MA35D1 使用 OP-TEE 進行 TSI 安全監控

NuMicro® 64 位元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 OP-TEE 實現使用 MA35D1 的協同處理器 TSI 做記憶體區塊的安全監控功能
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

本範例代碼使用 MA35D1 內的 TSI 進行特定記憶體區段的完整性檢測。

TSI 全名為 Trusted Secure Island，為 MA35D1 SoC 的一部分，旨在提供安全的加密和密鑰儲存等服務。在 MA35D1 中，TSI 是一個獨立的子系統，並且與 Cortex®-A35 主系統各自獨立運行。MA35D1 Cortex-A35 上運行的程序訪問 TSI 的唯一途徑是通過 Wormhole1，這是一個安全屬性的硬體元件，提供 Cortex-A35 主系統與 TSI 子系統之間雙向通信。TSI 以被動方式運作，始終等待來自 Cortex-A35 的請求。它會在一個無限迴圈中運行，接收請求、執行請求，並通過 Wormhole1 回應 ACK。為了節省電力，如果在一段時間內未接收到請求，TSI 本身會自動進入省電模式。

OP-TEE (Open Portable Trusted Execution Environment) 是一個開源的 Trusted Execution Environment (TEE, https://optee.readthedocs.io/en/latest/general/about.html)，由 Linaro 主導開發，專為 ARM TrustZone® 技術設計。其目的是在嵌入式系統中提供一個安全的執行環境，用以執行敏感的應用程序和處理敏感的數據。有如下的應用場景:

- 數字簽名與驗證：OP-TEE 可用於保護密鑰操作，確保數字簽名的私鑰只在安全環境中使用，防止被洩露。
- 數據加密與解密：通過 TEE 來處理加密和解密操作，保障敏感數據的安全性。
- 身份驗證：使用 TEE 來存儲和驗證身份憑證，防止憑證被竊取或篡改。
- 支付系統：TEE 可以保護金融交易的敏感數據，防止交易過程中的安全漏洞。

本範例程式根據 OP-TEE 建立一支在 Linux 端的應用程序 (Client Application)，透過該應用程序向安全世界 (OP-TEE OS) 中的安全程序 (Trusted Application) 發送請求，再由 TA 呼叫 TSI 進行記憶體的完整性檢驗。TSI 可存取 MA35D1 的所有記憶體空間，透過硬體加密加速器的 SHA-256，定期計算要監控記憶體區的雜湊值，來確保該記憶體空間沒有被竄改。區段的起始位置以及大小可由用戶自行設定，最多可以監測四段區域。由於 TSI 是一個獨立的子系統，因此可以確保其計算的獨立性及可靠性。

1.1 原理

Cortex-A35 只能通過 TrustZone Secure (TZS) 模式下訪問 TSI。由於 Linux 屬於運行在非安全世界的 OS，無權訪問 Wormhole1 和 TSI，必須有一個在安全模式下運行的操作系統，例如 OP-TEE，來幫助 Linux 訪問 TSI。MA35D1 的 Linux BSP 支援 OP-TEE 的 Crypto、Key Store 和 TRNG 客戶端驅動程式，以協助 Linux 訪問安全世界的 TSI。示意圖可參考如圖 1-1:

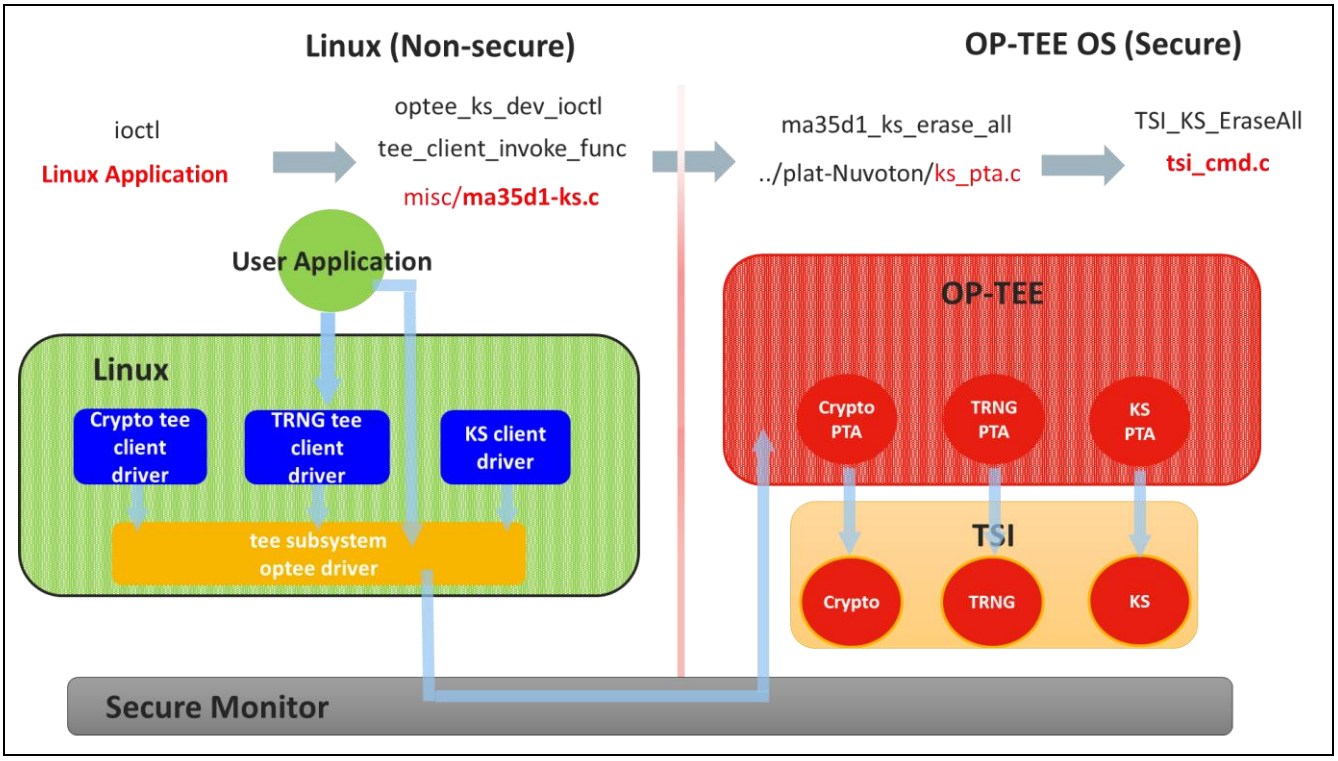


圖 1-1 Linux 與 OP-TEE 溝通示意圖

MA35D1 的 Wormhole1 是 TrustZone Secure (TZS)，只有在安全模式下運行的程式才能訪問 Wormhole1。A35 透過 Secure Monitor 切換安全屬性 (NS->S)，進到安全模式下運行 OP-TEE OS，而 Linux 核心則在非安全模式下運行。MA35D1 Linux 核心中的 Crypto、TRNG 和 Key Store 驅動程式作為 OP-TEE 應用端驅動實現，需要 OP-TEE 幫助訪問 TSI。TSI 從 Wormhole1 接收請求，執行該請求，並通過 ACK 回應執行結果，如圖 1-2 所示。

