MA35D1 TF-A FIP 映像備份功能

NuMicro® 64 位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例代碼透過 TF-A 中的 BL2 實現 FIP 映像的備份功能，確保在主要映像壞損的情形下仍能成功啟動。
BSP 版本	Linux-5.10.x
開發平台	NuMaker-HMI-MA35D1 V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本範例代碼實作了 Trusted Firmware-A (TF-A) 的備份功能，當 TF-A 中的 BL2 檢測到 FIP 映像載入失敗或者安全啟動的簽章驗證失敗時，會嘗試載入備份區域的映像來完成系統啟動。

TF-A 是 Arm® 官方提供的引導程式框架，用來實作晶片的安全開機流程，依據 Platform Security Architecture (PSA) 定義的架構，包含下列執行階段：

- BL1 (ROM code) ：晶片開機後第一個執行的 bootloader，MA35D1 實現於內部的 Mask ROM 當中，不可修改。
- BL2 (Trusted Bootloader)：由 BL1 載入與驗證，負責後續映像（如 FIP、SCP）的載入與驗證。
- BL31 (EL3 Runtime Firmware)：負責提供安全監控與 context switch 的功能。
- BL32：執行於 Secure World 的可信任執行環境，MA35D1 採用開源的 OP-TEE 。
- BL33 : Non-secure World 的 bootloader，MA35D1 採用 U-Boot。

Firmware Image Package (FIP) 是一種容器格式，用以打包多個 TF-A 映像如 BL31、BL32 以及 BL33，供 BL2 一次驗證與載入。其結構定義在 fiptool 工具中。

BL2 是 ARM TF-A 架構中非常關鍵的開機階段。它由 BL1 呼叫執行，主要負責以下任務：

- 載入 FIP(Image List Search) 中的映像檔並透過 ECDSA 算法驗證。
- 確保 Secure Boot 可信任鏈（Chain of Trust）持續成立。
- 初始化必要硬體與記憶體控制器（如 DDR）。
- 啟動下一階段的映像（BL31/BL32/BL33）。

1.1 原理

MA35D1 完整的啟動流程如下圖，每次上電都由 Mask ROM 啟動，當偵測到”secure boot”功能開啟時，第一階段由 Mask ROM 對 BL2 做檢驗，透過驗證 BL2 映像附帶的 ECDSA P-256 簽章，來確認該映像的完整性以及真實性，驗證成功後將 BL2 載入到系統指定的記憶體位址上，便開始由 BL2 接管後續 FIP 映像的載入過程。

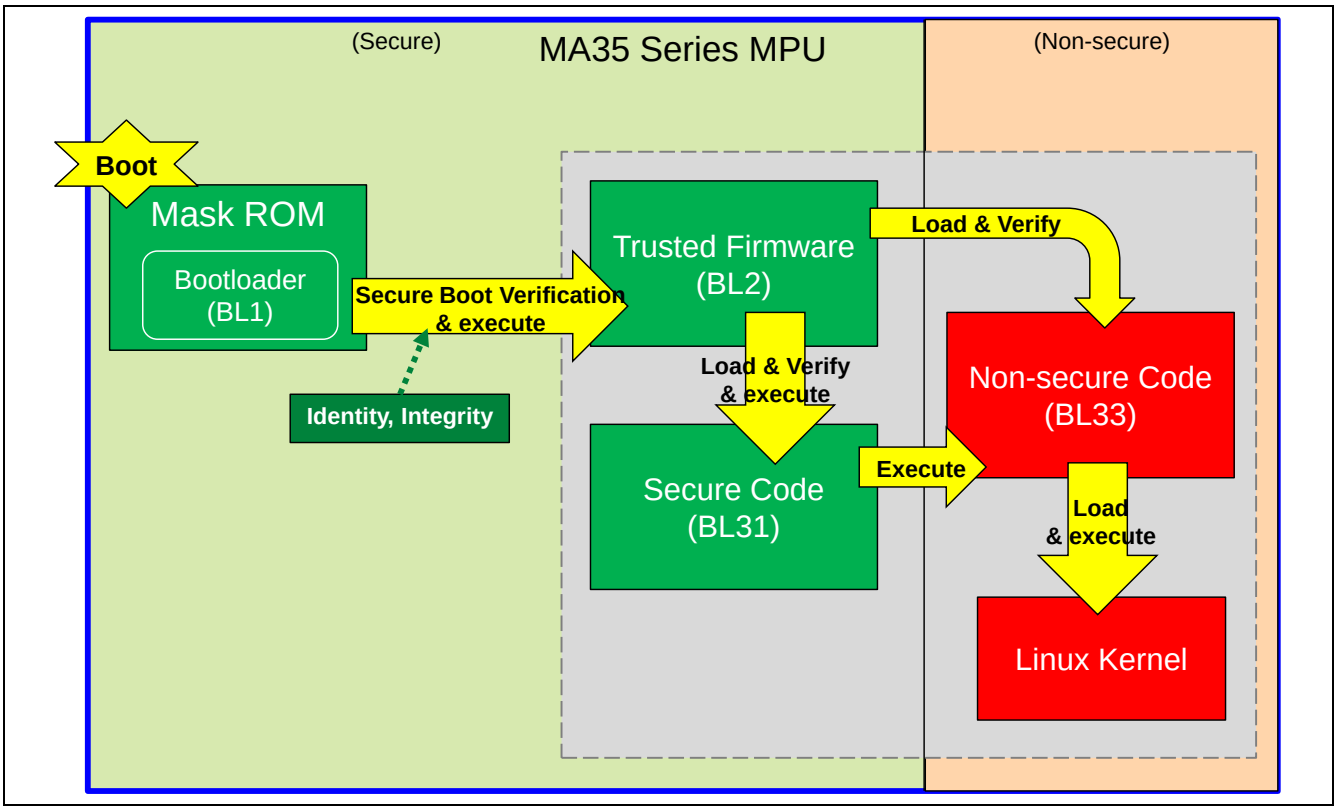


圖 1-1 MA35D1 啟動流程

BL2 載入 FIP 映像的過程可分成兩個階段來看，第一階段將存放在外部存儲裝置上的映像檔，分別載入到記憶體中的指定位址，若外部存儲裝置缺少映像檔相關資訊，則判斷啟動失敗，改由預先設定好的備份位址區域，提取備份映像進行載入。

當成功將 FIP 載入到 DDR 以及 SRAM 上後，BL2 下一階段開始對 FIP 映像做安全啟動檢驗，根據 Image ID 決定要驗證的 FIP 映像，再透過 ECDSA P-256 算法來驗證該映像(BL31、BL32、BL33)的真實性以及完整性。若驗證失敗則會回傳錯誤旗標，BL2 將重新進行第一階段工作，將備份區域的 FIP 映像載入到 DDR 以及 SRAM 中，再次進行安全啟動檢驗。示意圖可參考如圖 1-2:

