Application Note

AM243x/AM64x および AM275x デバイスにおける MCU+ SDK を使用したイーサネット経由のファームウェア ワイヤレス (FOTA) 更新

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

概要

このアプリケーション ノートでは、テキサス インスツルメンツの Sitara™AM243x/AM64x および AM275x マイコン ファミリにおけるイーサネット ベースのファームウェアのワイヤレス更新機能の実装について説明します。この設計は、OSPI ドライバおよびインターフェイス、Enet Low レベル ドライバ スタック、ブートローダー フレームワーク、さらには認証や暗号化といったセキュリティ メカニズムを含む TI の MCU+ SDK インフラストラクチャを活用し、産業および車載アプリケーション向けのセキュアなリモート ファームウェア更新メカニズムを実現します。このアプリケーション ノートは、ENET FOTA が TI のソフトウェア エコシステムとどのように統合されるかについて、システム アーキテクトや組込みエンジニアが包括的に理解できるようにするものです。

目次

商標

1 概要

ファームウェア ワイヤレス更新 (Firmware Over-the-Air、FOTA) とは、物理的なアクセスや手動の介入を必要とせずに、電子デバイスに組み込まれているソフトウェアをリモート更新する方法です。イーサネットを通じて実現される場合、FOTA は有線ネットワークの信頼性と速度を利用します。このアプローチは、産業用オートメーション、スマート インフラストラクチャ、およびデバイスが大規模に導入されているアプリケーションやアクセスが困難な場所に設置されるあらゆるアプリケーションにおいて特に有用です。

従来型のファームウェア アップデートでは、多くの場合、技術者やエンジニアはシリアル インターフェース、USB ケーブル、または専用のプログラミング ツールを使用して各デバイスに直接接続する必要があります。このプロセスは、工場の現場、ビル、都市全体に数百または数千台のデバイスを設置する場合、時間がかかり、コストが高く、実用的ではありません。

イーサネット ベースのファームウェア ワイヤレス更新 (Ethernet-Based Firmware Over-The-Air、ENET FOTA) は、標準的なイーサネット ネットワーク インフラストラクチャを活用することで、このリモート更新機能を拡張します。このアプローチにより、イーサネット接続を備えた組込みデバイスは、運用上の通信に使用されるのと同じネットワーク経由で完全なファームウェア更新を受信できます。既存のネットワーク インフラストラクチャを利用することで、組織は、保守時間のスケジュールを設定したり、技術者を物理的な場所に派遣したりすることなく、デバイス エコシステム全体に重要なセキュリティ パッチ、機能拡張、およびバグ修正を展開できます。

2 ENET FOTA のメカニズム

ENET FOTA は、新しいイメージを受信し、x509 証明書を使用して真正性と整合性を検証し、不揮発性のフラッシュ メモリに保存した後、再起動して更新されたファームウェアを実行することにより、イーサネット経由でファームウェアを更新します。このプロセスでは、承認された更新のみが適用されることを検証しつつ、ユーザーの介入を最小限に抑えます。

ENET FOTA の場合、AM243x/AM64x および AM275x デバイスで利用可能なデフォルトのフラッシュ部品は、S28HS512T シリアル NOR OSPI フラッシュです。図 2-1 に示すフラッシュ メモリのレイアウトは、ENET FOTA に使用されます。

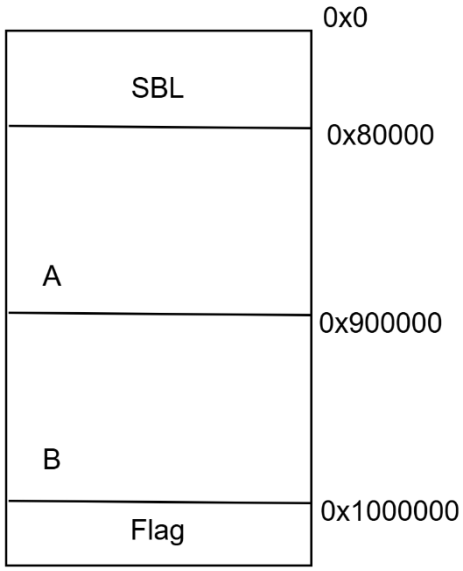


図 2-1. ENET FOTA のフラッシュ メモリ構成

1. SBL は、デフォルトで 0x0 に配置されています。ROM は SBL が 0x0 オフセットに存在することを想定しているため、このオフセットは変更されません。
2. アプリケーション イメージは、メモリ位置 A および B にロードまたは書き込まれます。これらはそれぞれメモリ位置 0x80000 および 0x900000 から始まります。
3. 0x1000000 フラッシュ メモリに保存された 1 バイトのブート フラグは、SBL OSPI ブートローダーがリセット後にどのメモリ位置からブートする必要があるかを示します。

