Application Note

在 AM243x/AM64x 和 AM275x 器件上使用 MCU+ SDK 实现以太网固件空中更新

Aryamaan Chaurasia, Prashant Shivhare, Teja Rowthu, Vaibhav Kumar, Meet Thakar

摘要

本应用手册介绍了如何在德州仪器 (TI) Sitara™ AM243x/AM64x 和 AM275x 微控制器系列上实现基于以太网的固件空中更新功能。此设计利用 TI 的 MCU+ SDK 基础设施，包括 OSPI 驱动程序和接口、Enet 底层驱动栈、引导加载程序框架以及身份验证和加密等安全机制，为工业和汽车应用提供安全的远程固件更新机制。本应用手册旨在帮助系统架构师和嵌入式工程师全面了解 ENET FOTA 如何与 TI 的软件生态系统集成。

内容

商标

1 简介

固件空中更新 (FOTA) 是一种无需物理接触或手动干预即可远程更新电子器件中嵌入式软件的方法。通过以太网传输时，FOTA 充分利用了有线网络的可靠性和速度。这种方法在工业自动化、智能基础设施以及任何大规模或在难以到达位置部署器件的应用中都特别有用。

传统的固件更新通常需要技术人员或工程师使用串行接口、USB 线缆或专用的编程工具直接连接到每个器件。当在工厂车间、楼宇或城市中安装了成百上千个器件时，此过程既耗时又成本高昂，而且不切实际。

基于以太网的固件空中更新 (ENET FOTA) 通过利用标准以太网网络基础设施，扩展了这一远程更新功能。这种方法使配备以太网连接的嵌入式器件能够通过其用于运行通信的同一网络接收完整的固件更新。利用现有的网络基础设施，组织无需安排维护窗口或派遣技术人员前往物理现场，即可在整个器件生态系统中部署关键的安全补丁、功能增强和错误修复。

2 ENET FOTA 机制

ENET FOTA 通过以太网更新固件：接收新映像，使用 x509 证书验证真实性和完整性，将其存储在非易失性闪存存储器中，然后重新引导以运行更新的固件。此过程只需极少的用户干预，同时可确保仅应用授权的更新。

对于 ENET FOTA，AM243x/AM64x 和 AM275x 器件上可用的默认闪存器件是 S28HS512T 串行 NOR OSPI 闪存。图 2-1 中所示的闪存存储器布局用于 ENET FOTA：

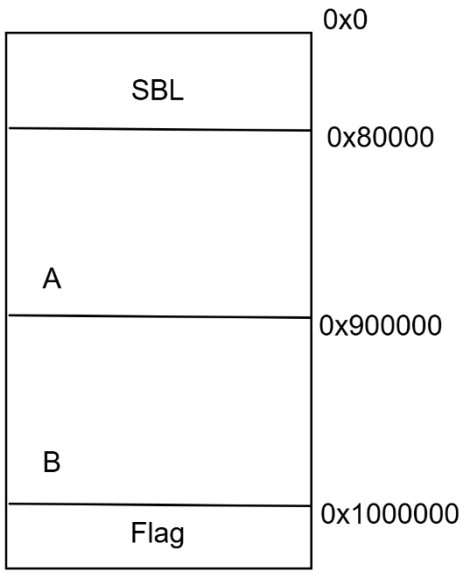


图 2-1. ENET FOTA 的闪存存储器配置

1. 默认情况下，SBL 位于 0x0 处。此偏移量保持不变，因为 ROM 期望 SBL 位于 0x0 偏移处。
2. 将应用映像加载和/或写入存储器位置 A 和 B，它们分别从存储器位置 0x80000 和 0x900000 开始。
3. 存储在 0x1000000 闪存存储器中的 1 字节引导标志，指示 SBL OSPI 引导加载程序在复位后必须从哪个存储器位置引导。