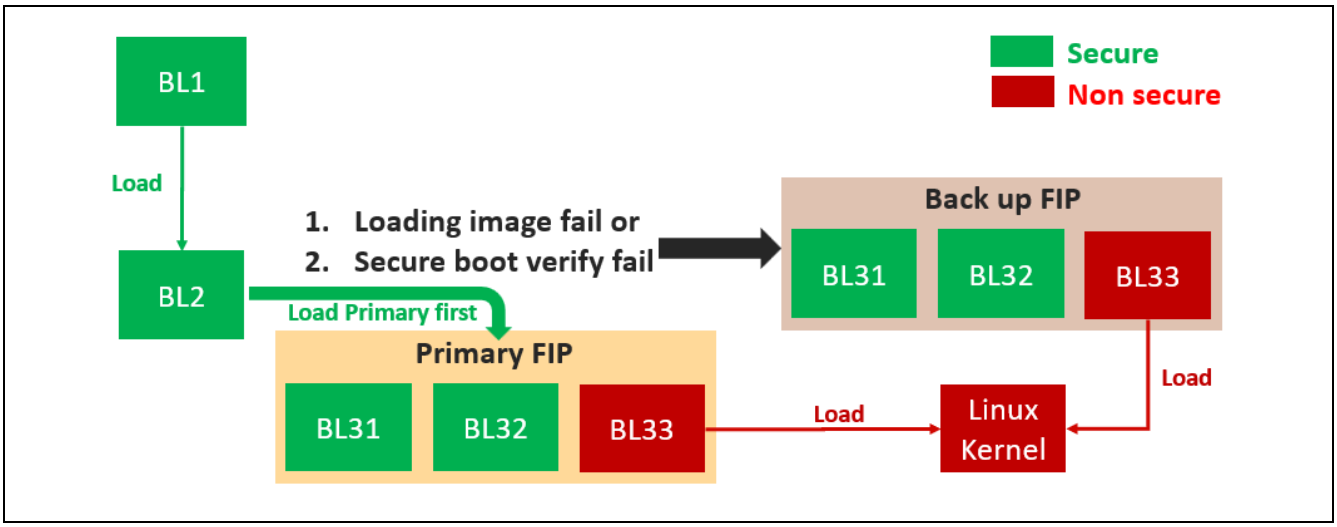


圖 1-2 FIP 映像載入流程

MA35D1 的個別映像檔，在編譯過程會根據啟動裝置的 pack.json，被 NuWriter 打包成單一映像，其中包含個別映像的燒錄資訊，透過 NuWriter 燒錄到外部存儲時會根據相關資訊，將 BL2、FIP、Kernel 以及檔案系統寫入到對應的位置。範例程式中修改了 pack.json 檔，將 FIP 備份映像一併打包進 pack.image 中，當主要 FIP 映像載入失敗時，便會將備份 FIP 載入到 DDR 上重新驗證。

以下以 SD card 作為外部存儲示例，pack-sdcard 映像架構如圖 1-3 所示。

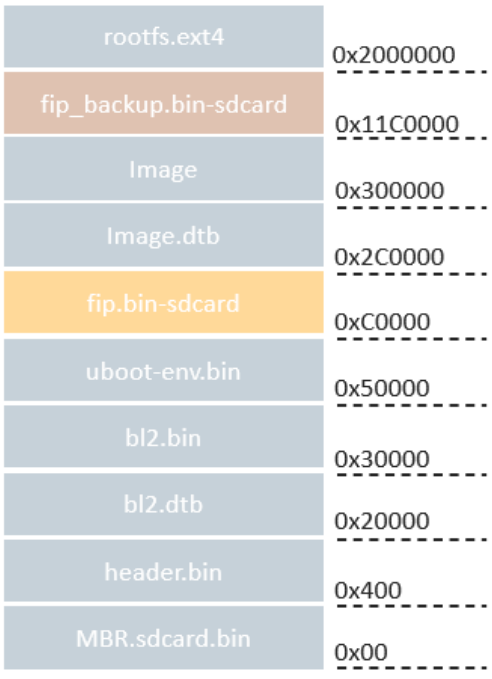


圖 1-3 pack-sdcard 架構示意圖

1.2 執行結果

本範例代碼執行結果如圖 1-4 所示。第一次執行前將主要 FIP 以外的映像檔，燒錄於外部儲存裝置上，MA35D1 啟動時，當 BL2 被成功載入後便會根據 FIP 資訊尋找映像位置，由於主要 FIP 並沒有燒錄到儲存裝置上，在顯示載入失敗的訊息後，便會嘗試從備份 FIP 映像位置讀取。