ENET FOTA では、次の手順を実行します。

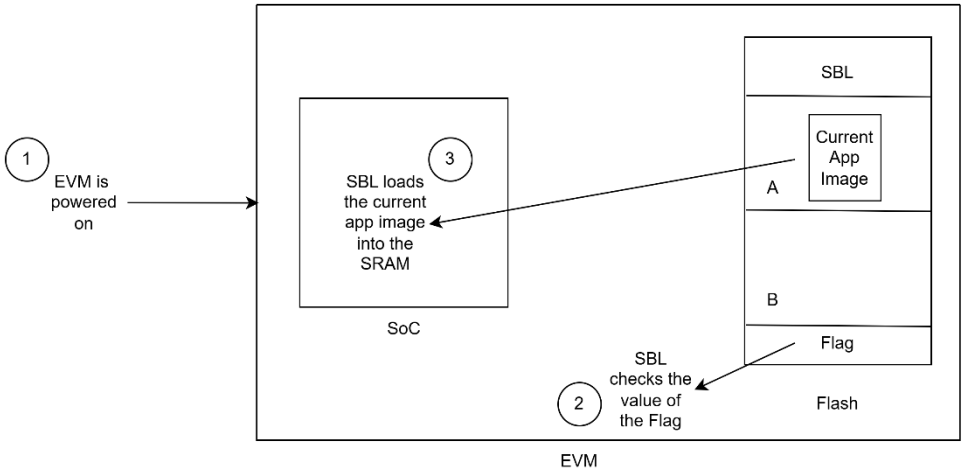


図 2-2. ENET FOTA のステップ 1 ～ 3

評価基板の電源がオンになり、SBL がフラグの値をチェックし、メモリ位置 A に存在する現在のアプリケーション イメージを SRAM にロードします。

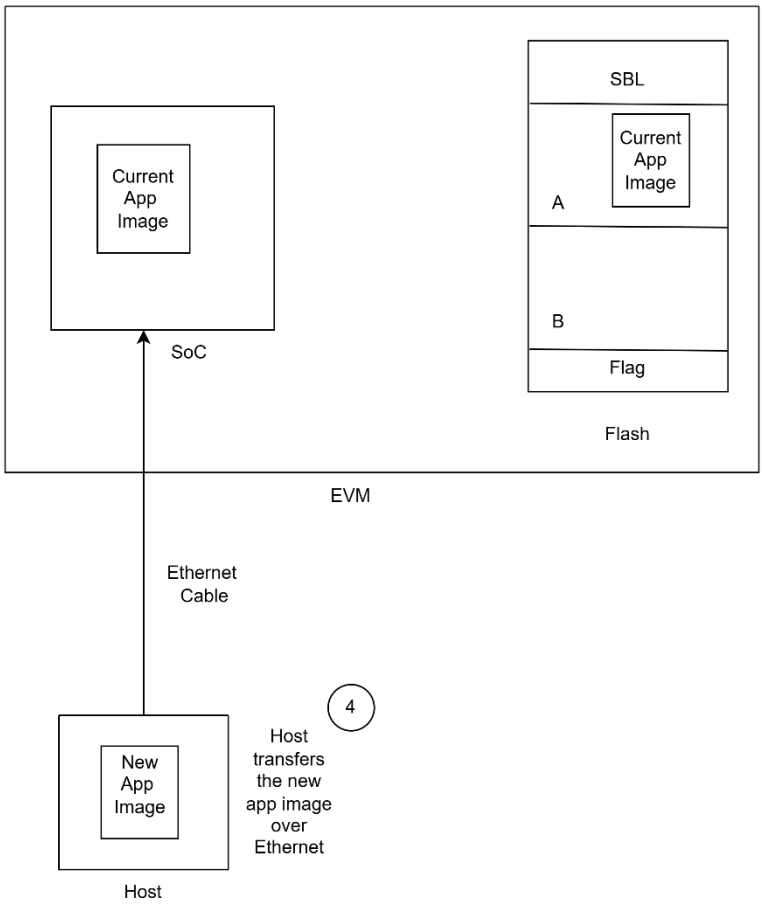


図 2-3. ENET FOTA のステップ 4

ホストが、新しいアプリケーション イメージをイーサネット経由で転送します。ホストとターゲット デバイスは、互換性のあるネットワーク設定で同じローカル ネットワークに接続する必要があります。現在のアプリケーション イメージが、事前定義された UDP ポートで受信ファームウェアの更新をリッスンします。

ファームウェア イメージが、管理可能なパケットで UDP を介して送信されます。各パケットには、ファームウェア セグメントとシーケンシング情報が含まれており、到着順序や再送信の必要性にかかわらず、デバイスはイメージ全体を再構成できます。

