
注　1：密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

3.2.2 密码分组链接模式

在**密码分组链接（Cipher Block Chaining， CBC）**模式中，如图 3-4 和图 3-5 所示，每个明文块在加密前，先与前面的块加密运算的结果进行异或运算。使用这种方式时，所有明文块均依赖于前面的块，这使得不进行检测就删除、增加或修改单个块变得更加困难。此外，使用明文块与一个**初始值 （Initial Value， IV）**的异或结果来对第一个块进行加密。可针对每条消息改变这个初始值，从而更好抵抗重放攻击。

该运算模式的主要缺点是加密过程必须按顺序进行，因此无法利用计算机的并行处理能力来加快运算。然而，对于解密，可根据 2 个密文块解密出每个明文块，因此，其更适合并行处理。

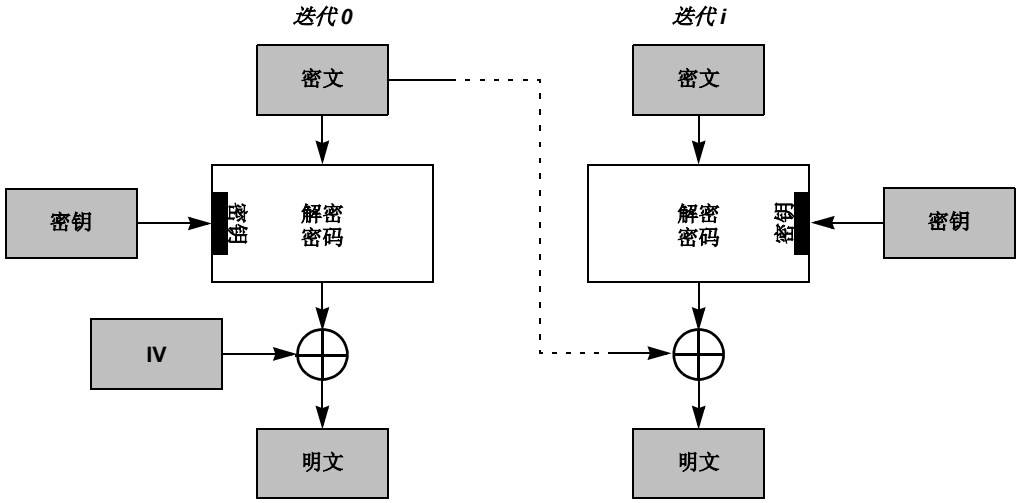
图 3-4： CBC 模式加密运算 （OPMOD<3:0> = 0000 和 CPHRMOD<2:0> = 001）



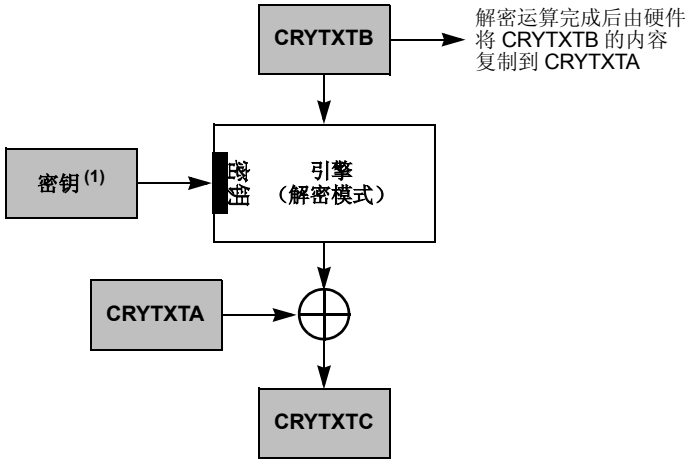
注 1：密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

图 3-5：　　CBC 模式解密运算 （OPMOD<3:0> = 0001 和 CPHRMOD<2:0> = 001）

算法： $P_i = D_K(C_i) \oplus C_{i-1}$，其中 $C_0 = IV$



实现：



注　1： 密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

3.2.3　　加密反馈模式

在**加密反馈（Cipher Feedback， CFB）**模式（图 3-6 和图 3-7）中，将明文块与前面的密文块的加密版本进行异或运算来产生每个密文块。与 CBC 模式一样，提供一个 IV 作为初始种子，启动消息加密过程。

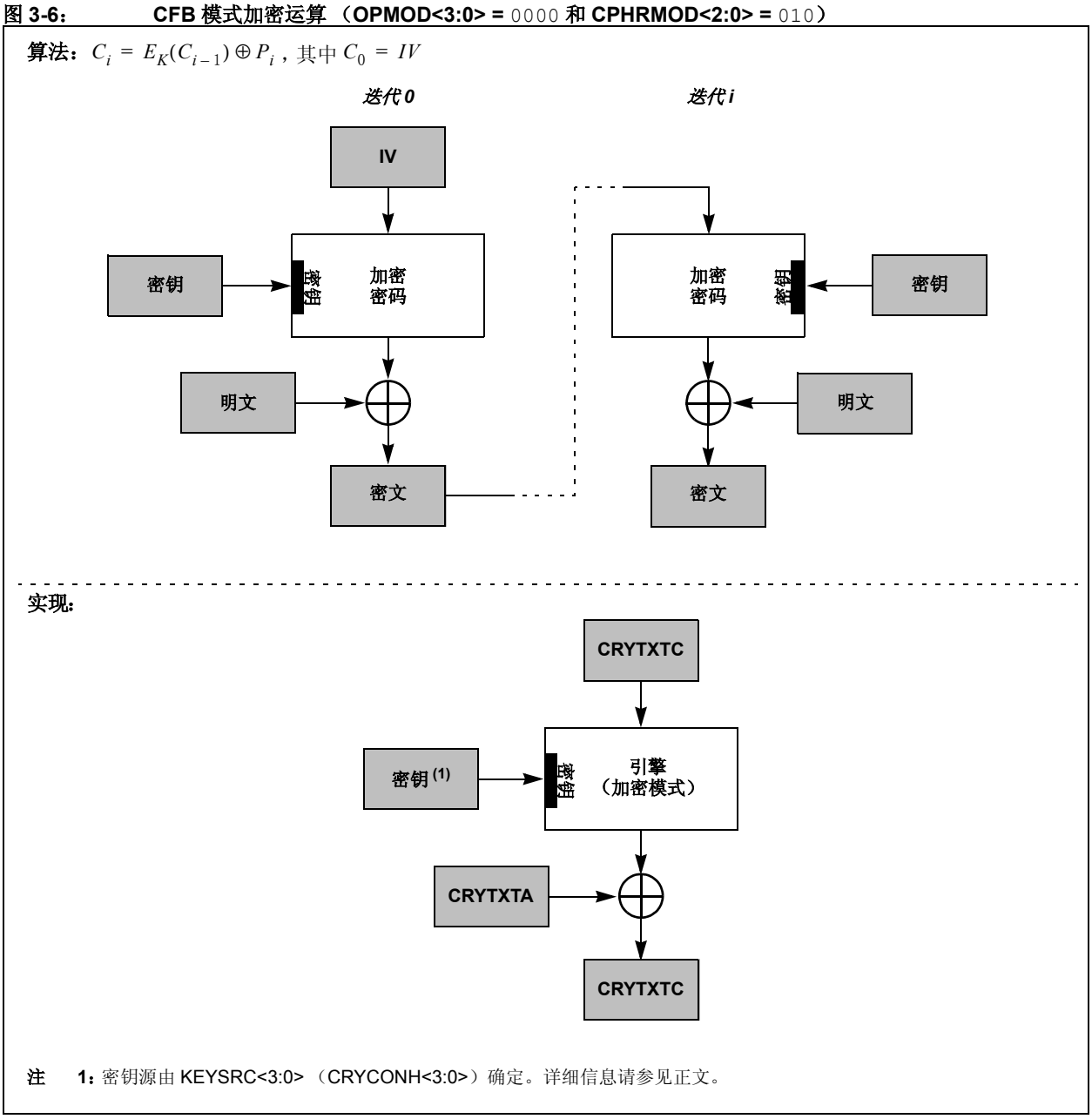
和 CBC 模式一样， CFB 模式无法并行进行加密，但可以并行解密。然而，相对于 CBC 模式，CFB 模式有 2 个明显的优势：低延迟和资源复用。

与任何块加密模式的加密运算一样，大部分计算时序花在实际的加密密码而不是异或运算上。在 CFB 模式中，只有前面的密文使用加密密码进行运算。这意味着一旦前面的明文已处理好，就可执行加密运算，但该操作必须先于当前明文可用。在这种方式下，当前明文可用与计算出相应密文之间的时间间隔（即加密运算的延时）将最小化。这对任何需要加密的实时应用程序产生重大影响。类似的论点也适用于解密过程。

CFB 模式也可以使用不同的反馈长度来运行。 AES 最常见的 CFB 模式是 CFB1、 CFB8 和 CFB128。对于 DES/TDES，最常见的模式为 CFB1、 CFB8 和 CFB64。该模块针对 AES 实现 CFB128，针对 DES/TDES 实现 CFB64。

CFB 解密运算使用相同的加密密码，而非不同的解密密码。当用软件实现时，具有代码重用性和代码大小方面的优势。当用硬件实现时，具有减小硬件大小的优势，前提是使用的是半双工操作（即无法同时进行加密和解密）。

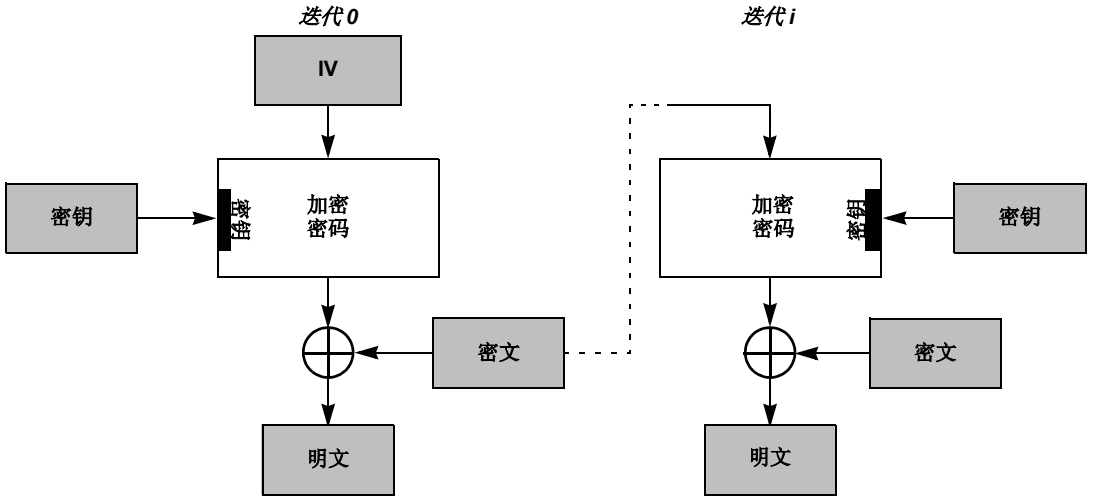
图 3-6： CFB 模式加密运算 （OPMOD<3:0> = 0000 和 CPHRMOD<2:0> = 010）