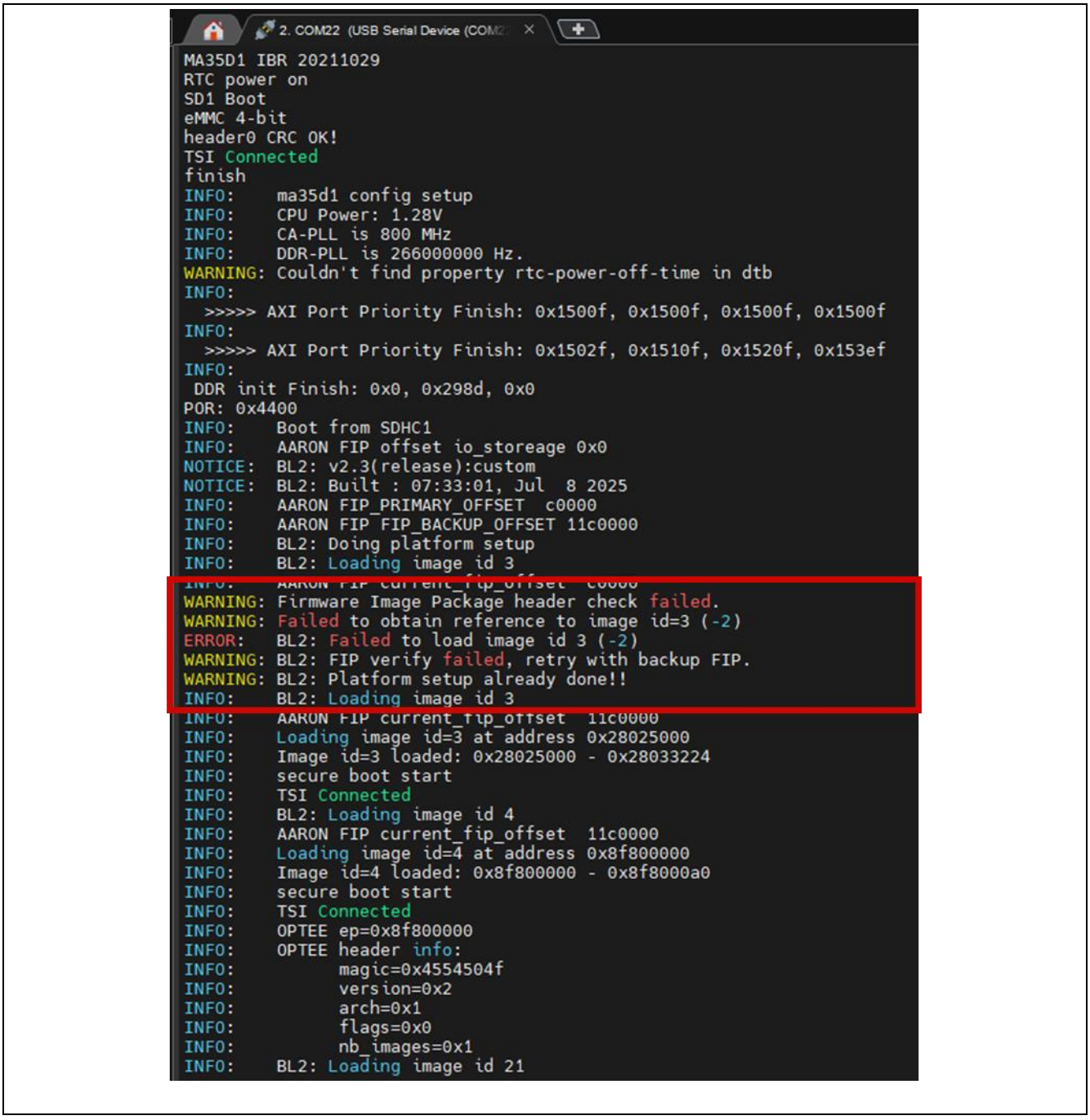


圖 1-4 範例執行結果

第二次執行測試，進行驗證簽章驗證失敗的載入動作，執行前將帶有錯誤簽章的 FIP 映像燒錄到主要位置。當 BL2 成功載入主要區域的 FIP 映像後，開始進行第二階段的簽章驗證，由於主要 FIP 的簽章是錯誤的，在驗證失敗後會嘗試從備份 FIP 映像位置讀取，並再次進行驗證。

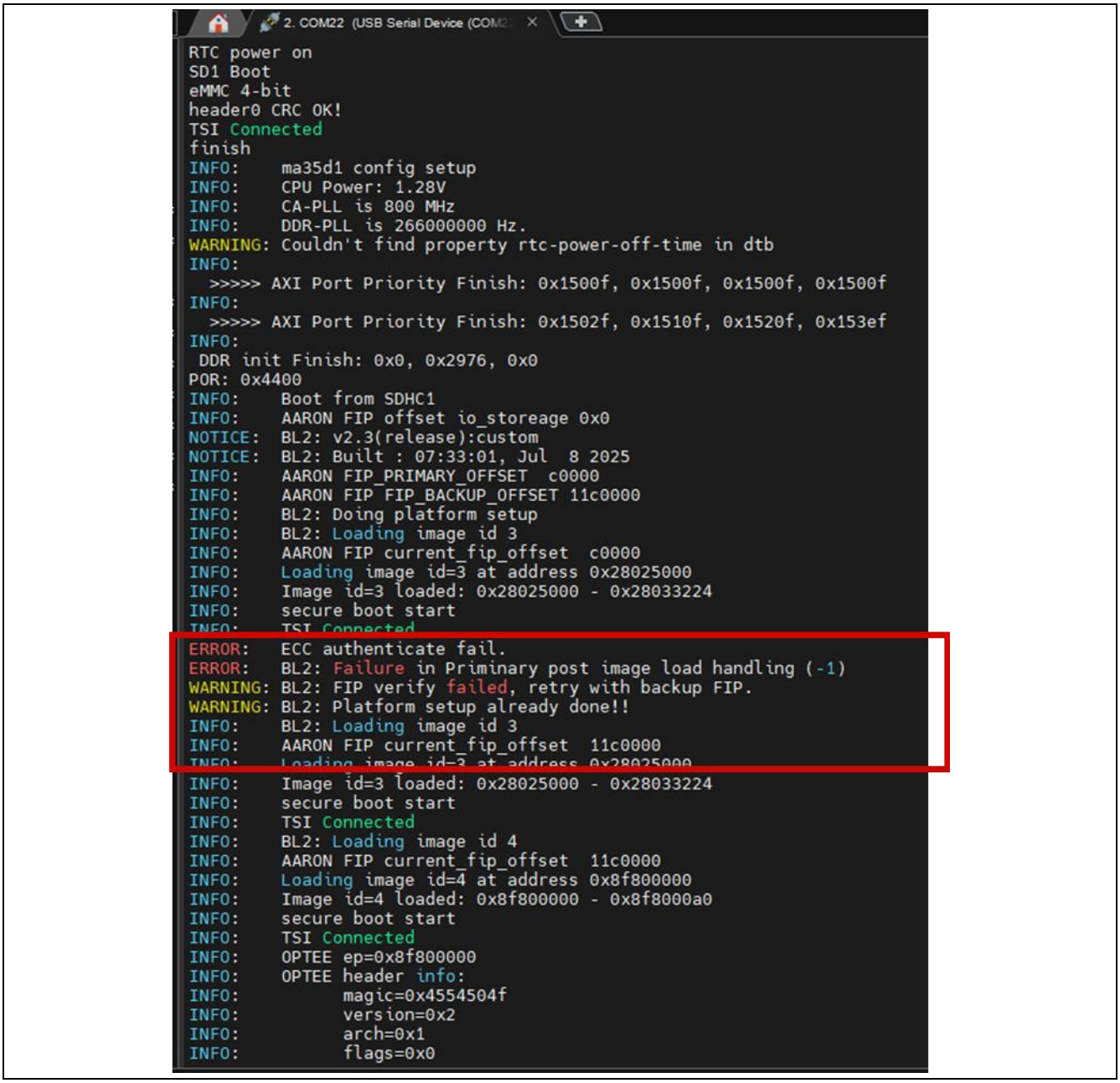


圖 1-5 範例執行結果

2. 代碼介紹

本範例主要透過修改 TF-A 中的 BL2 來實現備份功能，BL2 運行過程中會透過 bl2_load_images() 進行 FIP 映像的載入以及驗證。使用該範例前，請先確認系統可以成功運行 MA35D1 安全啟動功能。

範例拆成兩部分來理解，第一段:

```
int plat_get_image_source(unsigned int image_id, uintptr_t *dev_handle, uintptr_t
*image_spec)
```

透過呼叫該函式來取得下列兩部分資訊:

- **dev_handle**：儲存裝置的存取介面，例如 SD card、NAND flash、SPI NAND flash 等。
- **image_spec** ：指定映像在該裝置上的位置資訊，通常包含 offset、大小、格式等。