对于 ENET FOTA，执行以下步骤：

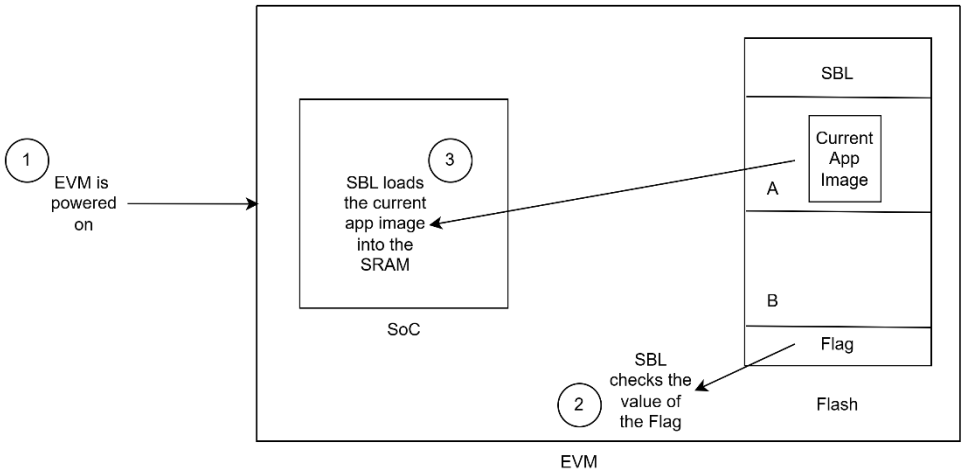


图 2-2. ENET FOTA 的步骤 1-3

EVM 通电，SBL 检查标志的值，并将存储器位置 A 中的当前应用映像加载到 SRAM 中。

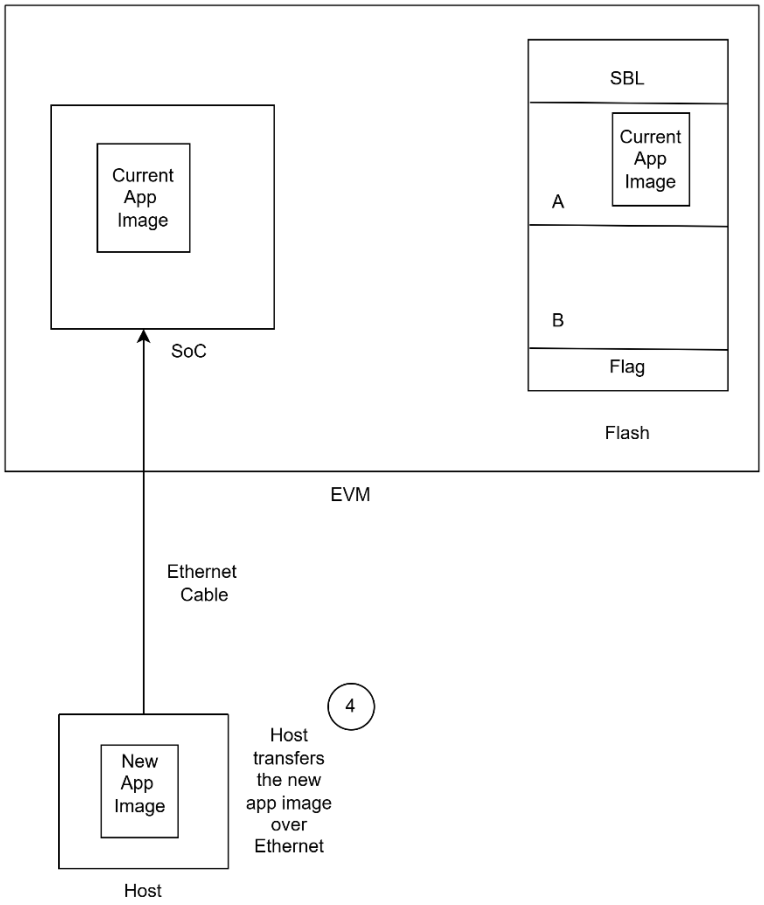


图 2-3. ENET FOTA 的步骤 4

主机通过以太网传输新应用映像。主机和目标器件必须连接到具有兼容网络设置的同一本地网络。当前应用映像会侦听预定义 UDP 端口上的传入固件更新。

固件映像通过 UDP 以可管理数据包的形式进行传输。每个数据包都包含固件段以及时序信息，无论到达顺序或重新传输要求如何，器件都可以重建完整的映像。

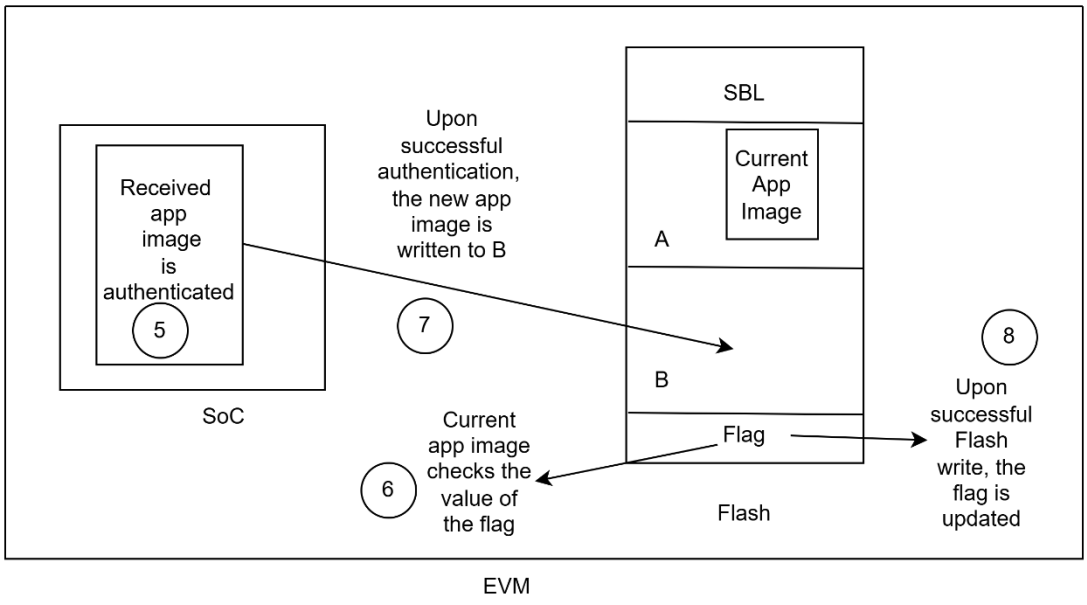


图 2-4. ENET FOTA 的步骤 5-8

应用将传入的数据包存储在 DDR 存储器缓冲区中，并应答来自主机的接收。接收完整的映像后，应用就会对固件进行身份验证，检验其是否源自可信的来源，且在传输过程中未进行修改。

如果身份验证失败，器件会拒绝该固件。成功完成身份验证后，应用会将新固件写入新的闪存存储器位置，并仅在写入成功完成后更新引导标志。

该应用使用双闪存存储器位置：新固件写入新的存储器位置 B，而当前存储器位置 A 保留当前工作的固件。

如果编程过程中发生电源故障或新固件损坏，原始固件在当前存储位置仍保持完好。由于引导标志仅在成功完成后才会更新，因此任何失败都会导致引导原始固件，确保器件保持正常运行。