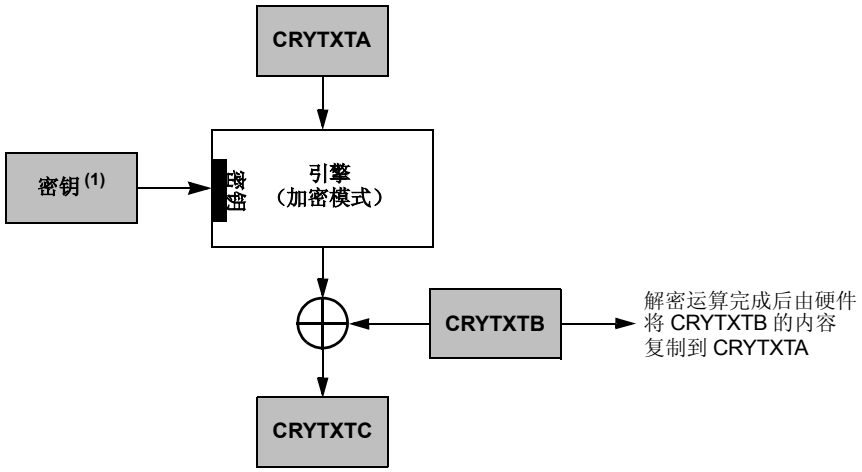
注 1： 密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

图 3-7：　CFB 模式解密运算 （OPMOD<3:0> = 0001 和 CPHRMOD<2:0> = 010）

算法： $P_i = E_K(C_{i-1}) \oplus C_i$ ，其中 $C_0 = IV$



实现：



注　1： 密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

3.2.4　输出反馈模式

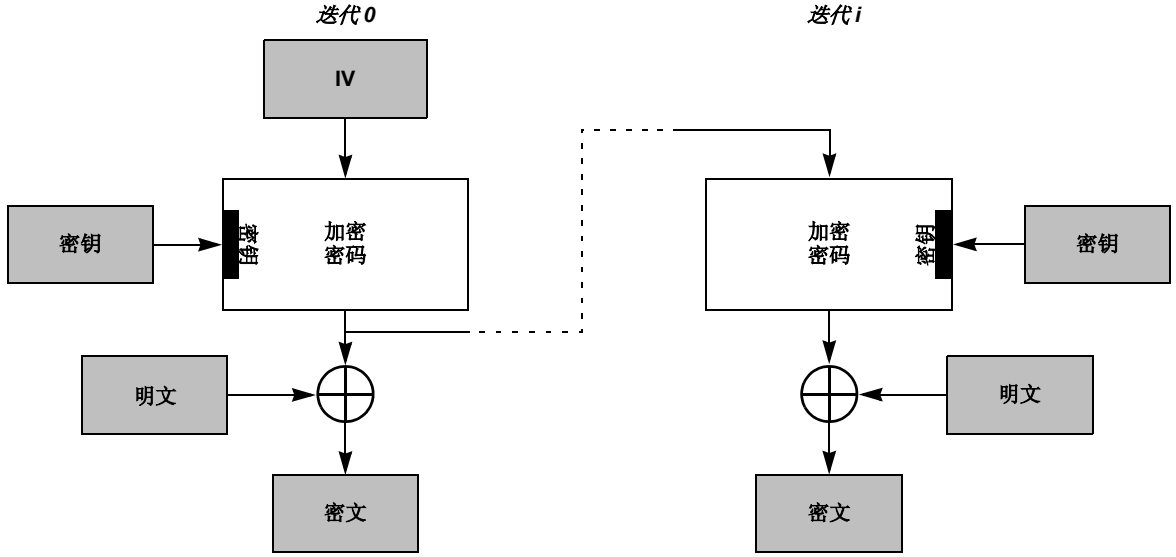
在**输出反馈（Output Feedback，OFB）**模式中，如图 3-8 和图 3-9 所示，将明文块与前面的加密密码输出的加密版本 （在图示的公式中称为 O_i）进行异或运算来生成每个密文块。与 CBC 和 CFB 模式一样，提供一个 IV 作为初始种子，启动消息加密过程。

与 CBC 和 CFB 模式一样， OFB 模式无法 100% 并行进行加密，但可以并行解密。然而， OFB 模式可比 CFB 进行更多地并行处理，原因是每次加密运算只需要前面加密运算的结果，而不需要使用明文或密文。因此，给定 IV 和要处理的明文块的数量后，甚至可在第一个明文块可用之前，就执行完所有加密运算。一旦这个步骤完成了，就可以并行计算密文块。

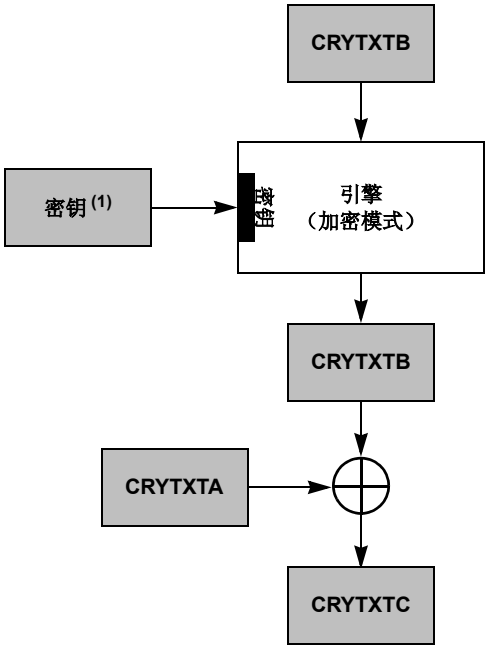
与 CFB 模式一样，OFB 模式可提供低延迟并复用软件或硬件资源。然而，OFB 模式的优点是加密和解密运算是完全相同的，从而进一步提高重用性。

图 3-8：　　OFB 模式加密运算 （OPMOD<3:0> = 0000 和 CPHRMOD<2:0> = 011）

算法： $C_i = P_i \oplus O_i$，其中 $O_0 = IV$ 和 $O_i = E_K(O_{i-1})$



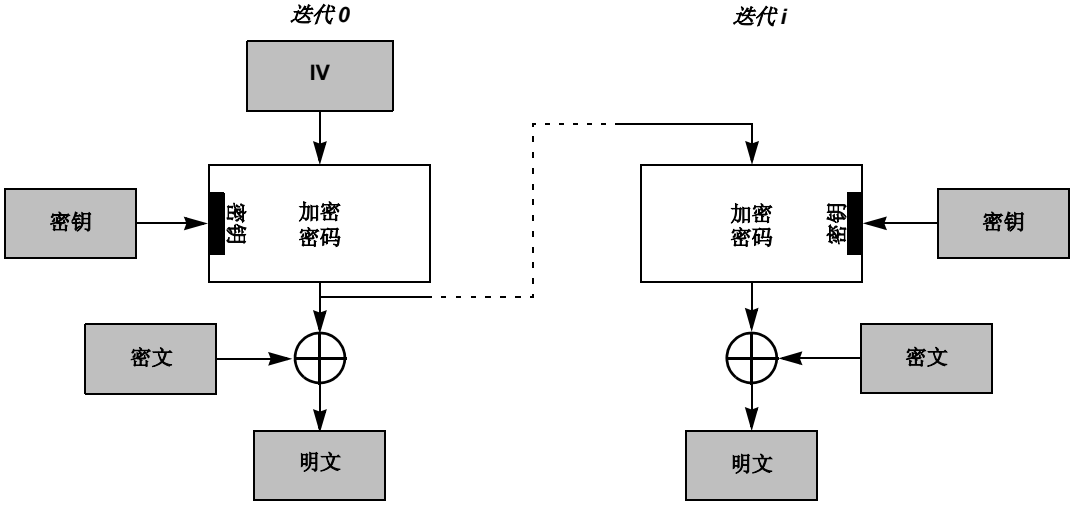
实现：



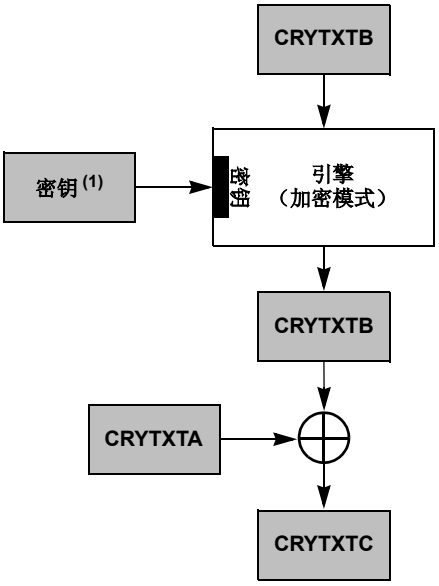
注　1： 密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

图 3-9：　　OFB 模式解密运算 （OPMOD<3:0> = 0001 和 CPHRMOD<2:0> = 011）

算法： $P_i = C_i \oplus O_i$，其中 $O_0 = IV$ 和 $O_i = E_K(O_{i-1})$



实现：



注　1： 密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

3.2.5 计数器模式

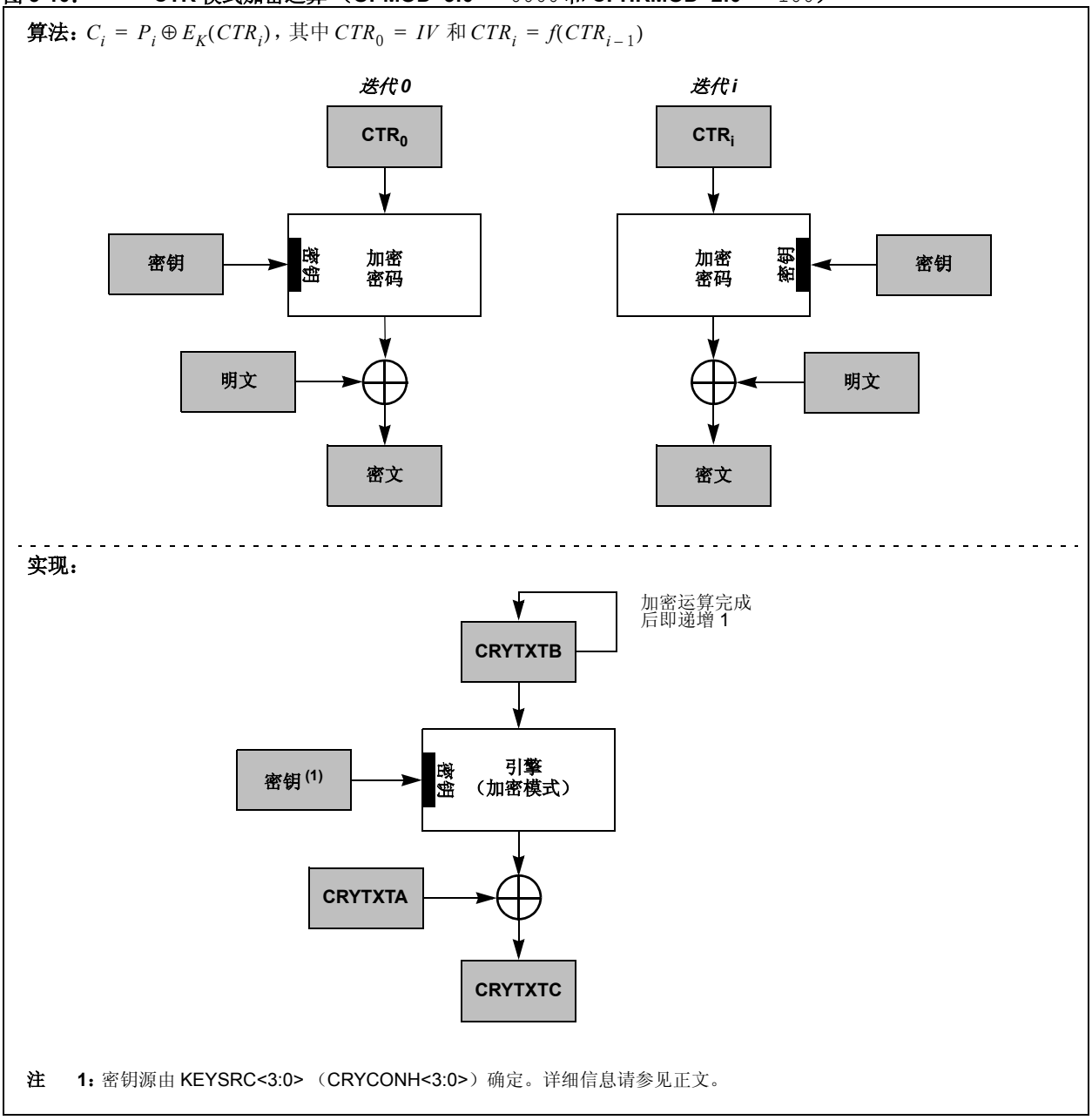
在**计数器（Counter， CTR）**模式中（图 3-10 和图 3-11），将明文块与计数器输入的加密版本做异或运算来生成每个密文块。计数器输入的初始值（IV）充当消息的 IV，并且和其他模式一样，可针对每条消息发生改变。