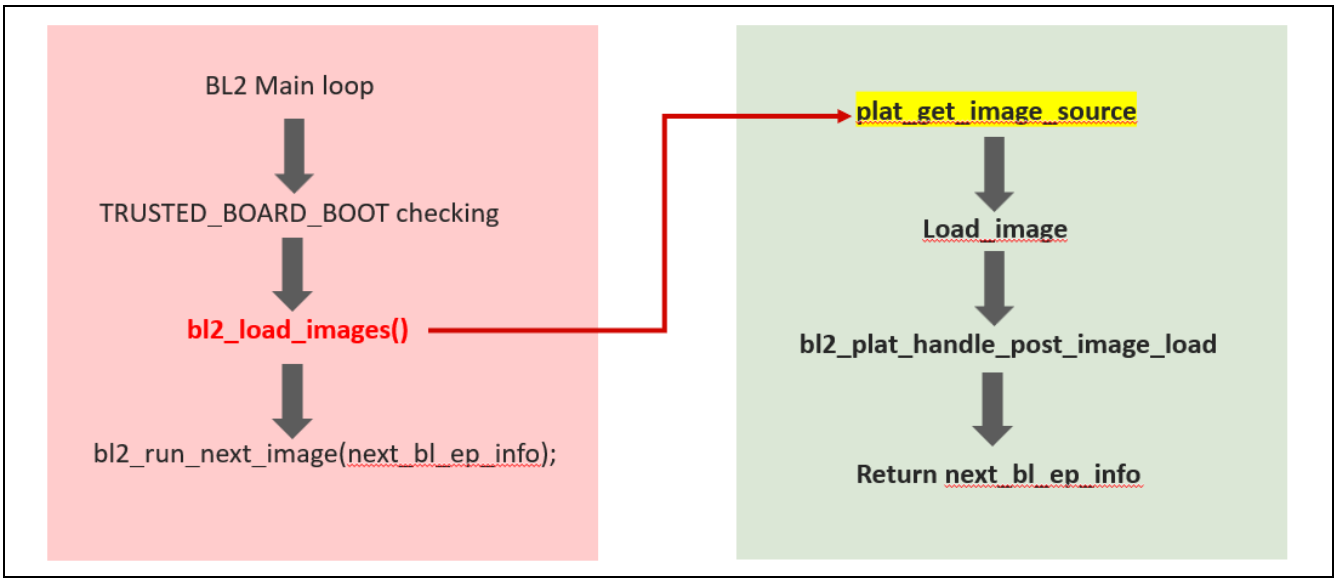


圖 2-1 BL2 運作流程-映像載入

第二階段，取得 dev_handle 與 image_spec 之後，Load_image 會呼叫：

- io_open(dev_handle, image_spec, &image_handle)：開啟裝置
- io_read(image_handle, image_base, image_size, &bytes_read)：從對應 offset 讀取內容進入 DDR

接下來進入 bl2_plat_handle_post_image_load，MA35D1 在該函式內部實現第二階段的安全啟動，透過 ECDSA P-256 來驗證 BL31、BL32 以及 BL33 映像的真實性以及完整性。