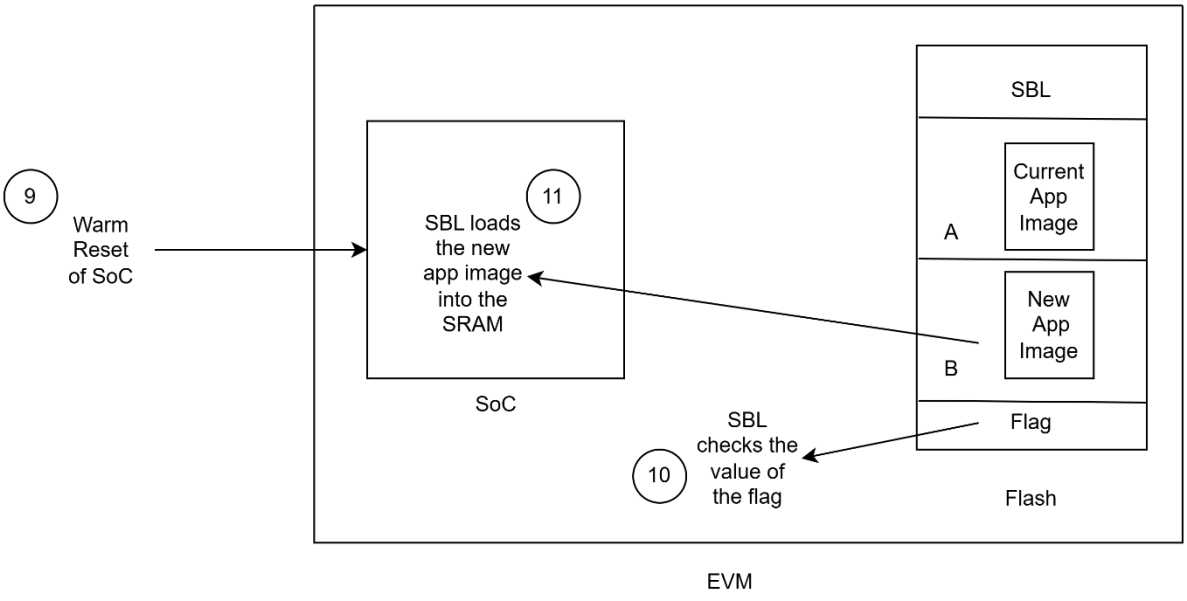


图 2-5. ENET FOTA 的步骤 9-11

标志成功更新后，执行 SoC 热复位。然后，SBL 检查更新的标志值，并将新应用映像从存储器位置 B 加载到 SRAM 中。

同样，从存储器位置 B 运行新应用映像时，传入的应用映像将存储在存储器位置 A。由于位置 A 现在未使用，因此会覆盖之前的旧应用映像。下图展示了 ENET FOTA 在两个存储器位置 A 和 B 之间的切换特性。