计数器输入通常由随机数和计数器两部分组成。计数器和随机数都是计数器输入的一部分，不同的是计数器针对一个会话中的每个块而改变，而随机数只随着会话的改变而改变。

尽管使用了“计数器”这个术语，但并不要求使用真正的计数器。任何易于计算的并且实际上不重复的（至少在很长一段时间内）函数都可使用。

与 CBC、 CFB 和 OFB 模式不同， CTR 模式下的加密和解密可以 100% 并行进行。

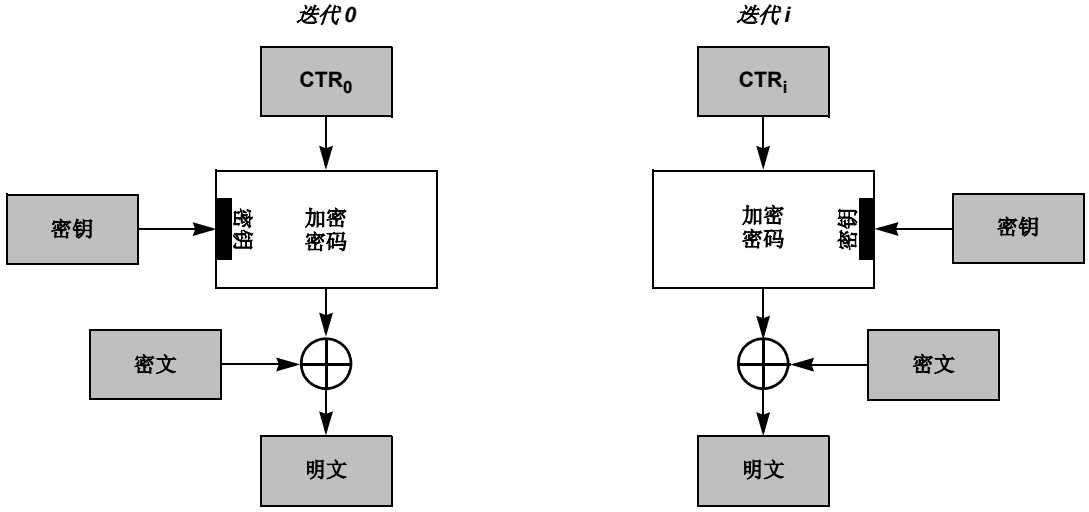
图 3-10： CTR 模式加密运算（OPMOD<3:0> = 0000 和 CPHRMOD<2:0> = 100）



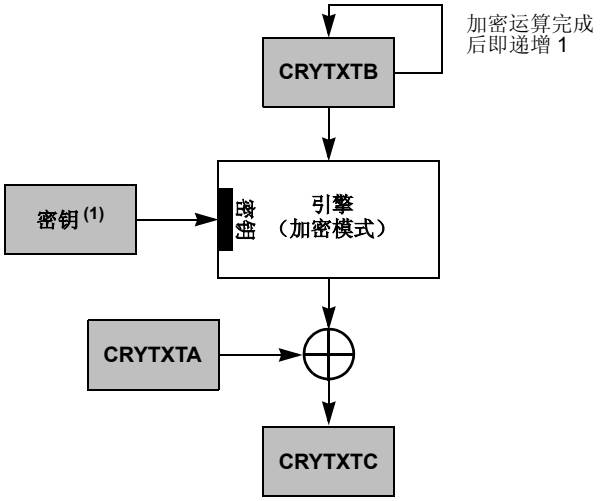
注 1：密钥源由 KEYSRC<3:0>（CRYCONH<3:0>）确定。详细信息请参见正文。

图 3-11： CTR 模式解密运算 （OPMOD<3:0> = 0001 和 CPHRMOD<2:0> = 100）

算法： $P_i = C_i \oplus E_K(CTR_i)$，其中 $CTR_0 = IV$ 和 $CTR_i = f(CTR_{i-1})$



实现：



注 1： 密钥源由 KEYSRC<3:0> （CRYCONH<3:0>）确定。详细信息请参见正文。

3.3 流密码

正如前面所讨论的那样，流密码使用一个状态（通过一个下一个状态函数生成）和明文来计算相关的密文（见图 3-1）。在此上下文中，该状态称为密钥流，因为它起到加密密钥的作用（即与明文相结合来产生密文）并且其值在不断改变（即一个密钥流）。

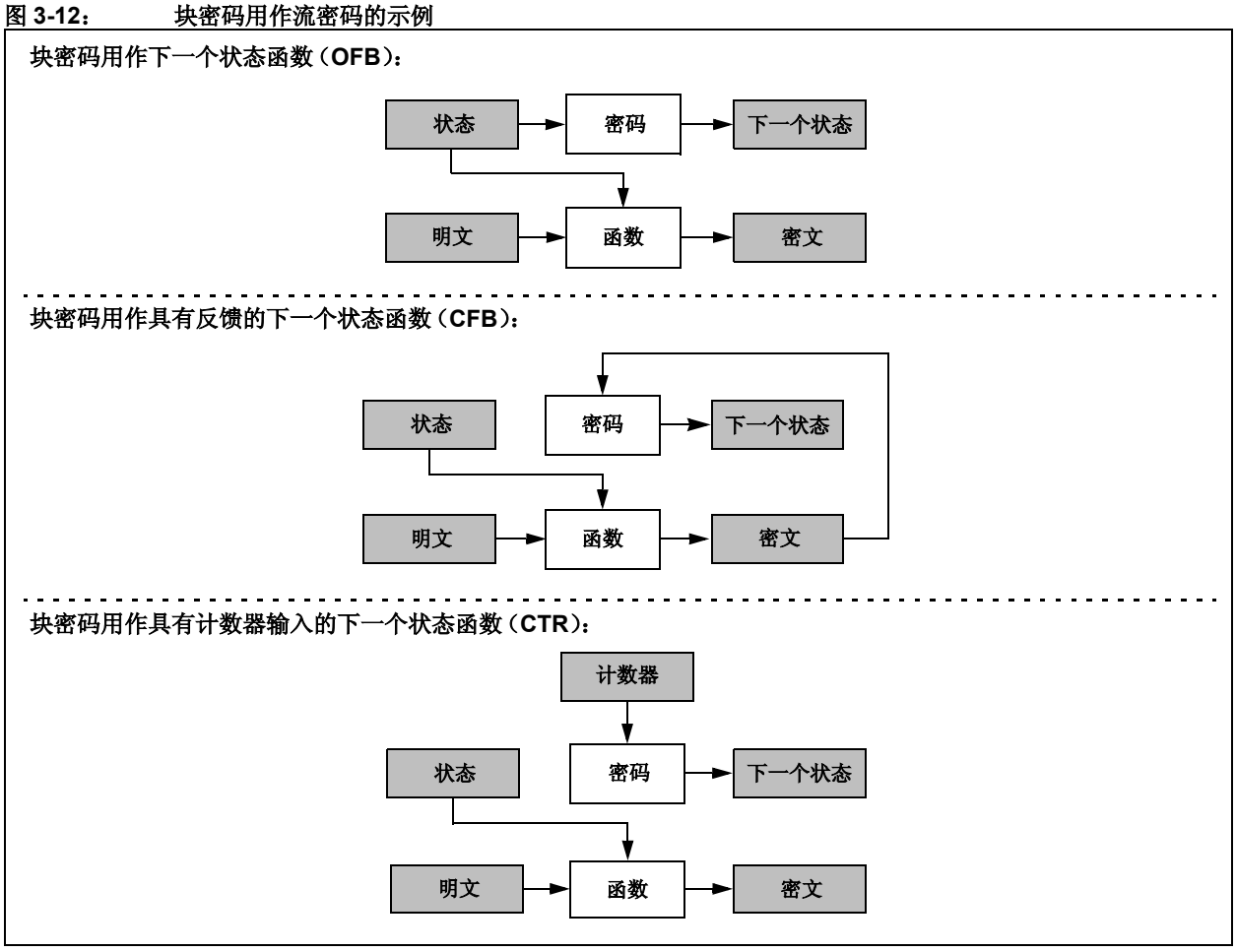
当密钥流独立于明文或密文生成（如图 3-12 所示的 OFB 和 CTR 模式）时，流密码称为同步流密码，因为要进行一次成功的解密运算，加密和解密密码必须具有相同的状态（密钥流）。如果添加或丢失了一条密文消息，接收器的下一个状态函数将计算出一个错误的“下一个状态”，而下一条消息的解密将失败。

然而，如果下一个状态是明文或密文的函数（如图 3-12 所示的 CFB 模式），则流密码变成自同步的流密码，因为可以在接收到合适数量的密文后恢复下一个状态。

3.3.1 使用块密码来创建流密码

如**第 3.1.1 节“块密码与流密码”**中所述，流密码包括一个下一个状态函数和一个将明文转换为密文的函数。要创建一个加密安全的流密码，只需要保证这两个函数中的其中一个加密安全。因此，可以使用现成的块密码函数来创建安全的流密码。事实上，前面讨论的 CFB、OFB 和 CTR 模式就是这样做的。如图 3-12 所示，块密码用作下一个状态函数，而一个简单的异或函数用作明文到密文的转换函数以创建一个流密码，这个流密码的加密安全性与基本块密码一样。

图 3-12： 块密码用作流密码的示例



3.4 密码的完整性保护

虽然到目前为止我们所讨论的密码模式（块密码和流密码）可保护隐私，但却不能保护消息的完整性。有人可在不知道明文消息的情况下修改加密消息，并且其篡改行为不会被检测到。为此，通常会在现代安全通信中为加密提供一定形式的完整性保护。通常使用**消息验证代码（Message Authentication Code， MAC）**来实现该完整性保护；请参见**第 3.4.1 节 “消息验证代码（MAC）”**。

当加密与 MAC 结合来提供保密性和可靠性 （它意味着完整性）时，生成算法称为**认证加密**。

3.4.1 消息验证代码 （MAC）

消息验证代码（MAC）保护消息的可靠性和 （暗含）完整性。 MAC 算法使用长度可变的输入消息和一个密钥，生成一个 **MAC 标记**作为输出。消息内容的任何更改都会导致 MAC 标记的变化，从而确保消息的完整性。由于使用不同密钥的相同消息也会生成不同的 MAC 标记，MAC 也可以提供可靠性保护。

术语 MIC 和 MAC 经常交替使用，而这两者的区别在于：MAC 中使用密钥。密钥的使用可确保消息的可靠性（和完整性），而 MIC 只能确保完整性。实际上，MIC 通常在传输前先对自己进行加密，以抵抗篡改行为。另一方面， MAC 已经使用了密钥，不需要在传输前进行加密。

MAC 算法通常使用加密散列或密码进行构建，如以下章节所述。

3.4.2 基于密码的 MAC

可使用密码轻松创建一个 MAC 标记，方法是结合每个块加密运算的结果来创建一个固定长度的标记。对于一个加密运算依赖于另一个加密运算的那些块密码模式（CBC 和 CFB 模式）， MAC 标记仅是消息中的最后一个密文块。