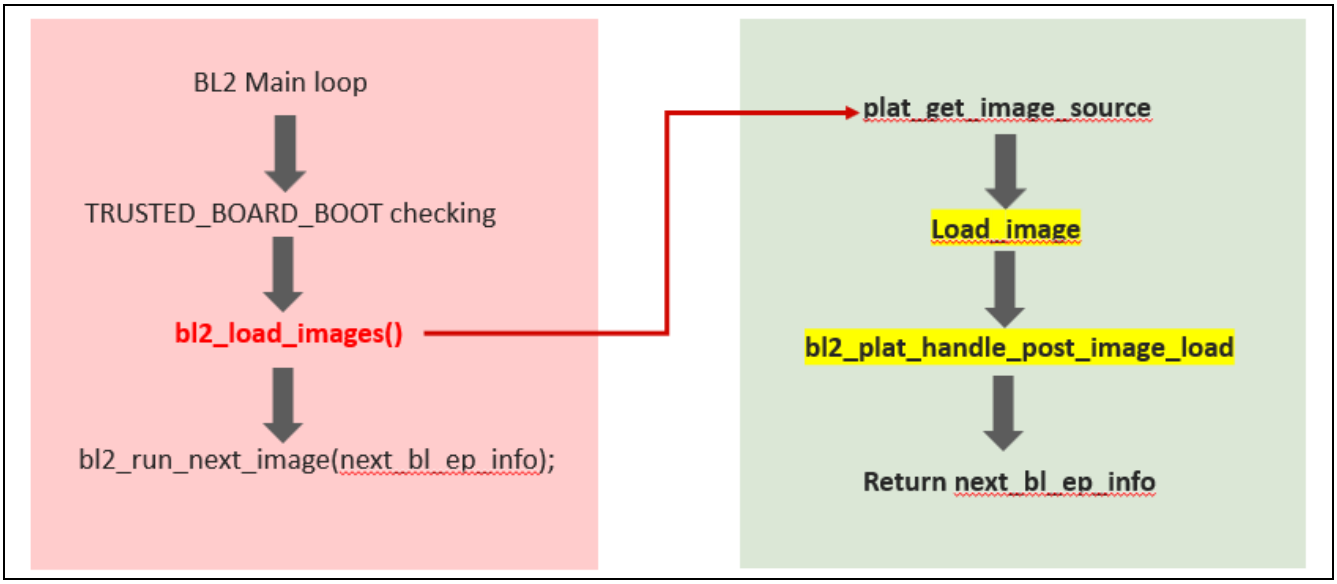


圖 2-2 BL2 運作流程-映像檢驗

BL2 中的主要架構 bl2_load_images，當映像載入(load_auth_image) 或者映像安全檢驗(bl2_plat_handle_post_image_load) 任一項失敗，FIP 映像載入位址被重新指向到備份區域，重新進行操作。當備份映像也啟動失敗時，打印出相關訊息。

```
/*
 * Copyright (c) 2016-2020, ARM Limited and Contributors. All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <assert.h>
#include <stdint.h>

#include <platform_def.h>

#include <arch.h>
#include <arch_helpers.h>
#include <common/bl_common.h>
#include <common/debug.h>
#include <common/desc_image_load.h>
#include <drivers/auth/auth_mod.h>
#include <plat/common/platform.h>

#include "bl2_private.h"

/*******************************************************************************
 * This function loads SCP_BL2/BL3x images and returns the ep_info for
 * the next executable image.
```

```
 ******************************************************************************/

 extern uint32_t FIP_PRIMARY_OFFSET;
 extern uint32_t FIP_BACKUP_OFFSET;
 extern uint32_t FIP_PRIMARY_OFFSET_SPI;
 extern uint32_t FIP_BACKUP_OFFSET_SPI;

 uint32_t current_fip_offset = 0x00;


struct entry_point_info *bl2_load_images(void)
{
    bl_params_t *bl2_to_next_bl_params;
    bl_load_info_t *bl2_load_info;
    const bl_load_info_node_t *bl2_node_info;
    int plat_setup_done = 0;
    int err;
    bool retried = false;

    INFO(" FIP_PRIMARY_OFFSET  %x\n", FIP_PRIMARY_OFFSET);
    INFO(" FIP FIP_BACKUP_OFFSET %x\n", FIP_BACKUP_OFFSET);
    /*
     * Current FIP offset setting.
     */
    current_fip_offset = FIP_PRIMARY_OFFSET;
    /*
     * Get information about the images to load.
     */
    bl2_load_info = plat_get_bl_image_load_info();
    assert(bl2_load_info != NULL);
    assert(bl2_load_info->head != NULL);
    assert(bl2_load_info->h.type == PARAM_BL_LOAD_INFO);
    assert(bl2_load_info->h.version >= VERSION_2);

    retry_fip:
    bl2_node_info = bl2_load_info->head;

    while (bl2_node_info != NULL) {
        /*
         * Perform platform setup before loading the image,
         * if indicated in the image attributes AND if NOT
         * already done before.
         */
        if ((bl2_node_info->image_info->h.attr &
             IMAGE_ATTRIB_PLAT_SETUP) != 0U) {
            if (plat_setup_done != 0) {
                WARN("BL2: Platform setup already done!!\n");
            } else {
                INFO("BL2: Doing platform setup\n");
                bl2_platform_setup();
                plat_setup_done = 1;
            }
        }

        err = bl2_plat_handle_pre_image_load(bl2_node_info->image_id);
        if (err != 0) {
```

```
            ERROR("BL2: Failure in pre image load handling (%i)\n", err);
            plat_error_handler(err);
        }

        if ((bl2_node_info->image_info->h.attr &
            IMAGE_ATTRIB_SKIP_LOADING) == 0U) {
            INFO("BL2: Loading image id %d\n", bl2_node_info->image_id);
            err = load_auth_image(bl2_node_info->image_id,
                bl2_node_info->image_info);
            if (err != 0) {
                ERROR("BL2: Failed to load image id %d (%i)\n",
                      bl2_node_info->image_id, err);
            goto try_backup;
            }
        } else {
            INFO("BL2: Skip loading image id %d\n", bl2_node_info->image_id);
        }

        /* Allow platform to handle image information.
         * Perform ECDSA verification and AES Image Decryption
         */
        err = bl2_plat_handle_post_image_load(bl2_node_info->image_id);
        if (err != 0) {
            ERROR("BL2: Failure in Primary post image load handling (%i)\n", err);
            goto try_backup;
        }

        /* Go to next image */
        bl2_node_info = bl2_node_info->next_load_info;
    }

    goto success;

    /*
     * Backup Image offset configuration.
     */
    try_backup:
    if (!retried){
        WARN ("BL2: Loading Primary failed, retry with backup FIP.\n");
        current_fip_offset = FIP_BACKUP_OFFSET;
        retried = true;
        goto retry_fip;

    }else{
        ERROR("BL2: Primary and backup FIP both failed\n");
        plat_error_handler(err);
    }



    success:
    /*
     * Get information to pass to the next image.
     */
    bl2_to_next_bl_params = plat_get_next_bl_params();
    assert(bl2_to_next_bl_params != NULL);
```

```
    assert(bl2_to_next_bl_params->head != NULL);
    assert(bl2_to_next_bl_params->h.type == PARAM_BL_PARAMS);
    assert(bl2_to_next_bl_params->h.version >= VERSION_2);
    assert(bl2_to_next_bl_params->head->ep_info != NULL);

    /* Populate arg0 for the next BL image if not already provided */
    if (bl2_to_next_bl_params->head->ep_info->args.arg0 == (u_register_t)0)
        bl2_to_next_bl_params->head->ep_info->args.arg0 =
                    (u_register_t)bl2_to_next_bl_params;

    /* Flush the parameters to be passed to next image */
    plat_flush_next_bl_params();

    return bl2_to_next_bl_params->head->ep_info;
}
```

第一階段，在 plat_get_image_source 內部會判斷該映像是否為 FIP 映像，若為 FIP 映像則會根據目前 current_fip_offset 來決定要讀取的位置。預設為主要 FIP 的儲存位置，若載入或者驗證失敗，會重新將該位置指到備份 FIP 地址。

```
/*
 * Return an IO device handle and specification which can be used to access
 * an image. Use this to enforce platform load policy.
 */
extern  uint32_t current_fip_offset;
int plat_get_image_source(unsigned int image_id, uintptr_t *dev_handle,
              uintptr_t *image_spec)
{
    uintptr_t init_params;

    assert(image_id < ARRAY_SIZE(policies));

    *dev_handle = *policies[image_id].dev_handle;

    /*
     * If image id is FIP, using “current_fip_offset”.
     * Others, using platform load policy.
     */
    if (image_id == FIP_IMAGE_ID){
        INFO(" FIP current_fip_offset  %x\n", current_fip_offset);
        static io_block_spec_t local_fip_spec;
        local_fip_spec.offset = current_fip_offset;
        local_fip_spec.length = MA35D1_FIP_SIZE;
        *image_spec = (uintptr_t)&local_fip_spec;
        init_params = policies[image_id].init_params;
    }else{
        *image_spec = policies[image_id].image_spec;
        init_params = policies[image_id].init_params;
    }
    return io_dev_init(*dev_handle, init_params);
}
```

根據設定的外部存儲介面，在初始化階段讀取設備樹文件(ma35d1.dtsi)所設定的存放位置 。

```
static void ma35d1_sdhc_setup(struct mmc *mmc)
{
    int node;

    /* get device tree information */
    if (fdt_check_header(fdt) < 0) {
        WARN("device tree header check error.\n");
    }

    if (mmc->base == SDH0_BASE) {
        node = fdt_node_offset_by_compatible(fdt, -1,
                "snps,dwcmshc-sdhci0");
        if (node < 0) {
            WARN("The compatible property `snps,dwcmshc-sdhci0` not found\n");
        }
    } else {
        node = fdt_node_offset_by_compatible(fdt, -1,
                "snps,dwcmshc-sdhci1");
        if (node < 0) {
            WARN("The compatible property `snps,dwcmshc-sdhci1` not found\n");
        }
    }

    FIP_PRIMARY_OFFSET= fdt_read_uint32_default(fdt, node, "mmc-image-offset", 0);
    FIP_BACKUP_OFFSET= fdt_read_uint32_default(fdt, node, "mmc-image-backup-offset", 0);

    mmc->bus_width = fdt_read_uint32_default(fdt, node, "bus-width", 1);

    ma35d1_sdhc_hw_init(mmc);
}
```

ma35d1.dtsi:

```
    sdhci0: sdhci@40180000 {
     compatible = "snps,dwcmshc-sdhci0";
     bus-width = <4>;
     /* sdhc information */
     mmc-image-offset = <0xc0000>;
     mmc-image-backup-offset = <0x11c0000>;
    };

    sdhci1: sdhci@40190000 {
     compatible = "snps,dwcmshc-sdhci1";
     bus-width = <4>;
     /* sdhc information */
     mmc-image-offset = <0xc0000>;
     mmc-image-backup-offset = <0x11c0000>;
    };
```