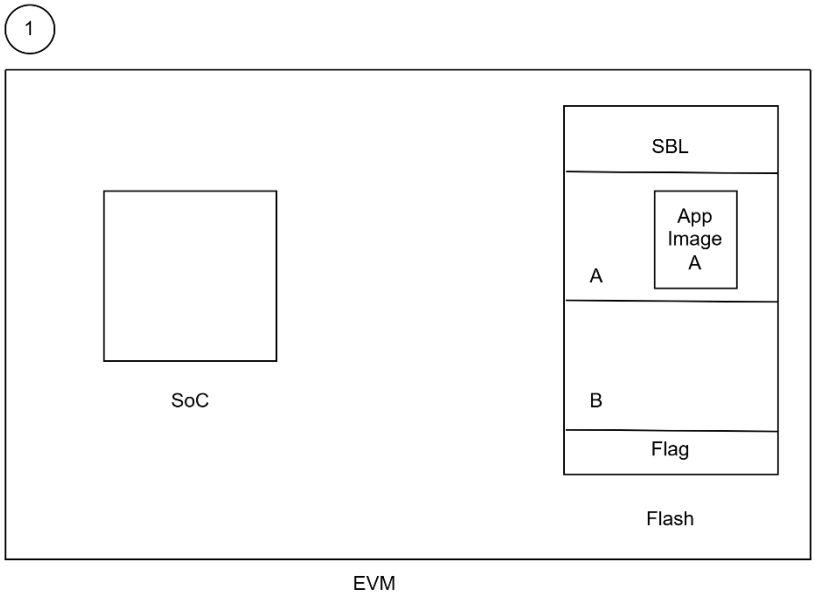


图 2-6. 布局 1

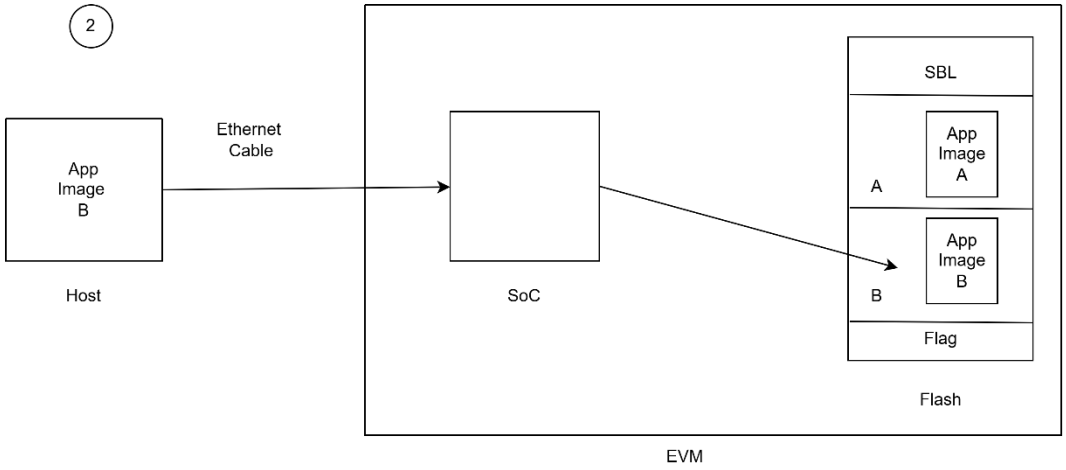


图 2-7. 布局 2

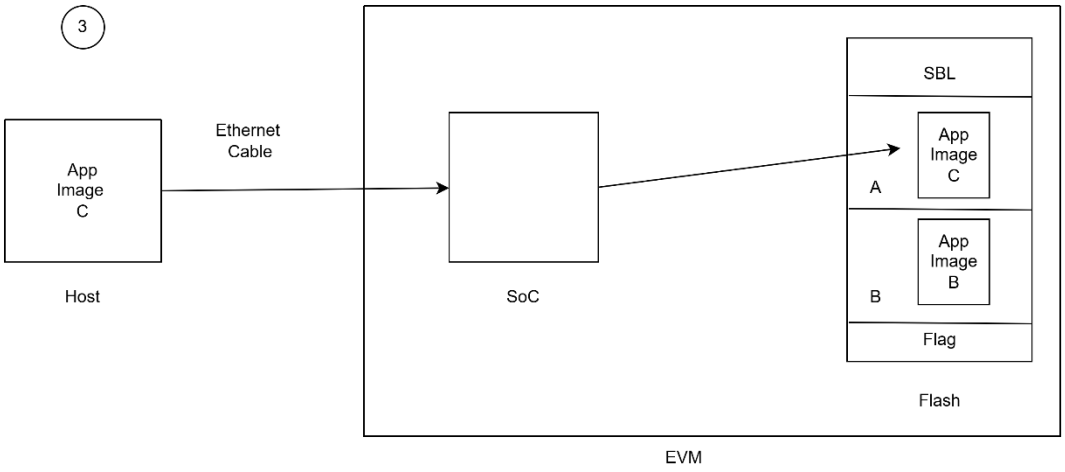


图 2-8. 布局 3

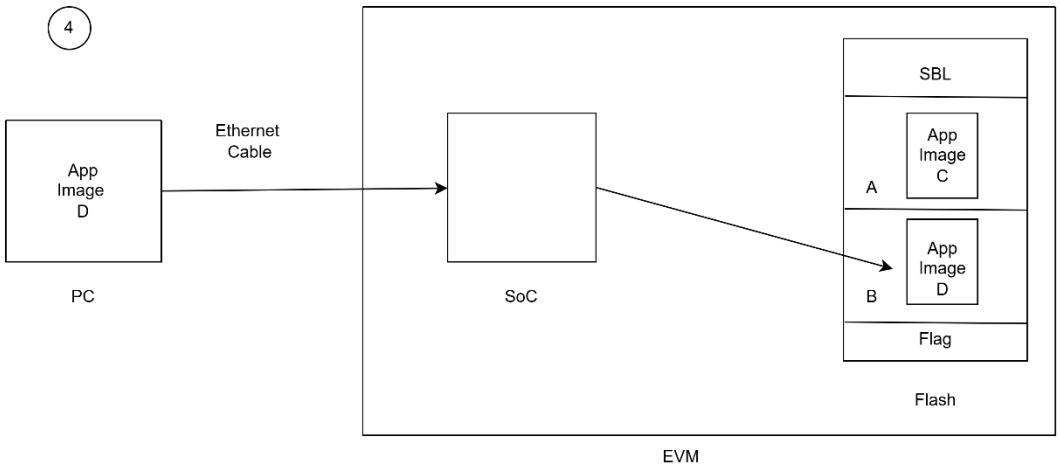


图 2-9. 布局 4

3 以太网设置

要在主机和 EVM 之间成功建立链路，应执行以下步骤：

3.1 Linux 系统

- 在主机和 EVM 之间连接以太网电缆。
- 使用以下命令分配与 EVM 位于同一子网的 IP 地址：

```
sudo ifconfig <接口名称> <主机 IP 地址> <子网掩码>
```

1. 示例：`sudo ifconfig eno1 192.168.0.193 netmask 255.255.255.0`
2. 运行 `ip link` 或 `ifconfig` 命令以查找网络接口的名称。

- 使用以下命令添加静态 ARP 条目：

```
sudo arp -i <接口名称> -s <EVM 的 IP 地址> <EVM 的 MAC 地址>
```

1. 示例：`sudo arp -i eno1 -s 192.168.0.195 44:6b:1f:2c:2d:25`

3.2 Windows 系统

- 在主机和 EVM 之间连接以太网电缆。
- 打开 PC 上的以太网设置 >> 选择相应的以太网适配器。
- 编辑 IP 设置，将 IP 地址设为 192.168.0.193，子网前缀长度设为 24，网关设为 192.168.0.195
- 确保将连接设置为“私有”，并将“计量连接”设置为“关闭”。
- 创建静态 ARP 条目需要管理员权限。在 PowerShell 中以管理员身份运行

  以下命令。

```
New-NetNeighbor -InterfaceIndex <ifIndex> -IPAddress '192.168.0.195'
-LinkLayerAddress '<EVM MAC Addr>' -State Permanent
```