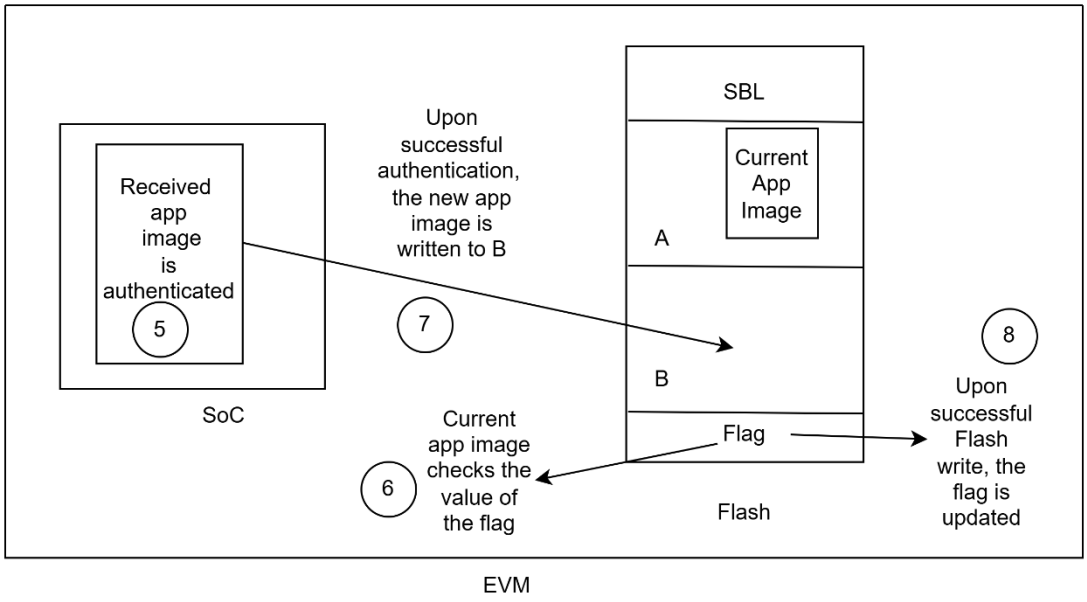


図 2-4. ENET FOTA のステップ 5 ～ 8

アプリケーションが受信パケットを DDR メモリ バッファに格納し、ホストからの受信をアクノリッジします。完全なイメージを受信すると、アプリケーションはファームウェアを認証し、それが信頼できる送信元からのものであり、かつ転送中に改ざんされていないことを検証します。

認証が失敗した場合、デバイスはファームウェアを拒否します。認証が成功すると、アプリケーションは新しいファームウェアを新しいフラッシュ メモリ位置に書き込み、書き込みが正常に完了した後にのみブート フラグを更新します。

アプリケーションはデュアル フラッシュ メモリ位置を使用します。この領域では、新しいファームウェアが新しいメモリ位置 B に書き込まれ、現在のメモリ位置 A は現 在動作しているファームウェアを維持します。

プログラミング中に電源が切断された場合、または新しいファームウェアが破損した場合、元のファームウェアは現在のメモリ位置でもそのまま残ります。ブート フラグが正常に完了した後にのみ更新されるため、障害が発生すると元のファームウェアがブートし、デバイスが動作可能なまま維持されます。