第二部分，又稱第二階段安全啟動檢驗，ma35d1_fip_verify 呼叫 TSI 進行 ECDSA 驗證，通過之後再呼叫 ma35d1_fip_deaes，將加密的映像做 AES 解密還原。

```
/*******************************************************************************
 * This function can be used by the platforms to update/use image
 * information for given `image_id`.
 ******************************************************************************/
int bl2_plat_handle_post_image_load(unsigned int image_id)
{
    int err = 0;
    bl_mem_params_node_t *bl_mem_params = get_bl_mem_params_node(image_id);

    bl_mem_params_node_t *pager_mem_params;
    bl_mem_params_node_t *paged_mem_params = NULL;

    assert(bl_mem_params != NULL);
#if FIP_DE_AES
    INFO("secure boot start\n");
    if(ma35d1_fip_verify(bl_mem_params->image_info.image_base,bl_mem_params-
>image_info.image_size)!=0) {
        ERROR("ECC authenticate fail.\n");
        return -1;
        //while(1);
    }
    ma35d1_fip_deaes(bl_mem_params->image_info.image_base,bl_mem_params-
>image_info.image_size);
#endif
    switch (image_id) {
    case BL32_IMAGE_ID:
        pager_mem_params = get_bl_mem_params_node(BL32_EXTRA1_IMAGE_ID);
        assert(pager_mem_params);

        paged_mem_params = get_bl_mem_params_node(BL32_EXTRA2_IMAGE_ID);
        assert(paged_mem_params);

        err = parse_optee_header(&bl_mem_params->ep_info,
                &pager_mem_params->image_info,
                &paged_mem_params->image_info);
        if (err != 0) {
            WARN("OPTEE header parse error.\n");
        }

        /*
         * When ATF loads the DTB the address of the DTB is passed in
         * arg2, if an hw config image is present use the base address
         * as DTB address an pass it as arg2
         */
        bl_mem_params->ep_info.args.arg0 = bl_mem_params->ep_info.args.arg1;
        bl_mem_params->ep_info.args.arg1 = 0;
        bl_mem_params->ep_info.args.arg2 = 0;
        bl_mem_params->ep_info.args.arg3 = 0;

        bl_mem_params->ep_info.spsr = ma35d1_get_spsr_for_bl32_entry();

        break;
```

```
    case BL33_IMAGE_ID:
        /* BL33 expects to receive the primary CPU MPID (through r0) */
        bl_mem_params->ep_info.args.arg0 = 0xffff & read_mpidr();
        bl_mem_params->ep_info.spsr = ma35d1_get_spsr_for_bl33_entry();
        break;

    case SCP_BL2_IMAGE_ID:
        /* The subsequent handling of SCP_BL2 is platform specific */
        err = plat_ma35d1_bl2_handle_scp_bl2(&bl_mem_params->image_info);
        if (err) {
            WARN("Failure in platform-specific handling of SCP_BL2 image.\n");
        }
        break;

    default:
        /* Do nothing in default case */
        break;
    }

    return err;
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - Linux-5.10.x

3.2 硬體需求

- 電路元件
  - NuMaker-HMI-MA35D1 V1.0

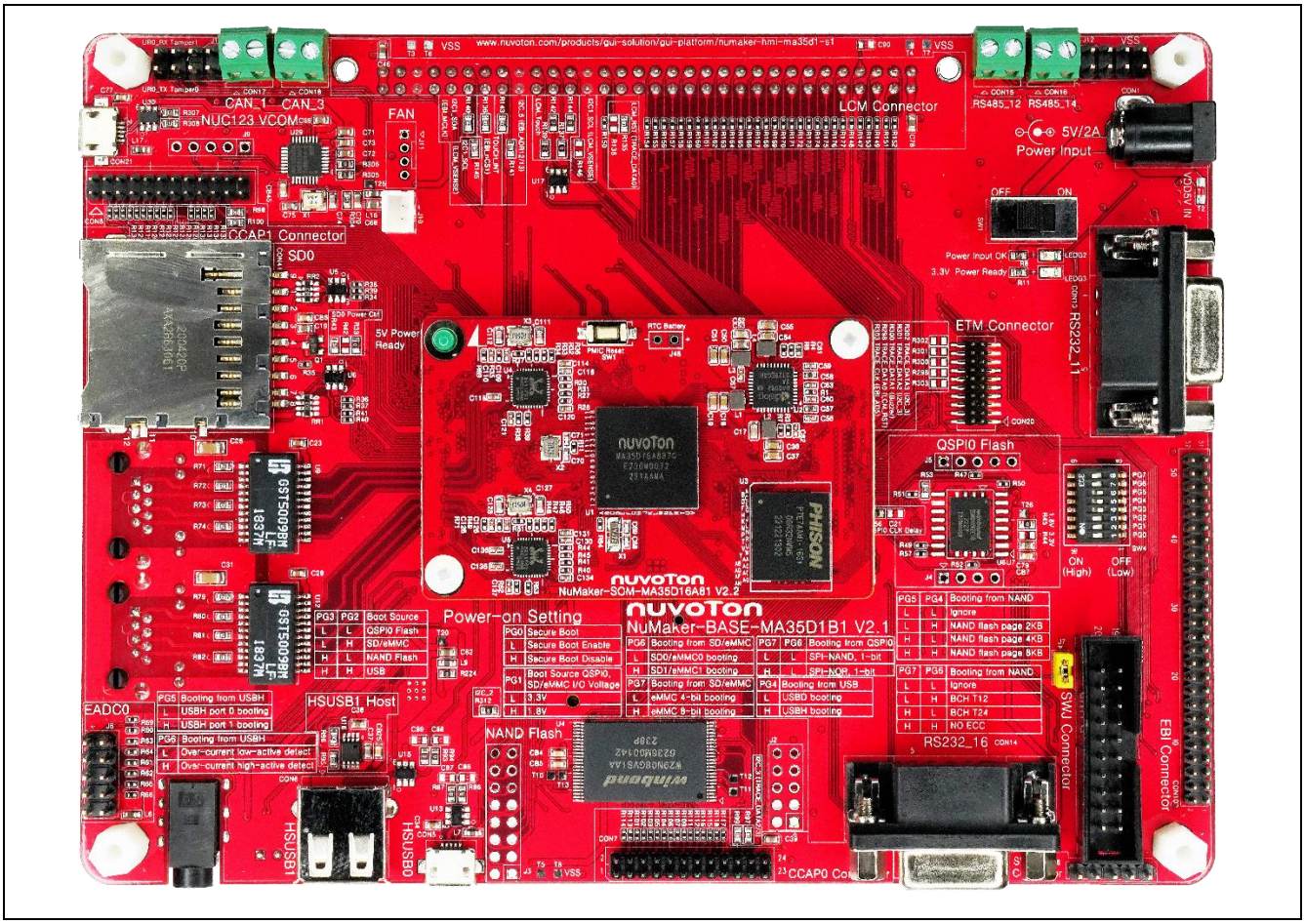


圖 3-1 NuMaker-HMI-MA35D1 V1.0

4. 目錄資訊

目錄結構如下圖所示。

EC_MA35D1_TF-A_FIP_Backup_V1.00	
arm_trusted_firmware	Reference implementation of secure world
bl2	Trusted Bootloader
common	
driver/nuvoton	
fdts	
lib/optee	
plat/nuvoton	

表 4-1 目錄資訊

5. 範例程式執行

MA35D1 預設會編譯 TF-A 映像，範例要能成功執行必須修改 arm-trusted-firmware 資料夾底下的相關 source code，以下提供 buildroot 相關步驟，外部存儲裝置為 SD card:

1. 進入 MA35D1 buildroot 相關路徑(根據個人設定會有所不同):