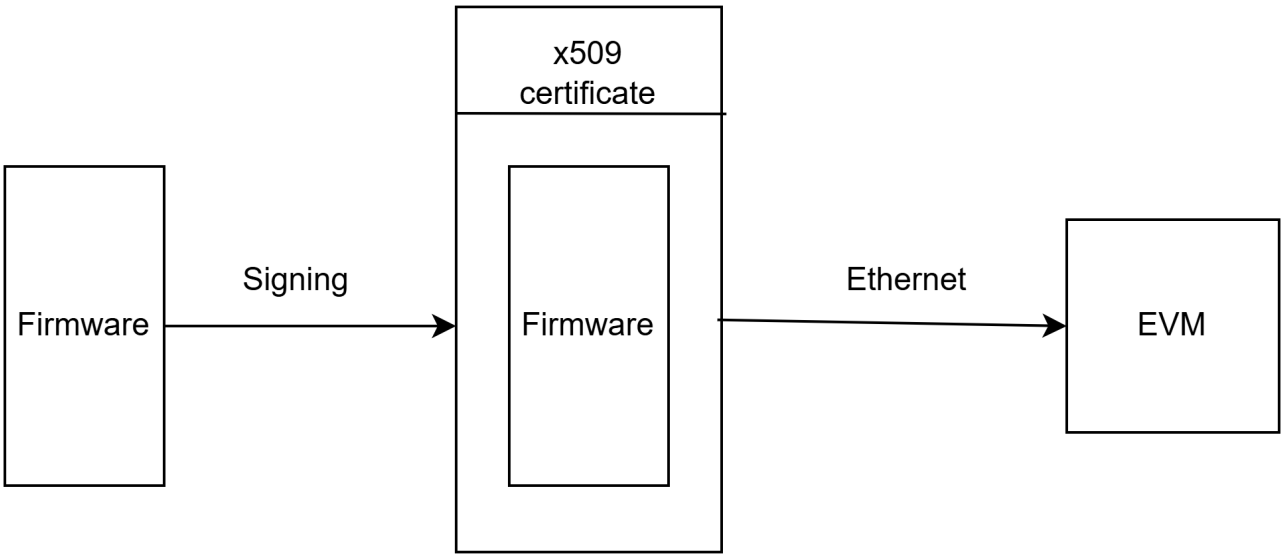
1. 将 `<ifIndex>` 替换为 PC 和 EVM 之间连接的接口索引。
2. 要找出 PC 与 EVM 之间以太网接口对应的接口索引，请使用以下 PowerShell 命令：`Get-NetAdapter`
3. 将 `<EVM MAC Addr>` 替换为 EVM 的 MAC 地址，格式为连续字符串，如 `70ff761decf2`
4. 设置 ARP 的示例命令如下所示：

```
New-NetNeighbor -InterfaceIndex 11 -IPAddress '192.168.0.195' -LinkLayerAddress
'70ff761decf2' -State Permanent
```

备注

对于 Windows 和 Linux 系统，打开 enet_fota.h 文件，并根据您的设定来设置 enet_host_pc_mac_address、enet_port、enet_source_ip_address 和 enet_destination_ip_address 的值；然后，在 enet_fota_file_transfer.py 文件中，将正确的 IP 地址和端口号分配给 hostIP、hostPort、boardIP 和 boardPort 参数。

4 经身份验证的 FOTA



对于经身份验证的 FOTA，要更新的固件会进行后处理，对该固件进行签名以确保真实性。在 EVM 上接收带签名的固件后，会在对实际固件进行编程之前对其进行身份验证。如果身份验证成功，会剥离签名证书，并将固件编程到介质中。

执行上述流程需要以下步骤：

4.1 对固件进行签名

运行以下命令对固件进行签名：

对于 AM243x/AM64x 器件：

- 将目录更改为：`${MCU_PLUS_SDK_PATH}/source/security/security_common/tools/boot/signing`
- 运行以下命令：

```
python appimage_x509_cert_gen.py --bin </path/to/firmware> --authtype 1 --key $(APP_SIGNING_KEY) --output </path/to/signed_firmware>
```

  - 此命令可在所构建示例的 ti-arm-clang 下的 Makefile 中找到。
  - `$(APP_SIGNING_KEY)` 可在 devconfig.mak 中找到，并可根据所使用的器件选择相应的密钥。