使用这样的 MAC 时，自然而然地会减少必须通过加密消息来实现的工作量，然后只需使用最后一个密文块作为 MAC 标记。但是可以看出，这样做将允许改变除了最后一块消息之外的每一块，而无法检测到这种改变。因此，使用相同的加密算法进行加密和 MAC 标记生成时，至少要使用不同的密钥。

基于密码的 MAC 通常命名为 `<cipher>-<mode>-MAC`，例如 AES-CBC-MAC。

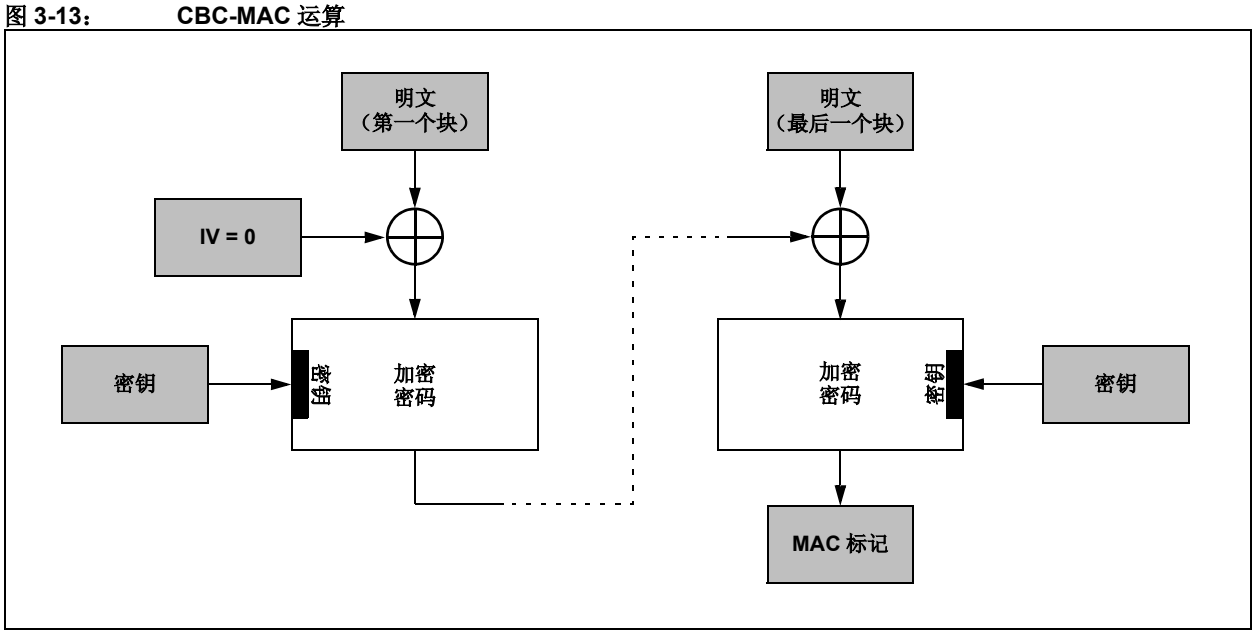
3.4.3 CBC-MAC

图 3-13 显示了如何在 CBC 模式下使用块密码根据消息生成 CBC-MAC 标记。正如前文所述，由于 CBC 模式自然创建了连续块之间的依赖性，则 MAC 标记仅是最后一个密文块。

可以看出，给定一条消息和对应于该消息的 CBC-MAC 标记，简单地将块连接到旧消息，就可以轻松地使用一个有效的 CBC-MAC 标记创建一条新消息。因此，CBC-MAC 只能用于固定长度的消息。对于长度可变的消息，通常使用其他基于密码的 MAC （例如 CMAC）。

请注意，创建 CBC-MAC 标记与执行 CBC 加密运算过程相同，除了初始值 （IV）为 0 和所有中间密文块被丢弃外。

图 3-13： CBC-MAC 运算



3.5 随机数生成

随机数在加密应用中非常有用，可用于很多目的，例如生成临时的加密密钥 （见下文）。可采用 NIST SP 800-90 之类的方法来实现伪随机数发生器 （Pseudorandom Number Generator，PRNG）。 PRNG 会生成一个数字，该数字是确定的，但在给定数量的生成数（称为*重新设置种子时间间隔*）中不重复。该方法与真随机数发生器（True Random Number Generator，TRNG）相反， TRNG 会生成一个不确定的数字，因此可能是重复值。

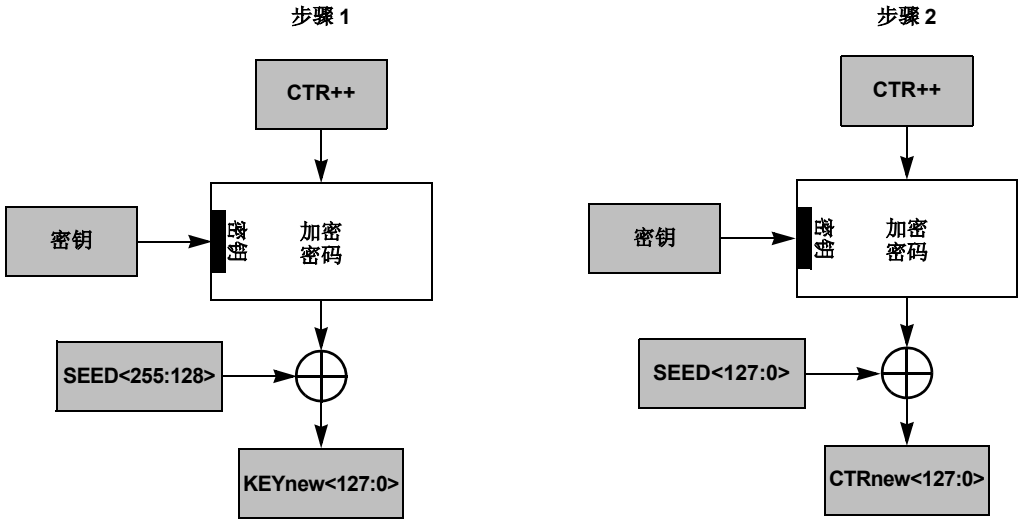
图 3-14 和图 3-15 所示为 SP 800-90 的处理流程。请注意，重新设置种子过程实际上是生成初始 PRN 的起点。典型的 PRNG 实现需要一个真随机数来为 PRNG 设置种子。一旦一系列 PRNG 数重复，必须执行重新设置种子操作以启动新的 PRNG 数序列。

3.6 密钥生成和封装

对于想要提供更高保护级别以防止密钥被获取的应用程序，可使用会话密钥来代替固定密钥。会话密钥可看作是使用一段固定时间后就丢弃的临时密钥。这提供了有限的时间窗口，在这段时间内系统必须被破解才能获取有用的密钥。在会话密钥到期后，通过系统中的主模块或从模块生成一个新的密钥 （密钥生成），并在传输前对其进行加密 （密钥封装）。

图 3-14： NIST SP 800-90 PRNG 重新设置种子操作 （128 位 PRNG）

算法：

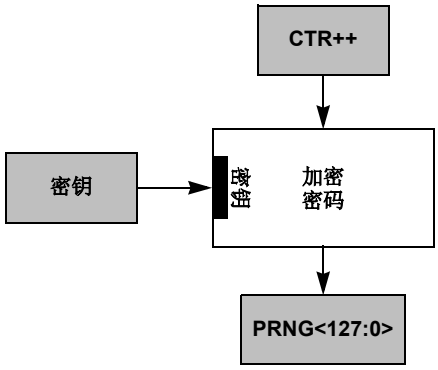


注： 第一次执行重新设置种子操作前，软件必须将 Key 和 CTR 均初始化为 0。每次执行重新设置种子操作前，软件必须重新加载正确的 Key 和 CTR 值 （由上一次重新设置种子操作生成）以进行生成操作。

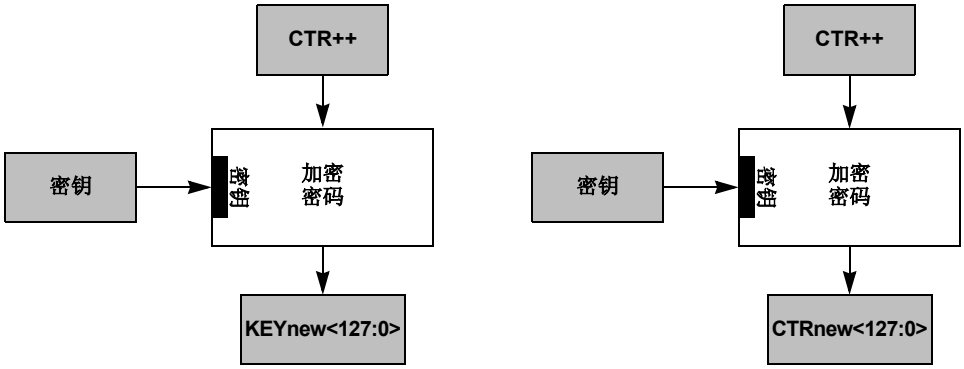
图 3-15： NIST SP 800-90 生成 PRNG （CTR DRBG）操作 （128 位 PRNG）

算法：

步骤 1. 密钥生成



步骤 2. 密钥更新与重新设置种子



注： 第一次执行生成操作前，软件必须执行一次重新设置种子操作。当设置种子时间间隔到期 （即 CRYTXTB 的计数器部分计满返回）后，必须执行后续的重新设置种子操作。

4.0 模块操作

4.1 使能引擎

通过将 CRYON 位置 1 使能加密引擎。清零该位将同时禁止 DES 和 AES 引擎，并导致以下寄存器位保持复位状态：

- CRYGO （CRYCONL<8>）
- TXTABSY （CRYSTAT<6>）
- CRYWR （CRYOTP<0>）

CRYON = 0 时，可读取和写入所有其他寄存器位和寄存器。

4.2 模块内的字节顺序

4.2.1 密钥和数据的字节顺序

模块内的密钥和数据按照字节从低位到高位的顺序存储，即，FIPS 规范中测试用例数据中的最左边字节，是密钥 / 数据存储中的最低有效字节（Least Significant Byte， LSB）。例 4-1 显示了如何存储将 CRYTXTC 用作密文输出的假想加密的数据。

例 4-1： 密钥和数据的字节顺序

```
DATA:
COUNT = 0
KEY = 80000000000000000000000000000000
PLAINTEXT = 00000000000000000000000000000000
CIPHERTEXT = 0edd33d3c621e546455bd8ba1418bec8
LS BYTES OF STORED DATA:
CRYKEY<7:0> = 80
CRYTXTC<7:0> = 0E
```

4.2.2 计数器 （CRYTXTB）字节顺序

按照 NIST 指定的顺序 （即字节按照从小到大顺序，位按照从大到小顺序），处理 CRYTXTB （CPHRMOD<2:0> = 100）中的计数器字节顺序。从标准数据角度来说，可能期望多字节寄存器以统一的位从大到小方式进行，逐位递增，以从右到左的顺序遍历所有字节。就 NIST 的定义而言，最右边的字节以正常的位从大到小方式在字节内递增；当它计满返回时，左侧的下一个字节从其最低有效位 （Least Significant bit， LSb）开始递增，以此类推。

模式如例 4-2 所示。

例 4-2： 递增 CRYTXTB 以显示计数器的字节顺序

CRYTXTB	127 120	119 112	...	15 8	7 0
Operation #1	0000_0000	0000_0000	...	0000_0000	0000_0000
Operation #2	0000_0001	0000_0000	...	0000_0000	0000_0000
Operation #3	0000_0010	0000_0000	...	0000_0000	0000_0000
:					
Operation #256	1111_1110	0000_0000	...	0000_0000	0000_0000
Operation #256	1111_1111	0000_0000	...	0000_0000	0000_0000
Operation #257	0000_0000	0000_0001	...	0000_0000	0000_0000
Operation #258	0000_0000	0000_0010	...	0000_0000	0000_0000
:					