```
user@e9e6e7eb0c21:~/shared/buildroot_2024/output/build/arm-trusted-firmware-custom$。
```

2. 將範例程式中各別資料夾的.c /.h /.dtsi 檔複製，並取代原 arm-trusted-firmware-custom 底下的對應檔案。注意，該動作是取代，並不是直接將範例程式資料夾直接整個複製到 arm-trusted-firmware-custom 資料夾。

3. 在 menuconfig 中開啟 Secure Boot 以及設定相關金鑰:

```
Bootloaders --- >
  [*] MA35D1 Secure Boot
  (0A0BC81E5AFBF836C5E0CFBEA12C1E269A2EBC3B0B6EC39EEE1C7142F760EBC4) AES Key
  (8F1571E9975006A545DF854264F7B18934B5CB2EE619E66DBC0FBEA52C71C919) ECDSA Key
```

4. 移除 arm-trusted-firmware-custom 底下的 build 資料夾。

5. 重新編譯 arm-trusted-firmware-custom。

```
make arm-trusted-firmware-rebuild
```

6. 進入以下路徑:

```
user@e9e6e7eb0c21:~/shared/buildroot_2024/board/nuvoton/ma35d1/nuwriter $
```

7. 修改"uboot-env.bin-sdcard" offset 位址，以及增加備份 fip 映像相關設定於 pack-sdcard.json。

```
{
  "image": [
    {
      "offset": "0x0",
      "file": "MBR.sdcard.bin",
      "type": 0
    },
    {
      "offset": "0x400",
      "file": "header.bin",
      "type": 0
    },
    {
      "offset": "0x20000",
```

```
      "file": "nuwriter/bl2.dtb",
      "type": 0
    },
    {
      "offset": "0x30000",
      "file": "nuwriter/bl2.bin",
      "type": 0
    },
    {
      "offset": "0x50000",
      "file": "uboot-env.bin-sdcard",
      "type": 0
    },
    {
      "offset": "0xC0000",
      "file": "fip.bin-sdcard",
      "type": 0
    },
    {
      "offset": "0x2c0000",
      "file": "Image.dtb",
      "type": 0
    },
    {
      "offset": "0x300000",
      "file": "Image",
      "type": 0
    },
    {
      "offset": "37748736",
      "file": "rootfs.ext4",
      "type": 0
    },
    {
      "offset": "0x11C0000",
      "file": "fip.bin-sdcard",
      "type": 0
    }
  ]
}
```

8. 重新編譯映像檔。

```
make
```

9. 透過 NuWriter 燒錄 pack 映像。

10. 透過 NuWriter 清除主要 fip.bin-sdcard 所寫入的位置(0xc0000-0x1c0000, 參考 pack_sdcard.json)。

11. 編譯成功後，NuWriter 會生成一”otp_key.json”檔案於 output/images/nuwriters 底下，包含 ECDSA 公鑰以及 AES 金鑰，透過 NuWriter 燒錄於 MA35D1 IBR Key 區域。

12. 將開發板上的 Power-on setting 指撥開關設定為 boot from SD0，啟動 MA35D1 開發板。

13. 進入 BL2 ，開始進行各 BL3x 映像的載入過程，由於主要 FIP 映像在步驟 10 時已被清除，BL2 無法正確獲取相關資訊顯示啟動失敗，改由備份位址獲取 FIP 映像，並完成啟動。

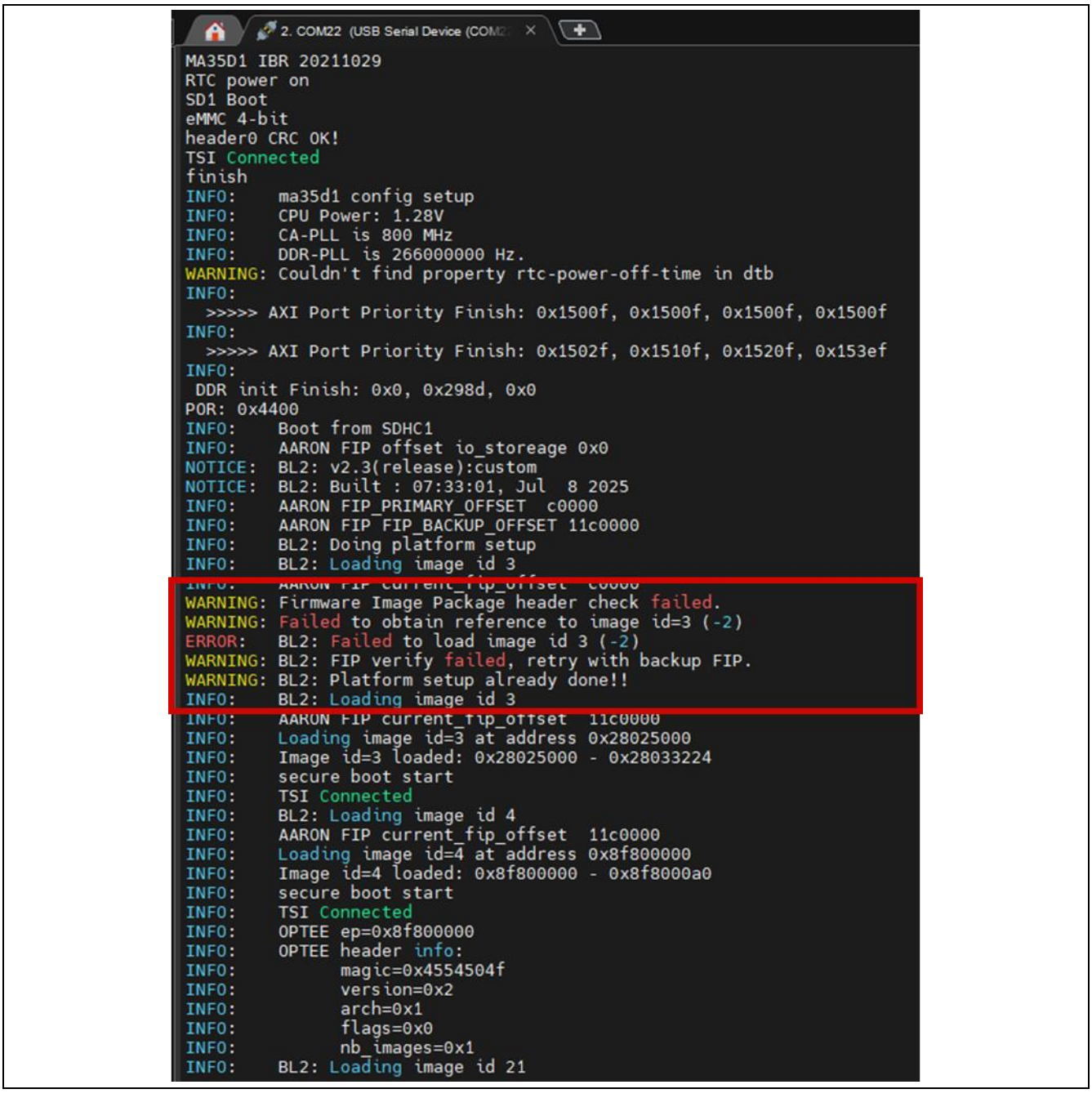


圖 5-1 範例執行結果-主映像載入失敗

14. 第二項測試，驗證錯誤私鑰加簽的 FIP 映像，首先清除 arm-trusted-firmware。