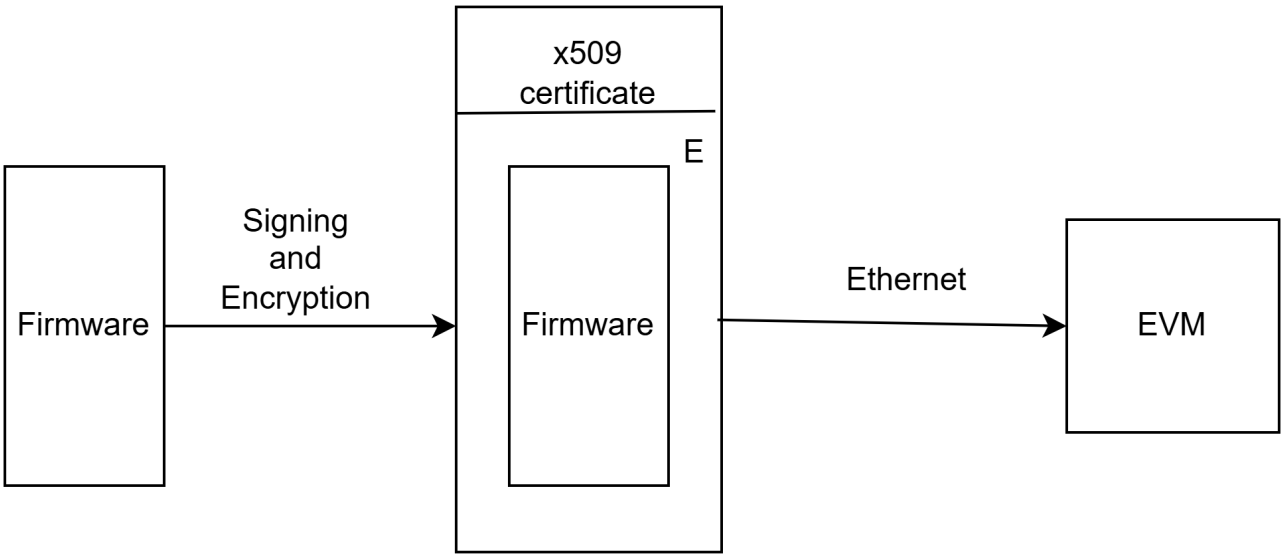
对于 AM275x 器件：

- 将目录更改为：`${FreeRTOS_SDK_PATH}/tools/boot/signing/appimage`
- 运行以下命令：

```
python appimage_x509_cert_gen.py --bin </path/to/firmware> --authtype 1 --key $(APP_SIGNING_KEY) --output </path/to/signed_firmware> --keyversion 1.5
```

  - 此命令可在所构建示例的 ti-arm-clang 下的 Makefile 中找到。
  - `$(APP_SIGNING_KEY)` 可在 devconfig.mak 中找到，并可根据所使用的器件选择相应的密钥。

5 加密的 FOTA



对于加密的 FOTA，要更新的固件会进行后处理，对该固件进行签名和加密，以确保真实性与机密性。在 EVM 上接收带签名的固件后，会在对实际固件进行编程之前对其进行身份验证。如果身份验证成功，会剥离签名证书，并将固件编程到介质中。

执行上述流程需要以下步骤：

5.1 对固件进行签名和加密

要对待传输的应用映像进行加密，请使用上述参数运行以下 python 脚本：

对于 AM243x/AM64x 器件：

- 将目录更改为：`${MCU_PLUS_SDK_PATH}/source/security/security_common/tools/boot/signing`
- 运行以下命令：

```
python appimage_x509_cert_gen.py --bin </path/to/firmware> --authtype 1 --key $(APP_SIGNING_KEY) --enc y --enckey $(APP_ENCRYPTION_KEY) --output </path/to/signed_firmware>
```

  - `$(APP_SIGNING_KEY)` 和 `$(APP_ENCRYPTION_KEY)` 可在 devconfig.mak 中找到，并可根据所使用的器件选择相应的密钥。

对于 AM275x 器件：

- 将目录更改为：`${FreeRTOS_SDK_PATH}/tools/boot/signing/appimage`
- 运行以下命令：

```
python appimage_x509_cert_gen.py --bin </path/to/firmware> --authtype 1 --key $(APP_SIGNING_KEY) --enc y --enckey $(APP_ENCRYPTION_KEY) --output </path/to/signed_firmware> -keyversion 1.5
```

  - `$(APP_SIGNING_KEY)` 和 `$(APP_ENCRYPTION_KEY)` 可在 devconfig.mak 中找到，并可根据所使用的器件选择相应的密钥。

备注

对于经身份验证和加密的 ENET FOTA，必须使用 x509 证书对新应用映像进行签名。建议采用这种做法，因为在应用端执行的身份验证会将新的应用映像以及额外的证书视为单个 blob。然后，对该 blob 进行身份验证，并在执行闪存写入之前剥离额外的证书。如果新应用映像没有额外的证书，则应在 enet_fota_app.c 文件中移除用于移除额外证书的逻辑。如果未移除此逻辑，将剥离原始证书，导致后续的引导失败。

6 软件结构

Beyond SDK Repository 包含 ENET FOTA 的代码实现。

备注

ENET FOTA 和 SBL OSPI 示例与 MCU+SDK 12.00 和 FreeRTOS 12.00 兼容。

6.1 AM243x/AM64x

- ENET FOTA 的 SBL OSPI 位于：
  - Beyond-SDK/am243x/mcupsdk/enet_fota/examples/drivers/boot/sbl_ospi_enet_fota
- ENET FOTA 示例位于：
  - Beyond-SDK/am243x/mcupsdk/enet_fota/examples/enet_fota
- ENET FOTA 示例文件夹还包含 patches 文件夹，其中包含必须应用到 MCU+ SDK 存储库中源文件夹的补丁。
  - 确保在应用补丁后重建所有库，然后构建 SBL OSPI 和 ENET FOTA 示例。
- 将新应用映像从主机传输到 EVM 所需的 python 脚本位于：
  - Beyond-SDK/am243x/mcupsdk/enet_fota/tools/boot/enet_fota_file_transfer.py
- 运行 enet_fota_file_transfer.py python 文件所需的 cfg 文件为：
  - Beyond-SDK/am243x/mcupsdk/enet_fota/tools/boot/sbl_prebuilt/am243x-evm/enet_fota_app.cfg
- 在 enet_fota_app.cfg 文件中添加新应用映像的位置并进行保存。
- 刷写相应的 SBL OSPI 和 ENET FOTA 应用映像。
- 为 EVM 上电后运行以下命令：
  - python enet_fota_file_transfer.py --cfg=sbl_prebuilt/am243x-evm/enet_fota_app.cfg