4.3 密钥管理

可从多个源中选出一个源作为加密或解密运算中使用的密钥。表 4-1 和表 4-2 中汇总了不同的源，并且下文对其进行了讨论。

4.3.1 存储的密钥 （安全 OTP 阵列）

加密引擎包括一个 512 字节的一次性可编程 （One-Time-Programmable， OTP）存储器阵列，用于存储加密密钥。为了确保密钥安全性，在运行时或器件编程期间，不能通过任何方式从加密引擎外进行访问来读取该存储空间的内容。 OTP 阵列因此被称为是 “编程安全”的。

安全 OTP 阵列的操作由一个单独的 32 位 OTP 阵列控制，称作配置页 0 （CFGPAGE）。实际上，可将 CFGPAGE 看作一个 32 位配置寄存器。寄存器 2-5 给出了其控制位的位置和功能。

注： 不能将CFGPAGE寄存器与标准器件配置寄存器或PIC24器件的闪存配置字相混淆。

器件复位时， CFGPAGE 以及安全 OTP 阵列中的其他存储单元不会恢复为默认设置。 1 个位被编程 （= 1）后，则不能以任何方式将其清零。在执行 OTP 编程操作期间，保持各个位未被编程 （= 0），则以后可以进行编程。

用户可将最多 7 个 DES 密钥或 4 个 AES 密钥预编程到安全密钥存储区。可用的密钥存储区和存储单元由加密引擎的工作模式和密钥源模式确定，详情见表 4-1 和表 4-2。

当 SKEYEN （CFGPAGE<19>）置 1 时，存储的 Key #1 专门用作密钥加密密钥 （KEK），加密和解密会话密钥。它不可用作通用加密 / 解密密钥。

4.3.1.1 编程 OTP 页

OTP 一次编程 64 位，并且对 KEYPG<3:0> 指定的页编程。编程的数据取自 CRYTXTC<63:0>。在对 OTP 配置页进行编程后，CRYREAD 位应置 1，并且用户应等待该位清零之后，再使用 OTP 作为源执行任何其他操作。要启动 OTP 写操作，必须在 CRYWR 位可置 1 前执行一次编程解锁序列。将 55h 和 AAh 连续写入 NVMKEY 寄存器即完成。然后可在解锁序列之后的几个周期内将 CRYWR 位置 1。例 4-3 显示了如何用汇编语言完成该操作。在使用 MPLAB® XC16 编译器时，有一个内置的函数 `__builtin_write_CRYOTP()` 可以实现这个功能。例 4-4 给出了将一个密钥写入 OTP 的完整序列。

例 4-3： 用汇编语言解锁 CRYWR 位

```
mov   #0x55, W0
mov   W0, _NVMKEY
mov   #0xAA, W0
mov   W0, _NVMKEY
nop
bset  CRYOTP, #0
```

例 4-4：　　将一个 128 位 AES 密钥编程到 OTP

```
BYTE key[] = {
    0x10, 0xa5, 0x88, 0x69,
    0xd7, 0x4b, 0xe5, 0xa3,
    0x74, 0xcf, 0x86, 0x7c,
    0xfb, 0x47, 0x38, 0x59
};

unsigned char otgconfig[] = {
    0b00000000,
    0b00000000,
    0b00100000,   /* KEY1TYPE<1:0> = 10, AES-128 mode */
    0b00000000
};

/* Turn module on. */
CRYCONLbits.CRYON = 1;

/* point to page 0 - the configuration page. */
CRYOTPbits.KEYPG = 0b0000;

memcpy((void*)&CRYTXTC0, otgconfig, sizeof(otgconfig));

/* If PGMFAIL is set, there is an error in the configuration. */
if(CRYSTATbits.PGMFAIL == 1) { return FAIL_PMGFAIL; }

__builtin_write_CRYOTP();
while(CRYOTPbits.CRYWR == 1){}

/* If PGMFAIL is set, there is an error in the programming sequence. */
if(CRYSTATbits.PGMFAIL == 1) { return FAIL_PMGFAIL; }

CRYOTPbits.CRYREAD = 1;
while(CRYOTPbits.CRYREAD == 1){}

/* point to page 1 - We are going to store this key in the first AES128 key
slot. */
CRYOTPbits.KEYPG = 0b0001;

/* copy the lower 8 bytes of data into CRYTXTC to get programmed. */
memcpy((void*)&CRYTXTC0, &key[0], 8);

/* If PGMFAIL is set, there is an error in the configuration. */
if(CRYSTATbits.PGMFAIL == 1) { return FAIL_PMGFAIL; }

/* Start the write operation and wait for it to complete. */
__builtin_write_CRYOTP();
while(CRYOTPbits.CRYWR == 1){}

/* Now we will program the second half of the key.
 * Point to page 2 of the OTP. */
CRYOTPbits.KEYPG = 0b0010;

/* Copy the upper 8 bytes of the key into the CRYTXTC register.*/
memcpy((void*)&CRYTXTC0, &key[8], 8);

/* If PGMFAIL is set, there is an error in the configuration. */
if(CRYSTATbits.PGMFAIL == 1) { return FAIL_PMGFAIL; }

/* Start the write operation and wait for it to complete. */
__builtin_write_CRYOTP();
while(CRYOTPbits.CRYWR == 1){}
```

4.3.1.2 使用 OTP 密钥

要使用存储在 OTP 中的密钥，烧写到 OTP 配置页上的密钥类型必须与 KEYMOD<1:0> 所选择的加密 / 解密模式匹配。如果模式不匹配，将发生配置错误，并且无法执行请求的操作（例 4-5）。

例 4-5： 使用 OTP 密钥进行 128 位 AES 加密

```
/* Example from NIST KAT tests ECBKeySbox128.rsp */

BYTE key[] = {
    0x10, 0xa5, 0x88, 0x69,
    0xd7, 0x4b, 0xe5, 0xa3,
    0x74, 0xcf, 0x86, 0x7c,
    0xfb, 0x47, 0x38, 0x59
};

BYTE plaintext[] = {
    0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00
};

BYTE expected_results[] = {
    0x6d, 0x25, 0x1e, 0x69,
    0x44, 0xb0, 0x51, 0xe0,
    0x4e, 0xaa, 0x6f, 0xb4,
    0xdb, 0xf7, 0x84, 0x65
};

//Turn module on.
CRYCONLbits.CRYON = 1;

//Select the key source (Key #1)
CRYCONHbits.KEYSRC = 0b0001;

//Select the operating mode (Encryption)
CRYCONLbits.OPMOD = 0b0000;

//Select the cipher (AES)
CRYCONLbits.CPHRSEL = 0b1;

//Select the encryption mode (ECB)
CRYCONLbits.CPHRMOD = 0b000;

//Set the key strength to 128-bit
CRYCONHbits.KEYMOD = 0b00;

//Load the plaintext block into CRYTXTA (16 bytes for AES)
memcpy((void*)&CRYTXTA0, plaintext, 16);

//Start the encryption
CRYCONLbits.CRYGO = 0b1;
while(CRYCONLbits.CRYGO == 1){}

if(CRYSTATbits.KEYFAIL == 1){ return FAIL_KEYFAIL_SET; }
```

4.3.2 软件密钥

当 SWKYDIS 位（CFGPAGE<0>）为 0 时，用于加密 / 解密操作的密钥直接取自 CRYKEY 寄存器。KEYSRC<3:0> 必须为 0000（如果 SRCLCK 为 0），或者 LKYSRC<7:0> 必须为 00h（如果 SRCLCK 为 1）。

将 SWKYDIS 编程为 1 会禁止软件对 CRYKEY 执行写操作，并永久禁止软件密钥功能。对于那些不使用软件密钥的应用程序，建议这样设置。然而，将 KEYSRC<3:0> 设置为 0000 还是会允许将 CRYKEY 选作密钥源，但只在结合会话密钥使用的情况下。**第 4.4.4 节“会话密钥加密”**中对此进行了描述。

注：	存储软件密钥实际上会绕过器件的安全密钥存储区，可能会将密钥暴露，从而损失保密性。硬件生成的密钥可提供更高级别的密钥安全性。

4.3.3 密钥加密密钥 （KEK）

密钥加密密钥（KEK）用于将所有生成的会话密钥加密。安全起见，这个密钥绝不能用于加密或解密除了会话密钥以外的任何东西。为此，使用 SKEYEN 位来使能该功能。

当 SKEYEN 被编程后， KEK 只能用于在会话密钥加密或加载操作期间加密或解密会话密钥。在 KEYSRC<3:0> = 0001 时尝试加密或解密会导致 KEYFAIL 状态位置 1。在该情况下，直到 KEYSRC<3:0> 位更改为有效值之后，才会允许进行加密或解密操作。

表 4-1： DES/3DES 密钥源选择

操作模式	KEYMOD<1:0>	KEYSRC<3:0>	会话密钥源 （SESSKEY）		OTP 阵列地址
			0	1	
64 位 DES	00	0000(1)	CRYKEY<63:0>		—
		0001	DES Key #1	密钥配置错误 (2)	<63:0>
		0010	DES Key #2		<127:64>
		0011	DES Key #3		<191:128>
		0100	DES Key #4		<255:192>
		0101	DES Key #5		<319:256>
		0110	DES Key #6		<383:320>
		0111	DES Key #7		<447:384>
		1111	保留 (2)		—
		所有其他值	密钥配置错误 (2)		—
64 位 2 密钥 3DES （标准 2 密钥 E-D-E/D-E-D）	01	0000(1)	CRYKEY<63:0> （第 1 个 / 第 3 个） CRYKEY<127:64> （第 2 个）		—
		0001	DES Key #1 （第 1 个 / 第 3 个） DES Key #2 （第 2 个）	密钥配置错误 (2)	<63:0> <127:64>
		0010	DES Key #3 （第 1 个 / 第 3 个） DES Key #4 （第 2 个）		<191:128> <255:192>
		0011	DES Key #5 （第 1 个 / 第 3 个） DES Key #6 （第 2 个）		<319:256> <383:320>
		0100	DES Key #7 （第 1 个 / 第 3 个） DES Key #8 （第 2 个）		<447:384> <511:448>
		1111	保留 (2)		—
		所有其他值	密钥配置错误 (2)		—
（保留）	10	xxxx	密钥配置错误 (2)		—
64 位 3 密钥 3DES	11	0000(1)	CRYKEY<63:0> （第 1 次迭代） CRYKEY<127:64> （第 2 次迭代） CRYKEY<191:128> （第 3 次迭代）		—
		0001	DES Key #1 （第 1 个） DES Key #2 （第 2 个） DES Key #3 （第 3 个）	密钥配置错误 (2)	<63:0> <127:64> <191:128>
		0010	DES Key #4 （第 1 个） DES Key #5 （第 2 个） DES Key #6 （第 3 个）		<255:192> <319:256> <383:320>
		1111	保留 (2)		—
		所有其他值	密钥配置错误 (2)		—

注 1： 当 SWKYDIS 也置 1 时，该配置视为密钥配置错误 （KEYFAIL 位置 1）。

2： 选择这些配置时， KEYFAIL 位 （CRYSTAT<1>）置 1，并且保持置 1 直至选择了有效配置。

图 4-1： DES 密钥 OTP 页分配 （SKEYEN = 0）

KEYMOD<1:0> = 00/01/11：

	63 … 32	31 … 0
Page 0		CFGPAGE
Page 1	DES Key #1	
Page 2	DES Key #2	
Page 3	DES Key #3	
Page 4	DES Key #4	
Page 5	DES Key #5	
Page 6	DES Key #6	
Page 7	DES Key #7	
Page 8	DES Key #8	