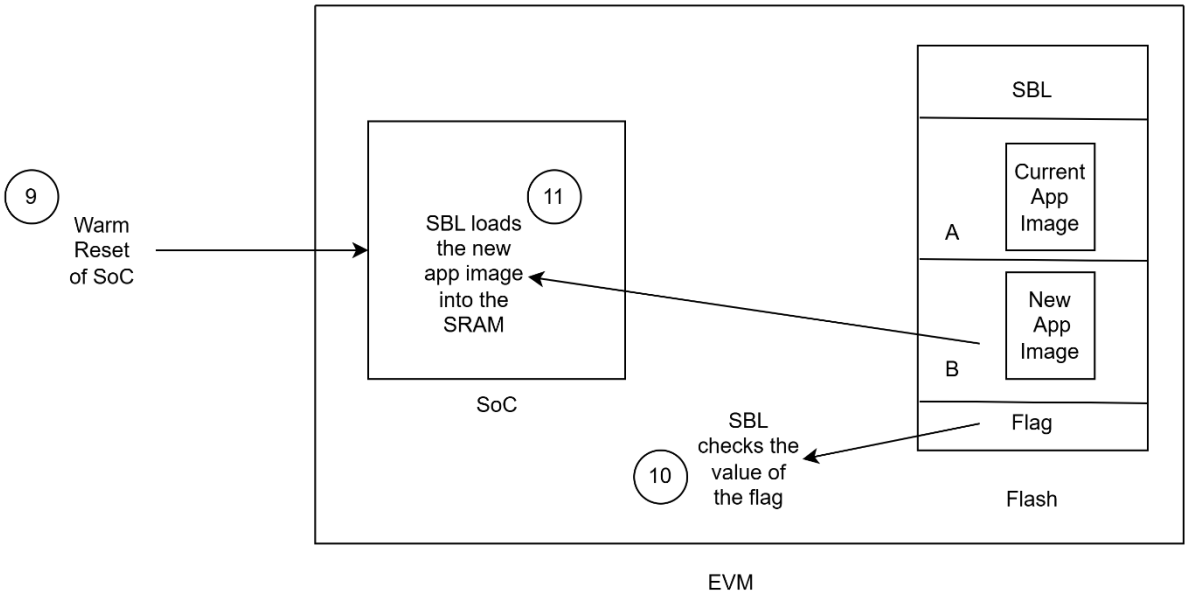


図 2-5. ENET FOTA のステップ 9 ～ 11

このフラグが正常に更新された後、SoC のウォーム リセットが実行されます。次に、SBL は更新されたフラグの値をチェックし、メモリ位置 B から SRAM に新しいアプリケーション イメージをロードします。

同様に、新しいアプリケーション イメージがメモリ位置 B から実行されている場合、受信アプリケーション イメージがメモリ位置 A に格納されます。位置 A は使用されていないため、以前に存在していた古いアプリケーション イメージが上書きされます。次の図は、A と B の 2 つのメモリ位置における ENET FOTA のスイッチング特性を示しています。