6.2 AM275x

- ENET FOTA 的 SBL OSPI 位于：
  - Beyond-SDK/am275x/mcupsdk/enet_fota/examples/drivers/boot/sbl_ospi_enet_fota
- SBL OSPI 文件的 SBL OSPI 可在以下位置找到：
  - Beyond-SDK/am275x/mcupsdk/enet_fota/examples/drivers/boot/common/soc/am275x/sbl_ospi_enet_fota.c
- ENET FOTA 示例位于：
  - Beyond-SDK/am275x/mcupsdk/enet_fota/examples/enet_fota
- ENET FOTA 示例文件夹还包含 patches 文件夹，其中包含必须应用到 MCU+ SDK 存储库中源文件夹的补丁。
  - 确保在应用补丁后重建所有库，然后构建 SBL OSPI 和 ENET FOTA 示例。
- 将新应用映像从主机传输到 EVM 所需的 python 脚本位于：
  - Beyond-SDK/am275x/mcupsdk/enet_fota/tool/boot/enet_fota_file_transfer.py
- 运行 enet_fota_file_transfer.py python 文件所需的 cfg 文件为：
  - Beyond-SDK/am275x/mcupsdk/enet_fota/tools/boot/sbl_prebuilt/am275x-evm/enet_fota_app.cfg
- 在 enet_fota_app.cfg 文件中添加新应用映像的位置并进行保存。
- 刷写相应的 SBL OSPI 和 ENET FOTA 应用映像。
- 为 EVM 上电后运行以下命令：
  - python enet_fota_file_transfer.py --cfg=sbl_prebuilt/am275x-evm/enet_fota_app.cfg

7 预期输出

7.1 AM243x/AM64x

R5 日志：

```
Old app image

Listening for new app image

[ ENETFOTA ] Starting Ethernet Transfer ...
Enabling clocks!
Mdio_open:320
Open MAC port 1
EnetPhy_bindDriver:1942
Open MAC port 2
EnetPhy_bindDriver:1942
PHY 0 is alive
PHY 3 is alive
[ ENETFOTA ] EVM MAC address: 34:08:e1:80:a7:c1
[ ENETFOTA ] Please wait for Linkup ...
Cpsw_handleLinkUp:1423
Cpsw_handleLinkDown:1449
Cpsw_handleLinkUp:1423
[ ENETFOTA ] Linkup Done!
[ ENETFOTA ] Receiving file, please wait ...
[ ENETFOTA ] Status:0
[ ENETFOTA SUCCESS ] Ethernet Transfer Done.
[ ENETFOTA ] Packets Received   :  156
[ ENETFOTA ] Total File Size    :  227864 Bytes
Cpsw_handleLinkDown:1449
Disabling clocks for ENET: 5, inst:0!

Image Authentication Successful

Flashed New App Image

--- Time Profile for ENET FOTA ---
Time Taken for Ethernet Setup:               12.100539 s
Time Taken for Transferring New App Image:   0.116197 s
Time Taken for Authenticating New App Image: 0.001007 s
Time Taken for Flashing New App Image:       1.361955 s

SOC Reset

DMSC Firmware Version 12.0.2--v12.00.02 (Clever Cat)
DMSC Firmware revision 0xc
DMSC ABI revision 4.0

BootLoader1 loaded
KPI_DATA: [BOOTLOADER_PROFILE] CPU Clock        : 800.000 MHz
KPI_DATA: [BOOTLOADER_PROFILE] Boot Media       : NOR SPI FLASH
KPI_DATA: [BOOTLOADER_PROFILE] Boot Media Clock : 166.667 MHz
KPI_DATA: [BOOTLOADER_PROFILE] Boot Image Size  : 217 KB
```

```
SOC Reset

DMSC Firmware Version 12.0.2--v12.00.02 (Clever Cat)
DMSC Firmware revision 0xc
DMSC ABI revision 4.0

BootLoader1 loaded
KPI_DATA: [BOOTLOADER_PROFILE] CPU Clock        : 800.000 MHz
KPI_DATA: [BOOTLOADER_PROFILE] Boot Media       : NOR SPI FLASH
KPI_DATA: [BOOTLOADER_PROFILE] Boot Media Clock : 166.667 MHz
KPI_DATA: [BOOTLOADER_PROFILE] Boot Image Size  : 217 KB
KPI_DATA: [BOOTLOADER_PROFILE] Cores present    :
r5f0-0
KPI_DATA: [BOOTLOADER_PROFILE] SYSFW init                      :      10863us
KPI_DATA: [BOOTLOADER_PROFILE] System_init                     :      13899us
KPI_DATA: [BOOTLOADER_PROFILE] Drivers_open                    :       1664us
KPI_DATA: [BOOTLOADER_PROFILE] Board_driversOpen               :      27558us
KPI_DATA: [BOOTLOADER_PROFILE] Sciclient Get Version           :       9938us
KPI_DATA: [BOOTLOADER_PROFILE] CPU load                        :      39671us
KPI_DATA: [BOOTLOADER_PROFILE] SBL End                         :          3us
KPI_DATA: [BOOTLOADER_PROFILE] SBL Total Time Taken            :     103599us

Image loading done, switching to application ...

New app image

Listening for new app image

[ ENETFOTA ] Starting Ethernet Transfer ...
Enabling clocks!
Mdio_open:320
Open MAC port 1
EnetPhy_bindDriver:1942
Open MAC port 2
EnetPhy_bindDriver:1942
PHY 0 is alive
PHY 3 is alive
[ ENETFOTA ] EVM MAC address: 34:08:e1:80:a7:c1
[ ENETFOTA ] Please wait for Linkup ...
Cpsw_handleLinkUp:1423
[ ENETFOTA ] Linkup Done!
[ ENETFOTA ] Status:0
[ ENETFOTA SUCCESS ] Ethernet Transfer Done.
[ ENETFOTA ] Packets Received    :  0
[ ENETFOTA ] Total File Size     :  0 Bytes
Cpsw_handleLinkDown:1449
Disabling clocks for ENET: 5, inst:0!
```