图 4-2： DES 密钥存储 （SKEYEN = 1）

KEYMOD<1:0> = 00：

	63 … 32	31 … 0
Page 0		CFGPAGE
Page 1	密钥加密 Key #1	
Page 2	DES Key #2	
Page 3	DES Key #3	
Page 4	DES Key #4	
Page 5	DES Key #5	
Page 6	DES Key #6	
Page 7	DES Key #7	
Page 8	DES Key #8	

KEYMOD<1:0> = 01：

	63 … 32	31 … 0
Page 0		CFGPAGE
Page 1	密钥加密 Key #1	
Page 2	密钥加密 Key #2	
Page 3	DES Key #3	
Page 4	DES Key #4	
Page 5	DES Key #5	
Page 6	DES Key #6	
Page 7	DES Key #7	
Page 8	DES Key #8	

KEYMOD<1:0> = 11：

	63 … 32	31 … 0
Page 0		CFGPAGE
Page 1	密钥加密 Key #1	
Page 2	密钥加密 Key #2	
Page 3	密钥加密 Key #3	
Page 4	DES Key #4	
Page 5	DES Key #5	
Page 6	DES Key #6	
Page 7	DES Key #7	
Page 8	DES Key #8	

表 4-2： AES 密钥模式 / 源选择

操作模式	KEYMOD<1:0>	KEYSRC<3:0>	密钥源		OTP 地址
			SKEYEN = 0	SKEYEN = 1	
128 位 AES	00	0000(1)	CRYKEY<127:0>		—
		0001	AES Key #1	密钥配置错误 (2)	<127:0>
		0010	AES Key #2		<255:128>
		0011	AES Key #3		<383:256>
		0100	AES Key #4		<511:384>
		1111	保留 (2)		—
		所有其他值	密钥配置错误 (2)		—
192 位 AES	01	0000(1)	CRYKEY<191:0>		—
		0001	AES Key #1	密钥配置错误 (2)	<191:0>
		0010	AES Key #2		<383:192>
		1111	保留 (2)		—
		所有其他值	密钥配置错误 (2)		—
256 位 AES	10	0000(1)	CRYKEY<255:0>		—
		0001	AES Key #1	密钥配置错误 (2)	<255:0>
		0010	AES Key #2		<511:256>
		1111	保留 (2)		—
		所有其他值	密钥配置错误 (2)		—
（保留）	11	xxxx	密钥配置错误 (2)		—

注 1： 当 SWKYDIS 也置 1 时，该配置视为密钥配置错误 （KEYFAIL 位置 1）。

2： 选择这些配置时， KEYFAIL 位 （CRYSTAT<1>）置 1，并且保持置 1 直至选择了有效配置。

图 4-3： AES 密钥 OTP 页分配 （SKEYEN = 0）

KEYMOD<1:0> = 00：

	63 31	30 0
Page 0		CFGPAGE
Page 1	AES Key #1 <63:0>	
Page 2	AES Key #1 <127:64>	
Page 3	AES Key #2 <63:0>	
Page 4	AES Key #2 <127:64>	
Page 5	AES Key #3 <63:0>	
Page 6	AES Key #3 <127:64>	
Page 7	AES Key #4 <63:0>	
Page 8	AES Key #4 <127:64>	

KEYMOD<1:0> = 01/10：

	63 31	30 0
Page 0		CFGPAGE
Page 1	AES Key #1 <63:0>	
Page 2	AES Key #1 <127:64>	
Page 3	AES Key #1 <191:128>	
Page 4	AES Key #2 <63:0>	
Page 5	AES Key #2 <127:64>	
Page 6	AES Key #2 <191:128>	
Page 7		
Page 8		

KEYMOD<1:0> = 11：

	63 31	30 0
Page 0		CFGPAGE
Page 1	AES Key #1 <63:0>	
Page 2	AES Key #1 <127:64>	
Page 3	AES Key #1 <191:128>	
Page 4	AES Key #1 <255:192>	
Page 5	AES Key #2 <63:0>	
Page 6	AES Key #2 <127:64>	
Page 7	AES Key #2 <191:128>	
Page 8	AES Key #2 <255:192>	

图 4-4： AES 密钥 OTP 页分配 （SKEYEN = 1）

KEYMOD<1:0> = 00：

	63 … 31	30 … 0
Page 0		CFGPAGE
Page 1	密钥加密 Key #1 <63:0>	
Page 2	密钥加密 Key #1 <127:64>	
Page 3	AES Key #2 <63:0>	
Page 4	AES Key #2 <127:64>	
Page 5	AES Key #3 <63:0>	
Page 6	AES Key #3 <127:64>	
Page 7	AES Key #4 <63:0>	
Page 8	AES Key #4 <127:64>	

KEYMOD<1:0> = 01/10：

	63 … 31	30 … 0
Page 0		CFGPAGE
Page 1	密钥加密 Key #1 <63:0>	
Page 2	密钥加密 Key #1 <127:64>	
Page 3	密钥加密 Key #1 <191:128>	
Page 4	AES Key #2 <63:0>	
Page 5	AES Key #2 <127:64>	
Page 6	AES Key #2 <191:128>	
Page 7		
Page 8		

KEYMOD<1:0> = 11：

	63 … 31	30 … 0
Page 0		CFGPAGE
Page 1	密钥加密 Key #1 <63:0>	
Page 2	密钥加密 Key #1 <127:64>	
Page 3	密钥加密 Key #1 <191:128>	
Page 4	密钥加密 Key #1 <255:192>	
Page 5	AES Key #2 <63:0>	
Page 6	AES Key #2 <127:64>	
Page 7	AES Key #2 <191:128>	
Page 8	AES Key #2 <255:192>	

4.4 选择操作模式

加密引擎支持以下几种操作模式（通过 OPMOD<3:0> 位确定）：

- 块加密
- 块解密
- AES 解密密钥扩展
- 随机数发生
- 会话密钥生成
- 会话密钥加密
- 会话密钥加载

OPMODx 位可在 CRYON 置 1 时发生更改。这些位应仅在加密操作未完成（CRYGO = 0）时发生更改。

选择加密操作和相应的有效密钥配置后，通过将 CRYGO 位置 1 来执行该操作。当操作完成时，该位由硬件自动清零。也可用软件手动清零 CRYGO 位，这可使任何正在进行中的操作立即终止。用软件清零该位还会将 CRYABRT 位（CRYSTAT<5>）置 1。

对于大多数操作，仅当 OTP 操作未在执行且未出现任何其他错误情况时，才可将 CRYGO 置 1。CRYREAD、CRYWR、CRYABRT、ROLLOVR、MODFAIL 和 KEYFAIL 必须全为 0。

同时置 1 CRYWR 和 CRYGO 将不会启动 OTP 编程操作或任何其他操作。模块禁止（CRYON = 0）时将 CRYGO 置 1 也无任何作用。

4.4.1 块加密

在块加密操作中，根据选择的加密模式，将明文加载到三个数据空间寄存器（CRYTXTA、CRYTXTB 或 CRYTXTC）中的其中一个。使用 KEYSRC<3:0> 和 KEYMOD<1:0> 指定的密钥和 CPHRMOD<2:0> 指定的模式对该数据进行加密。生成的密文将根据选择的模式放入三个文本寄存器中的其中一个。（关于每种模式的具体详细信息，请参见**第 3.2 节 “块密码”**。）

执行加密操作期间，CRYKEY 寄存器的内容不会受到干扰。

如果 CTRSIZE<6:0> 设置为 00h（通常为 CTR 模式）以外的其他值，CRYTXTB 寄存器会在加密操作完成*后*递增 1。CRYTXTB 内的计数器大小由 CTRSIZE<6:0> 位的值确定。

例 4-6 提供了用 C 语言编写 EBC 模式下单个块加密的示例。例 4-7 提供了 CBC 模式加密的类似示例。不同加密和密码模式下的块加密是相似的，都会受到每种模式的块和密钥大小的限制。

例 4-6： AES-ECB 模式加密 （NIST 示例）

```
/* The example is from the AES Multiblock Message Test (MMT) sample vectors
 * available on the NIST website
 * (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesmmt.zip)
 * Counts 3 of encrypt section of the CBCMMT128.rsp file
 */
unsigned char plaintext[] = {
0x9a, 0xc1, 0x99, 0x54, 0xce, 0x13, 0x19, 0xb3,
0x54, 0xd3, 0x22, 0x04, 0x60, 0xf7, 0x1c, 0x1e,
0x37, 0x3f, 0x1c, 0xd3, 0x36, 0x24, 0x08, 0x81,
0x16, 0x0c, 0xfd, 0xe4, 0x6e, 0xbf, 0xed, 0x2e,
0x79, 0x1e, 0x8d, 0x5a, 0x1a, 0x13, 0x6e, 0xbd,
0x1d, 0xc4, 0x69, 0xde, 0xc0, 0x0c, 0x41, 0x87,
0x72, 0x2b, 0x84, 0x1c, 0xda, 0xbc, 0xb2, 0x2c,
0x1b, 0xe8, 0xa1, 0x46, 0x57, 0xda, 0x20, 0x0e };

unsigned char key[] = {
0xb7, 0xf3, 0xc9, 0x57, 0x6e, 0x12, 0xdd, 0x0d,
0xb6, 0x3e, 0x8f, 0x8f, 0xac, 0x2b, 0x9a, 0x39 };

unsigned char iv[] = {
0xc8, 0x0f, 0x09, 0x5d, 0x8b, 0xb1, 0xa0, 0x60,
0x69, 0x9f, 0x7c, 0x19, 0x97, 0x4a, 0x1a, 0xa0 };

/* expected results:
 *  0x19, 0xb9, 0x60, 0x97, 0x72, 0xc6, 0x3f, 0x33,
 *  0x86, 0x08, 0xbf, 0x6e, 0xb5, 0x2c, 0xa1, 0x0b,
 *  0xe6, 0x50, 0x97, 0xf8, 0x9c, 0x1e, 0x09, 0x05,
 *  0xc4, 0x24, 0x01, 0xfd, 0x47, 0x79, 0x1a, 0xe2,
 *  0xc5, 0x44, 0x0b, 0x2d, 0x47, 0x31, 0x16, 0xca,
 *  0x78, 0xbd, 0x9f, 0xf2, 0xfb, 0x60, 0x15, 0xcf,
 *  0xd3, 0x16, 0x52, 0x4e, 0xae, 0x7d, 0xcb, 0x95,
 *  0xae, 0x73, 0x8e, 0xbe, 0xae, 0x84, 0xa4, 0x67
 */
unsigned char ciphertext[sizeof(plaintext)] = {0};
unsigned char i;

CRYCONLbits.CRYON = 0b1;                  //Turn module on
CRYCONHbits.KEYSRC = 0b0000;              //Select the key source (CRYKEY)
CRYCONLbits.OPMOD = 0b0000;               //Select the operational mode
                                          //(Encryption)
CRYCONLbits.CPHRSEL = 0b1;                //Select the cipher (AES)
CRYCONLbits.CPHRMOD = 0b001;              //Select encryption mode (CBC)
CRYCONHbits.KEYMOD = 0b00;                //Set key strength to 128-bit
memcpy((void*)&CRYKEY0, key, 16);         //Load the key into CRYKEY
                                          //(128-bit key in this example)
memcpy((void*)&CRYTXTB0, iv, 16);         //Load the initial vector (IV)
                                          //into CRYTXTB
for(i=0; i<sizeof(plaintext); i+=16)      //Loop over the data we have,
                                          //one 16-block at a time (AES)
{
    memcpy((void*)&CRYTXTA0, plaintext + i, 16);   //Load the plaintext block
                                                   //into CRYTXTA
    CRYCONLbits.CRYGO = 0b1;                       //Start the encryption
    while(CRYCONLbits.CRYGO == 0b1){}              //Wait for encryption to
                                                   //complete
    memcpy(ciphertext + i, (void*)&CRYTXTB0, 16);  //Read the results out of
                                                   //CRYTXTB
}
```