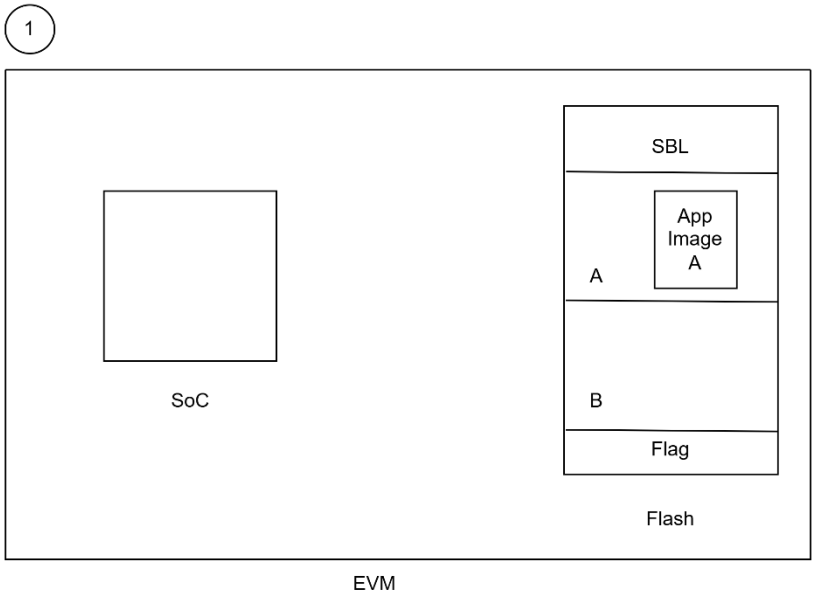


図 2-6. レイアウト 1

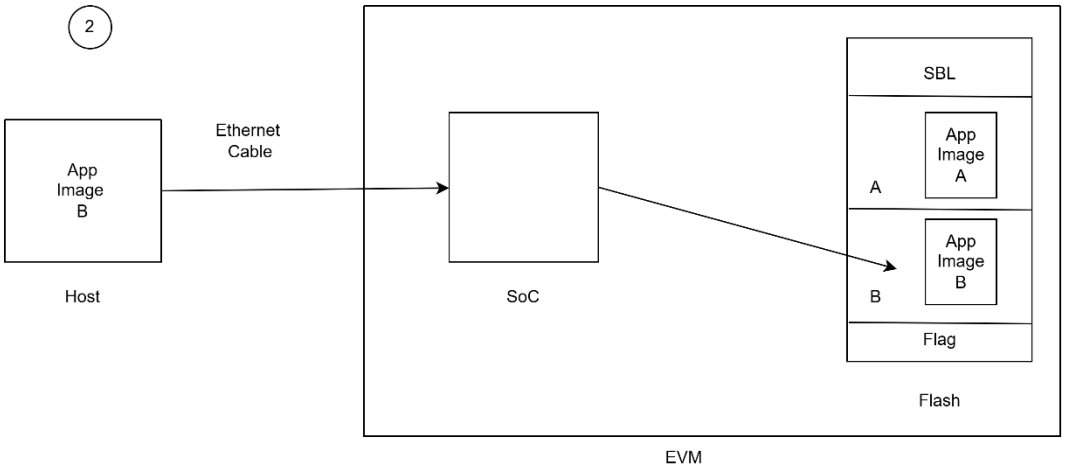


図 2-7. レイアウト 2

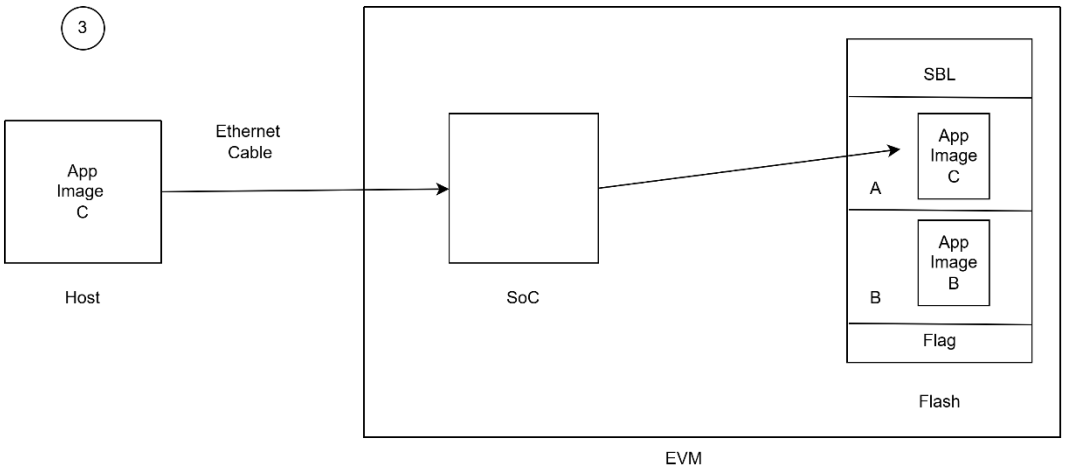


図 2-8. レイアウト 3

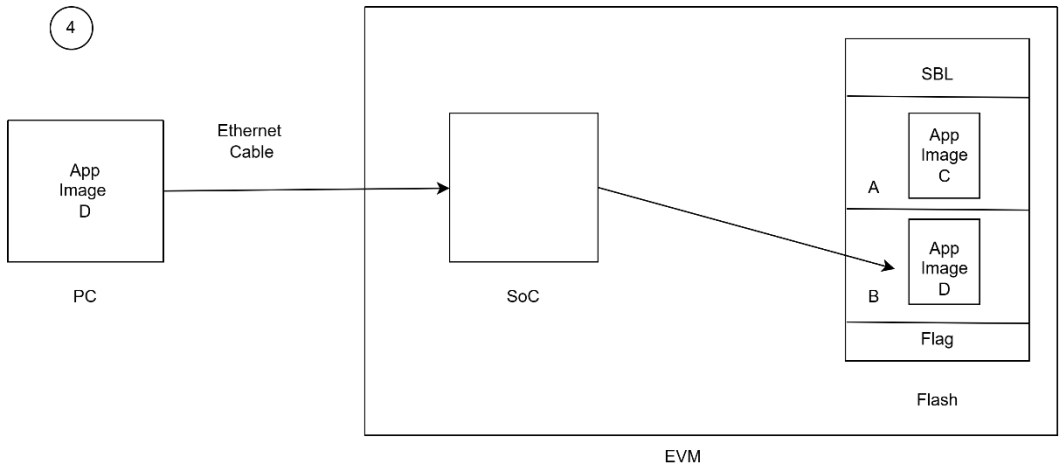


図 2-9. レイアウト 4

3 イーサネットの設定

ホストと評価基板を正常にリンクさせるには、以下の手順を実行する必要があります。

3.1 Linux システム

- ホストと評価基板をイーサネット ケーブルで接続します。
- 次のコマンドを使用して、評価基板と同じサブネットを持つ IP アドレスを割り当てます。

```
sudo ifconfig <interface-name> <Host IP Address> <Subnet mask>
```

1. 例:`sudo ifconfig eno1 192.168.0.193 netmask 255.255.255.0`
2. `ip link` または `ifconfig` コマンドを実行して、ネットワーク インターフェイスの名前を検索します。

- 次のコマンドを使用して、静的 ARP エントリを追加します。

```
sudo arp -i <interface-name> -s <IP Address of EVM> <MAC-Address of EVM>
```

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

3.2 Windows システム

- ホストと評価基板をイーサネット ケーブルで接続します。
- PC でイーサネット設定を開きます >> 対応するイーサネット アダプタを選択します。
- IP アドレスを 192.168.0.193、サブネット プレフィックス長を 24、ゲートウェイを 192.168.0.195 に設定して、IP 設定を編集します
- 接続を「プライベート」に設定し、「従量制課金接続」の設定がオフになっていることを確認します。
- 静的 ARP エントリを作成するには、管理者権限が必要です。PowerShell で次のコマンドを