```
make arm-trusted-firmwared-dirclean
```

15. 重新設定 ECDSA 私鑰。

```
Bootloaders --- >
  [*] MA35D1 Secure Boot
  (0A0BC81E5AFBF836C5E0CFBEA12C1E269A2EBC3B0B6EC39EEE1C7142F760EBC4) AES Key
  (8F1571E9975006A545DF854264F7B18934B5CB2EE619E66DBC0FBEA52C71C918) ECDSA Key
```

16. 重新透過 make 編譯新的映像。

17. 再次透過 NuWriter 將錯誤私鑰簽署過的 fip.bin-sdcard ，燒寫到主要 fip 映像存放位址 (0xc0000, 參考 pack_sdcard.json)。

18. 再次啟動 MA35D1 開發板，當 BL2 成功載入主要區域的 FIP 映像後，開始進行第二階段的簽章驗證，由於主要 FIP 的簽章是錯誤的，在驗證失敗後會嘗試從備份 FIP 映像位置讀取，並再次進行驗證，驗證成功後繼續往下執行完成系統啟動。

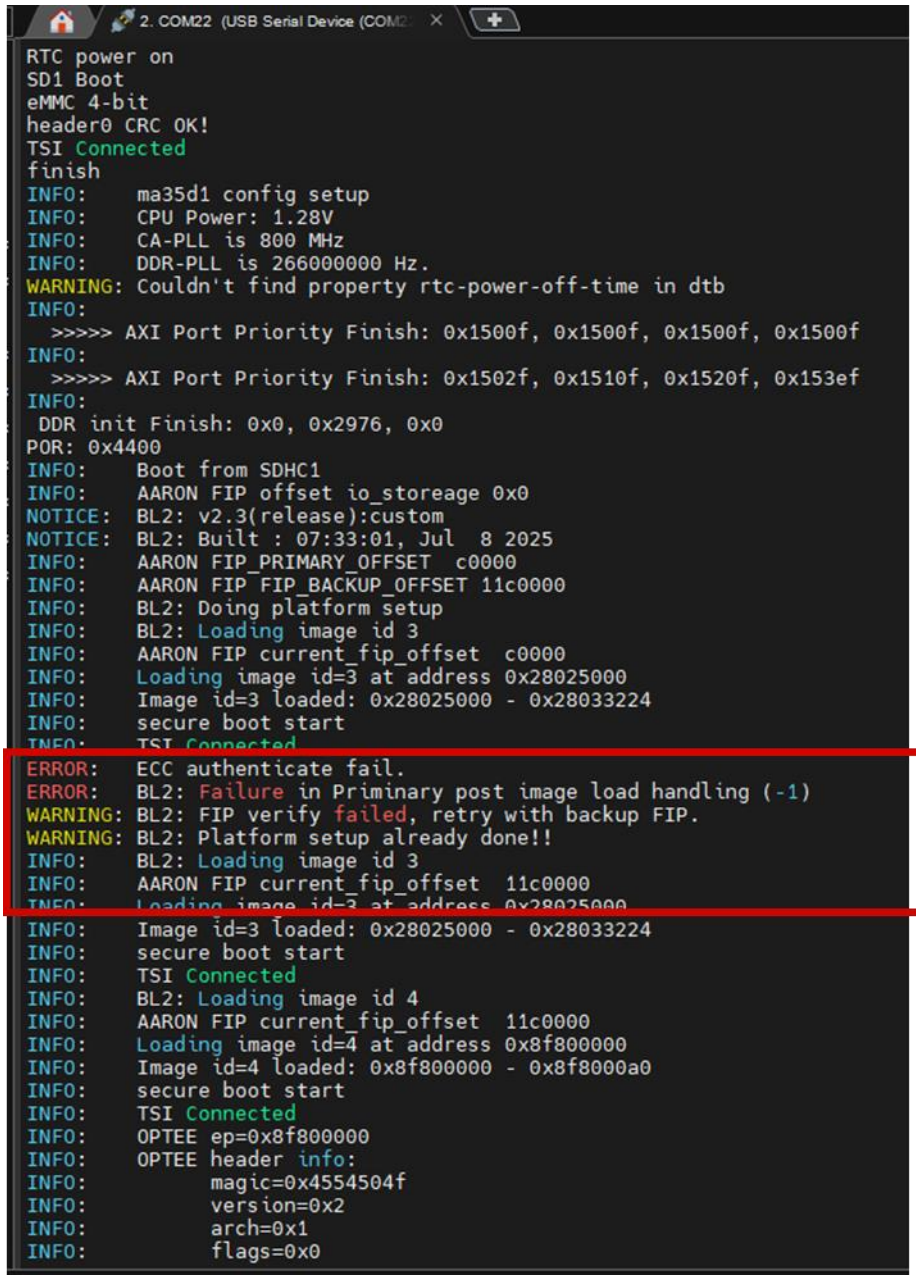


圖 5-2 範例執行結果- 主映像簽章驗證失敗

6. 修訂紀錄

Date	Revision	Description
2025.10.07	1.00	初始發布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.