例 4-7： AES-CBC 模式加密 （NIST 示例）

```
/* The example is from the AES Multiblock Message Test (MMT) sample vectors
 * available on the NIST website
 * (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesmmt.zip)
 * Counts 3 of encrypt section of the CBCMMT128.rsp file
 */

unsigned char plaintext[] = {
0x9a, 0xc1, 0x99, 0x54, 0xce, 0x13, 0x19, 0xb3,
0x54, 0xd3, 0x22, 0x04, 0x60, 0xf7, 0x1c, 0x1e,
0x37, 0x3f, 0x1c, 0xd3, 0x36, 0x24, 0x08, 0x81,
0x16, 0x0c, 0xfd, 0xe4, 0x6e, 0xbf, 0xed, 0x2e,
0x79, 0x1e, 0x8d, 0x5a, 0x1a, 0x13, 0x6e, 0xbd,
0x1d, 0xc4, 0x69, 0xde, 0xc0, 0x0c, 0x41, 0x87,
0x72, 0x2b, 0x84, 0x1c, 0xda, 0xbc, 0xb2, 0x2c,
0x1b, 0xe8, 0xa1, 0x46, 0x57, 0xda, 0x20, 0x0e };

unsigned char key[] = {
0xb7, 0xf3, 0xc9, 0x57, 0x6e, 0x12, 0xdd, 0x0d,
0xb6, 0x3e, 0x8f, 0x8f, 0xac, 0x2b, 0x9a, 0x39 };

unsigned char iv[] = {
0xc8, 0x0f, 0x09, 0x5d, 0x8b, 0xb1, 0xa0, 0x60,
0x69, 0x9f, 0x7c, 0x19, 0x97, 0x4a, 0x1a, 0xa0 };

unsigned char ciphertext[sizeof(plaintext)];
unsigned char i;

CRYCONbits.ON = 0b1;                        //Turn module on
CRYCONbits.KEYSRC = 0b0000;                 //Select the key source (CRYKEY)
CRYCONbits.OPMOD = 0b0000;                  //Select the operational mode
                                            //(Encryption)
CRYCONbits.CPHRSEL = 0b1;                   //Select the cipher (AES)
CRYCONbits.CPHRMOD = 0b001;                 //Select the encryption mode (CBC)
CRYCONbits.KEYMOD = 0b00;                   //Set the key strength to 128-bit
memcpy(CRYKEY, key, 16);                    //Load the key into CRYKEY
                                            //(128-bit key in this example)
memcpy(CRYTXTB, iv, 16);                    //Load the initial vector (IV)
                                            //into CRYTXTB

for(i=0; i<sizeof(plaintext); i+=16)        //Loop over the data we have,
                                            //one 16-block at a time (AES)
{
    memcpy(CRYTXTA, plaintext + i, 16);     //Load the plaintext block
                                            //into CRYTXTA (16-bytes for AES)
    CRYCONbits.START = 0b1;                 //Start the encryption
    while(CRYCONbits.START == 0xb1){}       //Wait for encryption to complete
    memcpy(ciphertext + i, CRYTXTB, 16);    //Read the results out of CRYTXTB
                                            //(16-bytes for AES)
}
```

4.4.2 块解密

在块解密操作中，根据选择的密码模式，将密文加载到三个文本寄存器（CRYTXTA、CRYTXTB 或 CRYTXTC）中的其中一个。使用 KEYSRC<3:0> 和 KEYMOD<1:0> 指定的密钥和 CPHRMOD<3:0> 位指定的模式来解密该数据。生成的明文将根据选择的模式放入三个文本寄存器中的其中一个。（关于每种模式的具体详细信息，请参见**第 3.2 节 “块密码”**。）

执行解密操作期间， CRYKEY 寄存器的内容不会受到干扰。

如果 CTRSIZE<6:0> 设置为 00h（通常为 AES-CTR 模式）以外的其他值，CRYTXTB 寄存器会在加密操作完成*后*递增 1。 CRYTXTB 内的计数器大小由 CTRSIZE<6:0> 位的值确定。

注：	用户应确保加密和解密操作中的密钥源、密钥长度、数据模式和初始计数器值均匹配。

4.4.2.1 根据轮密钥生成 AES 解密密钥

要将一个新密钥用于 AES-ECB 或 AES-CBC 解密操作 （包括会话密钥加载）时，在执行首次解密操作前必须先执行 AES 解密密钥扩展操作。该操作的执行顺序是：先选择用于后续解密操作（如果要使用软件密钥，还包括将密钥写入 CRYKEY）的密钥，然后执行 AES 解密密钥扩展操作。请注意， CRYTXTn 寄存器的内容是无关紧要的，因为会在操作过程中被改写。解密密钥扩展后，使用该相同密钥执行 AES-ECB 或 AES-CBC 解密，或会话密钥加载操作时，无需再次执行该操作。

执行 AES 解密密钥扩展操作期间，会改写 CRYKEY 寄存器的内容。如果使用软件密钥，从 AES-ECB 或 AES-CBC 解密操作转换为其他某种模式将会因此而需要重写 CRYKEY 寄存器。AES 解密密钥扩展模式仅在 CPHRSEL = 1 且 CPHRMOD<2:0> = 00x 时有效。任何其他设置将导致 MODFAIL 位置 1。

例 4-8：　　根据加密密钥计算解密密钥

```
/* This example comes from the FIPS-197 standard, Appendix A.1
 * (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).
 */
unsigned char encryption_key[] = {
0x2b, 0x7e, 0x15, 0x16,
0x28, 0xae, 0xd2, 0xa6,
0xab, 0xf7, 0x15, 0x88,
0x09, 0xcf, 0x4f, 0x3c };

CRYCONLbits.CRYON = 0b1;          //Turn module on
CRYCONHbits.KEYSRC = 0b0000;      //Select the key source (CRYKEY)
CRYCONLbits.OPMOD = 0b0010;       //Select the operational mode
CRYCONLbits.CPHRSEL = 0b1;        //Select the cipher (AES)
                                  //(AES decryption key generation)
CRYCONHbits.KEYMOD = 0b00;        //Set the key strength (128-bit key)

memcpy((void*)&CRYKEY0, encryption_key, 16); //Load the key into CRYKEY
                                             //(128-bit key in this example)
CRYCONLbits.CRYGO = 0b1;                     //Start the encryption
while(CRYCONLbits.CRYGO == 0b1){}            //Wait for encryption
                                             //to complete

/* The module is now ready to perform decryption operations. CRYKEY
 * has been updated with the decryption key. You may now perform decryption
 * operations. If you need to switch keys, you will need to recalculate the
 * decryption key if you use the AES engine to calculate it for you.
 * The CRYKEY register is write only so the value of the decryption key
 * can't be read out.
*/
```

4.4.2.2　AES 解密密钥扩展

要将一个新密钥用于 AES-ECB 或 AES-CBC 解密操作（包括会话密钥加载）时，在执行首次解密操作前必须先执行 AES 解密密钥扩展操作。该操作的执行顺序是：先选择用于后续解密操作的密钥（如果要使用软件密钥，还包括将密钥写入 CRYKEY），然后执行 AES 解密密钥扩展操作。请注意， CRYTXTn 寄存器的内容是无关紧要的，因为会在操作期间被改写。

解密密钥扩展后，使用相同密钥执行 AES-ECB 或 AES-CBC 解密，或会话密钥加载操作时，无需再次执行该操作。

执行 AES 解密密钥扩展操作期间，会改写 CRYKEY 寄存器的内容。如果使用软件密钥，从 AES-ECB 或 AES-CBC 解密操作转换为其他某种模式将会因此而需要重写 CRYKEY 寄存器。

AES 解密密钥扩展模式仅在 CPHRSEL = 1 且 CPHRMOD<2:0> = 00x 时有效。任何其他设置都会导致 MODFAIL 位置 1。

4.4.3 随机数生成

加密引擎包含一个基于硬件的随机数发生器（Random Number Generator， RNG）。该模块可生成一个 128 位的随机数。通过将 OPMOD<3:0> 设置为 1010 以及 CRYGO 位置 1 来触发 RNG。当模块成功生成随机数时， CRYGO 位将自动清零，该随机数将放置在 CRYTXTA 寄存器中。

例 4-9： 生成一个随机数

```
/* Turn module on */
CRYCONLbits.CRYON = 0b1;

/* Select to generate a random number */
CRYCONLbits.OPMOD = 0b1010;
/* Start the process */
CRYCONLbits.CRYGO = 1;

/* Wait until the module is done generating the random number */
while(CRYCONLbits.CRYGO == 1){}

/* The random number is now located in the CRYTXTA register set. */
```

4.4.4 会话密钥加密

在会话密钥加密操作中，使用当前的算法、密钥长度和密码模式对软件或硬件生成的会话密钥进行加密。该会话密钥被放入 CRYKEY 寄存器中，然后使用密钥加密密钥 （KEK）来对其加密，加密后的会话密钥写入相应的 CRYTXTn 寄存器 （具体取决于密码模式）。

会话密钥加密完成后，在新的会话密钥可用于加密数据之前，软件必须选择 CRYKEY 寄存器作为加密密钥的源，方法是将 0000 写入 KEYSRC<3:0>。

由于会话密钥像软件密钥一样存储在 CRYKEY 中，要使会话密钥加密操作正常进行，不能将 SWKYDIS 位置 1。

在 OFB 密码模式中不允许对会话密钥进行加密。设置 OPMODE<3:0> = 111x 和 CPHRMOD<2:0> = 011 将导致 MODFAIL 位置 1。

注： 不要在 SKEYEN 置 1 时也将 SWKYDIS 置 1。否则将永久禁止会话密钥操作。

4.4.5 会话密钥加载

在会话密钥加载操作中，使用密钥加密密钥 （KEK）对位于相应的 CRYTXTn 寄存器 （具体取决于密码模式）中的加密后会话密钥进行解密，然后将结果写入 CRYKEY 的低（KEYSEL = 0）或高 （KEYSEL = 1） 128 位。

会话密钥加载 （解密）完成后，在新的会话密钥可用于解密数据之前，软件必须选择 CRYKEY 寄存器作为加密密钥的源，方法是将 0000 写入 KEYSRC<3:0>。

使能会话密钥加载功能 （SKEYEN = 1）时，密钥加密密钥仅可用于会话密钥加密和解密。在该情况下，除了生成会话密钥，硬件将禁止密钥加密密钥用于任何其他的加密或解密操作。这可防止恶意软件将一个加密会话密钥简单地解密到 CRYTXTA 寄存器中。

由于会话密钥是软件密钥，要使会话密钥加密操作正常进行，不能将 SWKYDIS 位置 1。然而，SWKYDIS 只是禁止软件对 CRYKEY 进行写操作，所以当 SWKYDIS 置 1 时，会话密钥加载操作仍可在器件上执行。

在 OFB 密码模式中不允许加载会话密钥。设置 OPMOD<3:0> = 111x 和 CPHRMOD<2:0> = 011 会导致 MODFAIL 位置 1。

注：	当密钥大小超过 128 位时，会话密钥加载在 ECB 或 CBC 模式下不可用。

4.4.6 测试密钥源配置

可始终通过写相应的寄存器位，然后读取 KEYFAIL 寄存器位来测试密钥源配置的有效性。执行该检查无需启动任何操作；甚至不需要使能模块。

4.4.7 验证编程的密钥

为了保持密钥安全性，安全 OTP 阵列不支持以任何操作模式将其数据读回任何用户可访问的存储空间。因此，没有方法可直接验证编程的数据。验证数据已正确编程的唯一方法是：针对每个编程的密钥，对已知的明文 / 密文对执行加密操作。