  管理者として実行します。

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

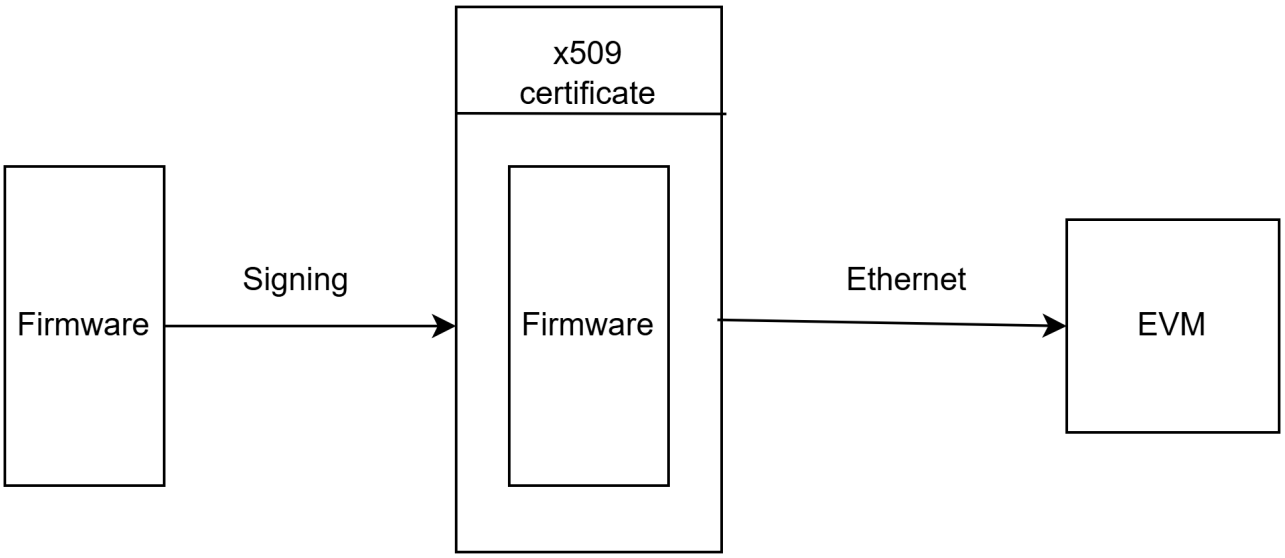
1. `<ifIndex>` を、PC と評価基板間の接続のインターフェイス インデックスに置き換えます。
2. PC と評価基板の間のイーサネット インターフェイスに対応するインターフェイス インデックスを確認するには、次の PowerShell コマンドを使用します: `Get-NetAdapter`
3. `<EVM MAC Addr>` を、`70ff761decf2` のような連続した文字列形式の評価基板の MAC アドレスに置き換えます。
4. ARP を設定するコマンドの例は次のとおりです。

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

注

Windows システムと Linux のいずれのシステムにおいても、enet_fota.h ファイルを開き、設定に従って enet_host_pc_mac_address、enet_port、enet_source_ip_address、および enet_destination_ip_address の値を設定します。次に、enet_fota_file_transfer.py ファイルで、hostIP、hostPort、boardIP、および boardPort の引数に正しい IP アドレスとポート番号を割り当てます。

4 認証済み FOTA



認証済み FOTA の場合、真正性を確保するためにファームウェアに署名するため、更新対象のファームウェアが後処理されます。署名済みファームウェアが評価基板上で受信されると、実際のファームウェアがプログラムされる前に認証されます。認証が成功すると、署名証明書が削除され、ファームウェアがメディアにプログラムされます。

上記のフローを実行するには、次の手順が必要です。

4.1 ファームウェアへの署名

次のコマンドを実行して、ファームウェアに署名します。

AM243x/AM64x デバイスの場合:

- ディレクトリを次のように変更します: `${MCU_PLUS_SDK_PATH}/source/security/security_common/tools/boot/signing`
- 次のコマンドを実行します。

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

  - このコマンドは、ビルドされているサンプルの ti-arm-clang 下の Makefile にあります。
  - devconfig.mak には `$(APP_SIGNING_KEY)` があり、使用するデバイスに応じて対応するキーを選択できます。