7.2 AM275x

WKUP R5 日志：

```
BootLoader0 loaded
[BOOTLOADER_PROFILE] Boot Media       : FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 166.667 MHz
[BOOTLOADER_PROFILE] Boot Image Size  : 219 KB
[BOOTLOADER_PROFILE] Cores present    :
[BOOTLOADER PROFILE] System_init                    :       2743us
[BOOTLOADER PROFILE] TIFS init                      :        292us
[BOOTLOADER PROFILE] Board_init                     :          7us
[BOOTLOADER PROFILE] FreeRtosTask Create            :        255us
[BOOTLOADER PROFILE] Drivers_open                   :       5785us
[BOOTLOADER PROFILE] Board_driversOpen              :        159us
[BOOTLOADER PROFILE] sciServer_init                 :      15080us
[BOOTLOADER PROFILE] SBL Drivers_open               :       3205us
[BOOTLOADER PROFILE] SBL Board_driversOpen          :       5843us
[BOOTLOADER PROFILE] Sciclient Get Version          :       6604us
[BOOTLOADER PROFILE] R5FSS0_0 Image Load            :       4508us
[BOOTLOADER PROFILE] SBL Total Time Taken           :      44485us

Image loading done...
Starting RTOS/Baremetal applications
Sciserver Testapp Built On: Jun  1 2026 14:46:47
Sciserver Version: v2023.11.0.0REL.MCUSDK.MM.NN.PP.bb
RM_PM_HAL Version: vMM.NN.PP
Starting Sciserver..... PASSED

Starting OSPI Bootloader ...

SYSFW ABI: 4.0 (firmware rev 0x000c '12.0.2--v12.00.02 (Clever Cat)')
BootLoader1 loaded
[BOOTLOADER_PROFILE] Boot Media       : FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 166.667 MHz
[BOOTLOADER_PROFILE] Boot Image Size  : 219 KB
[BOOTLOADER_PROFILE] Cores present    :
[BOOTLOADER PROFILE] System_init                    :       2556us
[BOOTLOADER PROFILE] TIFS init                      :        292us
[BOOTLOADER PROFILE] Board_init                     :          6us
[BOOTLOADER PROFILE] FreeRtosTask Create            :        255us
[BOOTLOADER PROFILE] Drivers_open                   :       5785us
[BOOTLOADER PROFILE] Board_driversOpen              :        160us
[BOOTLOADER PROFILE] sciServer_init                 :      15082us
[BOOTLOADER PROFILE] SBL Drivers_open               :       3211us
[BOOTLOADER PROFILE] SBL Board_driversOpen          :       5828us
[BOOTLOADER PROFILE] Sciclient Get Version          :       6605us
[BOOTLOADER PROFILE] R5FSS0_0 Image Load            :       4512us
[BOOTLOADER PROFILE] SBL Total Time Taken           :      44298us

Image loading done...
Starting RTOS/Baremetal applications
```

R5 日志：

```
Old app image

Listening for new app image

[ ENETFOTA ] Starting Ethernet Transfer ...
Enabling clocks!
EnetAppUtils_reduceCoreMacAllocation: Reduced Mac Address Allocation for CoreId:1 From 4 To 1
Open MAC port 2
EnetPhy_bindDriver:1942
PHY 3 is alive
[ ENETFOTA ] EVM MAC address: 44:6b:1f:2c:2d:25
[ ENETFOTA ] Please wait for Linkup ...
Cpsw_handleLinkUp:1423
Cpsw_handleLinkDown:1449
Cpsw_handleLinkUp:1423
[ ENETFOTA ] Linkup Done!
[ ENETFOTA ] Receiving file, please wait ...
[ ENETFOTA ] Status:0
[ ENETFOTA SUCCESS ] Ethernet Transfer Done.
[ ENETFOTA ] Packets Received    :  158
[ ENETFOTA ] Total File Size     :  229976 Bytes
Cpsw_handleLinkDown:1449
Disabling clocks for ENET: 5, inst:0!

Image Authentication Successful

Flashed New App Image

--- Time Profile for ENET FOTA ---
Time Taken for Ethernet Setup:               12.028640 s
Time Taken for Transferring New App Image:   0.110134 s
Time Taken for Authenticating New App Image: 0.027497 s
Time Taken for Flashing New App Image:       1.407127 s

SOC Reset

New app image

Listening for new app image

[ ENETFOTA ] Starting Ethernet Transfer ...
Enabling clocks!
EnetAppUtils_reduceCoreMacAllocation: Reduced Mac Address Allocation for CoreId:1 From 4 To 1
Open MAC port 2
EnetPhy_bindDriver:1942
PHY 3 is alive
[ ENETFOTA ] EVM MAC address: 44:6b:1f:2c:2d:25
[ ENETFOTA ] Please wait for Linkup ...
```

8 性能指标

表 8-1. OSPI 配置

输入时钟频率	166MHz
PHY	启用
DMA	启用
协议	8D-8D-8D

表 8-2. 时间曲线

以太网设置所需时间	约 8-12 秒
传输新应用映像所需时间	约 0.1 秒
对新应用映像进行身份验证所需时间	约 0.02 秒
将新应用映像写入闪存所需时间	约 1.4 秒

9 总结

ENET FOTA 提供了一种强大、高效且安全的方法，保持嵌入式器件始终处于最新状态。利用有线网络的速度和可靠性，并结合身份验证和加密等安全实践，组织可以保持大型器件部署，而无需承担手动更新的成本和复杂性。本应用手册详细介绍了在 AM243x/AM64x 和 AM275x 器件上实现 ENET FOTA 的机制和步骤。

10 参考资料

1. 德州仪器 (TI)，*AM243x MCU+ SDK*，网页。
2. 德州仪器 (TI)，*AM275x FreeRTOS SDK*，网页。。
3. Github，*Texas Instruments Beyond-SDK Repository*，网页。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月