4.5 操作时间

表 4-3 给出了可执行的不同操作的基本操作时间。要记住，执行这些操作无需 CPU 的介入，并且在执行加密操作时器件将继续执行指令。

请注意，会话密钥加密仅仅是通用加密操作的子集，而会话密钥加载仅仅是通用解密操作的子集。

表 4-3： 近似的操作周期计数

模式	时钟周期数 （近似值）	
	每一块	额外的 加载 / 卸载
DES 加密 / 解密	10(1)	2
3DES 加密 / 解密	26(1)	2
128 位 AES 加密 / 解密	219(2,3)	32
192 位 AES 加密 / 解密	275(2,3)	32
256 位 AES 加密 / 解密	299(2,3)	32
DES 64 位会话密钥加密	10	2
DES 2x 64 位会话密钥加密	10	2
DES 3x 64 位会话密钥加密	20	4
AES 128 位会话密钥加密 （128 位 KEK）	219	32
AES 128 位会话密钥加密 （192 位 KEK）	275	32
AES 128 位会话密钥加密 （256 位 KEK）	299	32
AES 192 位 /256 位会话密钥加密 （128 位 KEK）	438	48
AES 192 位 /256 位会话密钥加密 （192 位 KEK）	550	48
AES 192 位 /256 位会话密钥加密 （256 位 KEK）	598	48
DES 64 位会话密钥加载	10	2
DES 2x 64 位会话密钥加载	10	2
DES 3x 64 位会话密钥加载	20	4
AES 128 位会话密钥加载 （128 位 KEK）	219(3)	32
AES 128 位会话密钥加载 （192 位 KEK）	275(3)	32
AES 128 位会话密钥加载 （256 位 KEK）	299(3)	32
AES 192 位 /256 位会话密钥加载 （128 位 KEK）	438(3)	48
AES 192 位 /256 位会话密钥加载 （192 位 KEK）	550(3)	48
AES 192 位 /256 位会话密钥加载 （256 位 KEK）	598(3)	48

注 1： 64 位块。

2： 128 位块。

3： 不包括切换密钥之后 AES 解密操作初始化所需要的周期数。

4.6 中断

加密引擎生成 4 种中断来指示密钥事件的发生：

- 加密操作完成 （CRYDONIF）
- OTP 操作完成 （KEYSTRIF）
- CRYTXTA 为空 （空闲）（CRYFREEIF）
- CRYTXTB 计满返回事件 （CRYROLLIF）

一些中断可被多个条件触发。用户可能需要使用软件现场或查询 CRYSTAT 和 CRYOTP 寄存器中的其他位来确定中断的确切性质。

4.6.1 操作完成中断

无论当前加密操作何时完成，DONEIE 位（CRYCONL<11>）都会置 1，从而导致产生中断。该中断仅在 CRYGO 位由硬件清零时产生；手动清零 CRYGO 以中止一个操作不会产生中断。

可通过查询 CRYGO 位确认中断源。如有需要，可通过查询 CRYABRT 位 （CRYSTAT<5>）来确认操作的软件终止。

4.6.2 OTP 操作完成中断

无论当前 OTP 编程或读操作何时完成，OTPIE 位（CRYOTP<6>）都会置 1，从而导致产生中断。

执行任何 OTP 操作时，CRYBSY 位（CRYSTAT<7>）就会由硬件置 1 ；当操作完成时，该位自动清零。 CRYREAD 和 CRYWR 位 （CRYOTP<5,0>）指示何时进行 OTP 读或编程操作；它们均自动置 1 和清零。当发生器件上电复位（POR）时，CRYREAD 和 CRYBSY 位也会短暂置 1，并在完成对阵列的初始读取后清零。确认所有 3 位均设置为 0，确保没有 OTP 操作在进行。

4.6.3 CRYTXTA 空闲中断

当 CRYTXTA 寄存器针对某个操作使用完所有数据且处于空闲状态可以将新数据 （明文或者密文）写入其中时， FREEIE 位 （CRYCONL<10>）会置 1，从而导致产生中断。该中断仅在执行ECB和CBC模式操作时发生。在所有其他模式中，CRYTXTA会一直被使用直至或临近操作结束。

TXTABSY 位 （CRYSTAT<6>）指示 CRYTXTA 的状态。当未处理的数据保留在寄存器中时，该位保持置 1 ；所有数据均被处理后，该位自动清零。

TXTABSY 仅在执行 ECB 操作、CBC 加密、CFB 解密或者生成会话密钥操作时有效。在所有其他情况下， CRYGO 置 1 时，应将 CRYTXTA 视为在使用中。

注：	不为 CRYTXTB 和 CRYTXTC 的状态提供单独的中断。CRYGO 位置 1 时，这些寄存器总是被认为处于忙状态。

4.6.4 CRYTXTB 计满返回中断

发生 CRYTXTB 计满返回时， ROLLIE 位 （CRYCONL<12>）会置 1，从而导致产生中断。当然，这只会发生在CRYTXTB用作计数器（CPHRMOD<2:0> = 100）的操作模式下。当CRYTXTB用作单个位计数器 （CTRSIZE<6:0> = 00h）时，中断也不会发生。

当 CRYTXTB 计满返回事件发生时， ROLLOVR 位 （CRYSTAT<4>）置 1。

5.0 休眠和空闲模式下的操作

5.1 休眠模式下的操作

每当器件进入任何休眠或深度休眠模式时，所有操作停止，所有引擎状态机都会复位。通过丢弃任何可用于破解密钥的中间文本，该特性有助于保存被加密或解密的所有数据的完整性。

进入休眠模式时，正在进行的任何 OTP 编程操作也将停止。这可能导致永久丧失一个存储单元或可能使用整个安全 OTP 阵列，具体取决于所编程的内容。建议用户仅在禁止进入节能模式时执行 OTP 编程。

注：	OTP 编程错误，无论来源为何，均不可恢复。用户应确保禁止编程操作的所有可预见中断，包括器件中断和进入节能模式。

5.2 空闲模式下的操作

当 CRYSIDL 位（CRYCONL<13>）为 0 时，器件进入空闲模式后，引擎将继续任何正在进行的操作，而不会中断。

当 CRYSIDL 为 1 时，模块的行为类似于在休眠模式下。

5.3 外设模块禁止 （PMD）寄存器

外设模块禁止（PMD）寄存器提供了一种方法，可通过停止向加密引擎提供的所有时钟源来禁止该模块。当通过将 CRYMD 控制位 （位于 PMDx 寄存器中）置 1 来禁止模块时，外设处于最小功耗状态。与外设相关的控制寄存器和状态寄存器也会被禁止，因此写入这些寄存器不会有任何影响，且读取值无效。

该模块仅在 CRYMD 位清零时使能。

6.0 复位的影响

为了保持加密引擎的安全性，复位状态期间该模块的行为与大部分其他 dsPIC33/PIC24 模块不同。

与大部分 dsPIC33/PIC24 外设一样，发生任何器件复位时，控制寄存器中的所有位均复位为其默认状态。此外，每当模块被禁止 （通过外设模块禁止位 （即 CRYMD = 1））时，大部分的位也将复位。当模块关闭 （CRYON = 0）时一些其他的位也会复位。这些额外的复位条件有助于确保在模块进入停用状态时任何进行中的加密操作都会终止。

OTP 阵列的 Page 0 包含了加密引擎运行时操作的配置信息。器件复位后，必须读取和加载 Page 0 的信息以配置加密引擎。在这个时间间隔内，许多功能都无法使用。在上电复位后，CRYREAD 位自动置 1 以启动一个 OTP 读操作；该操作完成后加密引擎才可用。读操作完成时，CRYREAD 由硬件自动清零。此时，加密引擎可进行操作。从休眠状态唤醒时遵循相同的序列。

在 OTP 读操作期间，其他标志位的行为有所不同：

- CRYREAD （如前所述）和 CRYBSY 位都在初始 OTP 读操作期间置 1 （= 1），并在读操作完成时自动清零。
- PGMFAIL 和 KEYFAIL 位在初始 OTP 读操作期间清零 （= 0），并在读操作完成时它们呈现其相应的值 （基于选择的密钥和 OTP 编程配置）。
- TSTPGM 位在初始 OTP 读操作期间清零 （= 0）。当读操作完成时该位呈现 PGMTST 位的当前值。

7.0 寄存器映射

表 7-1 给出了与加密引擎相关的存储器映射的寄存器和数据空间的汇总。

注： CFGPAGE 不映射到数据存储空间，所以没有出现在这里。关于其结构的详细信息，请参见寄存器 2-5。

表 7-1： 加密引擎寄存器映射

寄存器	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值
CRYCONH	—	CTRSIZE6	CTRSIZE5	CTRSIZE4	CTRSIZE3	CTRSIZE2	CTRSIZE1	CTRSIZE0	SKEYSEL	KEYMOD1	KEYMOD0	—	KEYSRC3	KEYSRC2	KEYSRC1	KEYSRC0	0000
CRYCONL	CRYON	—	CRYSIDL	ROLLIE	DONEIE	FREEIE	—	CRYGO	OPMOD3	OPMOD2	OPMOD1	OPMOD0	CPHRSEL	CPHRMOD2	CPHRMOD1	CPHRMOD0	0000
CRYSTAT	—	—	—	—	—	—	—	—	CRYBSY	TXTABSY	CRYABRT	ROLLOVR	—	MODFAIL	KEYFAIL	PGMFAIL	--00
CRYOTP	—	—	—	—	—	—	—	—	PGMTST	OTPIE	CRYREAD	KEYPG3	KEYPG2	KEYPG1	KEYPG0	CRYWR	--00
CRYKEY	加密密钥寄存器（256 位宽，只写）																xxxx xxxx(1)
CRYTXTA	文本寄存器 A（128 位宽）																xxxx xxxx(2)
CRYTXTB	文本寄存器 B（128 位宽）																xxxx xxxx(2)
CRYTXTC	文本寄存器 C（128 位宽）																xxxx xxxx(2)

图注： — = 未实现，读为 0， x = 未知或未定义的值。复位值用十六进制显示。数据空间的相对大小未按比例显示。

注 1： 复位值是 32 字节的 xx；但是，无法读取实际复位值。

2： 复位值是 16 字节的 xx。

8.0 相关参考资料

关于加密和数据安全的更多信息，请访问美国国家标准与技术研究院 （National Institute of Standards and Technology， NIST）的计算机安全资源中心的网站：

http://csrc.nist.gov/groups/STM/cavp/

网站提供关于 AES （FIPS-197）和三重 DES （NIST SP 800-67）的完整信息，包括验证候选算法的测试向量。还提供了其他数据加密标准的相关信息。

Microchip Technology Inc. 还为其 dsPIC33/PIC24 DSC 和单片机提供了加密软件库。该库包含一个 API，使得应用程序开发者只需开发较少的附加代码即可使用加密引擎。您可从 www.microchipdirect.com 网站购买该软件库的光盘 （部件编号：SW300052）。

注： Microchip Technology Inc. 所有包含加密安全功能的产品均受到许可限制和联邦出口监管。更多信息，请联系 Microchip Technology Inc.。

9.0 版本历史

版本 A （2013 年 9 月）

本文档的初始版本。

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

ISBN：978-1-62077-762-6

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 杭州
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

德国 Germany - Pforzheim
Tel: 49-7231-424750

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Venice
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

波兰 Poland - Warsaw
Tel: 48-22-3325737

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

10/28/13