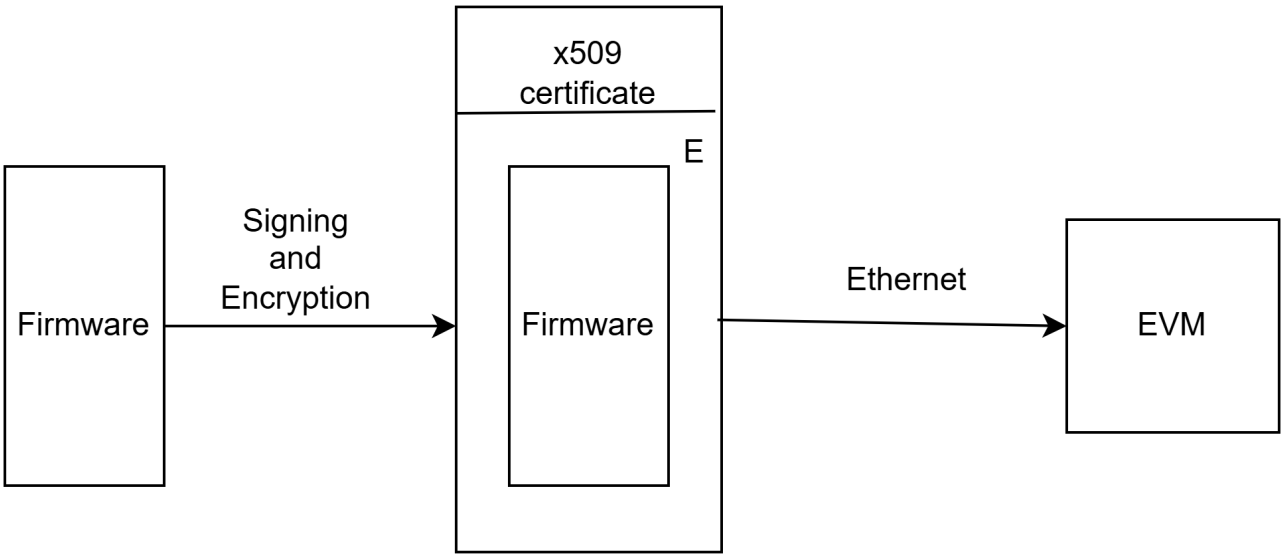
AM275x デバイスの場合:

- ディレクトリを次のように変更します: `${FreeRTOS_SDK_PATH}/tools/boot/signing/appimage`
- 次のコマンドを実行します。

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

  - このコマンドは、ビルドされているサンプルの ti-arm-clang 下の Makefile にあります。
  - devconfig.mak には `$(APP_SIGNING_KEY)` があり、使用するデバイスに応じて対応するキーを選択できます。

5 暗号化済み FOTA



暗号化された FOTA の場合、真正性と機密性を確保するためにファームウェアに署名し暗号化するために、更新対象のファームウェアが後処理されます。署名済みファームウェアが評価基板上で受信されると、実際のファームウェアがプログラムされる前に認証されます。認証が成功すると、署名証明書が削除され、ファームウェアがメディアにプログラムされます。

上記のフローを実行するには、次の手順が必要です。

5.1 ファームウェアへの署名と暗号化

転送するアプリケーション イメージを暗号化するには、次の引数を指定して Python スクリプトを実行します。

AM243x/AM64x デバイスの場合:

- ディレクトリを次のように変更します: `${MCU_PLUS_SDK_PATH}/source/security/security_common/tools/boot/signing`
- 次のコマンドを実行します。

```
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>
```

  - devconfig.mak には `$(APP_SIGNING_KEY)` と `$(APP_ENCRYPTION_KEY)` があり、使用するデバイスに応じて対応するキーを選択できます。

AM275x デバイスの場合:

- ディレクトリを次のように変更します: `${FreeRTOS_SDK_PATH}/tools/boot/signing/appimage`
- 次のコマンドを実行します。

```
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
```

  - devconfig.mak には `$(APP_SIGNING_KEY)` と `$(APP_ENCRYPTION_KEY)` があり、使用するデバイスに応じて対応するキーを選択できます。

注

認証および暗号化された ENET FOTA の場合は、x509 証明書を使用して新しいアプリケーション イメージに署名する必要があります。アプリケーション側で実行される認証では、追加の証明書とともに新しいアプリケーション イメージが単一の BLOB として処理されるため、この方法が推奨されています。この BLOB が認証され、フラッシュ書き込みを実行する前に余分な証明書が削除されます。新しいアプリケーション イメージに追加の証明書がない場合は、enet_fota_app.c ファイルから、その追加の証明書を削除するロジックを削除する必要があります。これが削除されない場合、元の証明書は削除され、その後のブートは失敗します。

6 ソフトウェアの構造

Beyond SDK リポジトリには ENET FOTA の実装コードが含まれています。

注

ENET FOTA および SBL OSPI のサンプルは、MCU+SDK 12.00 や FreeRTOS 12.00 と互換性があります。

6.1 AM243x/AM64x

- ENET FOTA の SBL OSPI は、次の場所にあります。
  - Beyond-SDK/am243x/mcupsdk/enet_fota/examples/drivers/boot/sbl_ospi_enet_fota
- ENET FOTA のサンプルは、次の場所にあります。
  - Beyond-SDK/am243x/mcupsdk/enet_fota/examples/enet_fota
- ENET FOTA のサンプル フォルダには、MCU+ SDK リポジトリ内のソース フォルダに適用する必要があるパッチを格納した「patches」フォルダも含まれています。
  - パッチを適用した後で、すべてのライブラリが再ビルドされていることを確認してから、SBL OSPI および ENET FOTA サンプルをビルドします。
- 新しいアプリケーション イメージをホストから評価基板に転送するために必要な Python スクリプトは、次の場所にあります。
  - Beyond-SDK/am243x/mcupsdk/enet_fota/tools/boot/enet_fota_file_transfer.py
- enet_fota_file_transfer.py Python ファイルの実行に必要な cfg ファイルは次のとおりです。
  - Beyond-SDK/am243x/mcupsdk/enet_fota/tools/boot/sbl_prebuilt/am243x-evm/enet_fota_app.cfg
- enet_fota_app.cfg ファイルに新しいアプリケーション イメージの場所を追加して保存します。
- 対応する SBL OSPI および ENET FOTA アプリケーション イメージを書き込みます。
- 評価基板の電源をオンにした後、次のコマンドを実行します。
  - python enet_fota_file_transfer.py --cfg=sbl_prebuilt/am243x-evm/enet_fota_app.cfg

6.2 AM275x

- ENET FOTA の SBL OSPI は、次の場所にあります。
  - Beyond-SDK/am275x/mcupsdk/enet_fota/examples/drivers/boot/sbl_ospi_enet_fota
- SBL OSPI の SBL OSPI ファイルは、次の場所にあります。
  - Beyond-SDK/am275x/mcupsdk/enet_fota/examples/drivers/boot/common/soc/am275x/sbl_ospi_enet_fota.c
- ENET FOTA のサンプルは、次の場所にあります。
  - Beyond-SDK/am275x/mcupsdk/enet_fota/examples/enet_fota
- ENET FOTA のサンプル フォルダには、MCU+ SDK リポジトリ内のソース フォルダに適用する必要があるパッチを格納した「patches」フォルダも含まれています。
  - パッチを適用した後で、すべてのライブラリが再ビルドされていることを確認してから、SBL OSPI および ENET FOTA サンプルをビルドします。
- 新しいアプリケーション イメージをホストから評価基板に転送するために必要な Python スクリプトは、次の場所にあります。
  - Beyond-SDK/am275x/mcupsdk/enet_fota/tool/boot/enet_fota_file_transfer.py
- enet_fota_file_transfer.py Python ファイルの実行に必要な cfg ファイルは次のとおりです。
  - Beyond-SDK/am275x/mcupsdk/enet_fota/tools/boot/sbl_prebuilt/am275x-evm/enet_fota_app.cfg
- enet_fota_app.cfg ファイルに新しいアプリケーション イメージの場所を追加して保存します。
- 対応する SBL OSPI および ENET FOTA アプリケーション イメージを書き込みます。
- 評価基板の電源をオンにした後、次のコマンドを実行します。
  - python enet_fota_file_transfer.py --cfg=sbl_prebuilt/am275x-evm/enet_fota_app.cfg

7 想定出力

7.1 AM243x/AM64x

R5 ログ :

```
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 ログ :

```
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 ログ :

```
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 性能指標

表 8-1. OSPI の構成

入力クロック周波数	166MHz
PHY	有効
DMA	有効
プロトコル	8D-8D-8D

表 8-2. 時間プロファイル

イーサネットの設定にかかる時間	約 8 ～ 12 秒
新しいアプリケーション イメージの転送にかかる時間	約 0.1 秒
新しいアプリケーション イメージの認証にかかる時間	約 0.02 秒
フラッシュへの新しいアプリケーション イメージの書き込みにかかる時間	約 1.4 秒

9 まとめ

ENET FOTA は、組込みデバイスを最新の状態に維持するための強力で効率的、かつ安全な方法を提供します。有線ネットワークの速度と信頼性を活用し、認証や暗号化などのセキュリティ対策と組み合わせることで、手動更新のコストや複雑さを伴わずに大規模なデバイス展開を維持できます。このアプリケーション ノートでは、AM243x/AM64x および AM275x デバイスに ENET FOTA を実装するメカニズムと手順について詳細に説明しています。

10 参考資料

1. テキサス インスツルメンツ、『AM243x MCU+ SDK』、Web ページ。
2. テキサス インスツルメンツ、『AM275x FreeRTOS SDK』、Web ページ。.
3. Github、『テキサス インスツルメンツ Beyond-SDK リポジトリ』、Web ページ。

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月