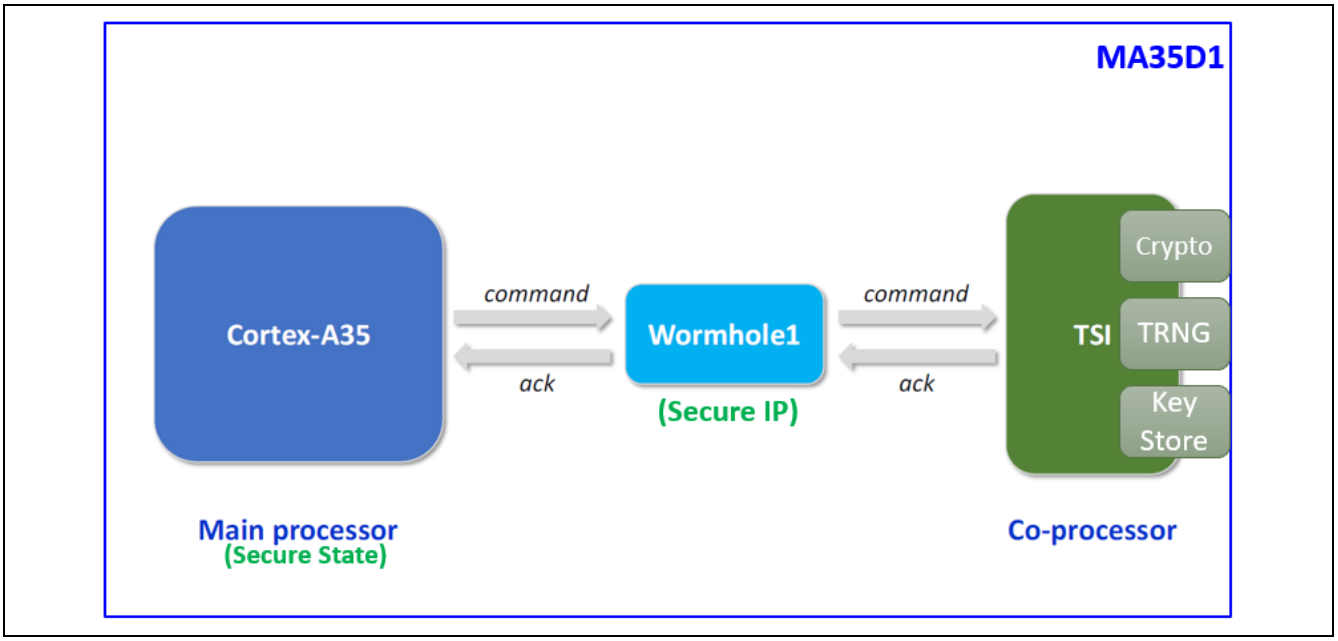


圖 1-2 A35 與 TSI 通訊示意圖

TSI 支援 TSI_Monitor_Set、TSI_Monitor_Status 和 TSI_Monitor_Close 命令，這些命令允許 TSI 監控指定記憶區域的完整性。TSI 最多同時支援四個監控器運行。

MA35D1 使用 TSI_Monitor_Set 命令來指定要監控的記憶區域以及執行記憶檢查的時間間隔。TSI 通過 SHA 硬體加速器檢查並比較記憶區域。MA35D1 可以隨時使用 TSI_Monitor_Status 來檢查記憶體完整性狀態。監控器會定期檢查記憶體，直到接收到 TSI_Monitor_Close 命令為止。

- TSI_Monitor_Set

  設定要監控的記憶體起始位置、大小(最大:32 MB) 以及時間間格。

- TSI_Monitor_Status

  請求 TSI 回報該監控段的狀態。

- TSI_Monitor_Close

  關閉該監控段的定期檢驗。

1.2 執行結果

本範例代碼執行結果如圖 1-3 所示。MA35D1 啟動成功後，進到 usr/bin 下可以看到 OP-TEE 相關範例，除 optee_example_TSI_Monitor 外皆為 OP-TEE 預設提供。執行

optee_example_TSI_Monitor，第一次帶入 TSI_Monitor_Set 相關參數 (optee_example_TSI_Monitor *cmd set, time interval, base address, size*) ，設定成功後會看到 TSI 回傳 ID (0x35) ，當執行 TSI_Monitor_Status 以及 TSI_Monitor_Close 時，該 ID 值須作為參數帶入已成功執行範例。

第二次執行 optee_example_TSI_Monitor 範例程式，帶入 TSI_Monitor_Status 相關參數 (optee_example_TSI_Monitor *cmd status, ID*)，執行成功後會看到 TSI 回傳該監控區域的狀態 (0x14300035)，其中 143 為 0x0143，表示輸入的命令為 TSI_Monitor_Status ，00 為 0X00 表示無異常狀態，35 則為該監控區域對應的表示 ID 0x35。

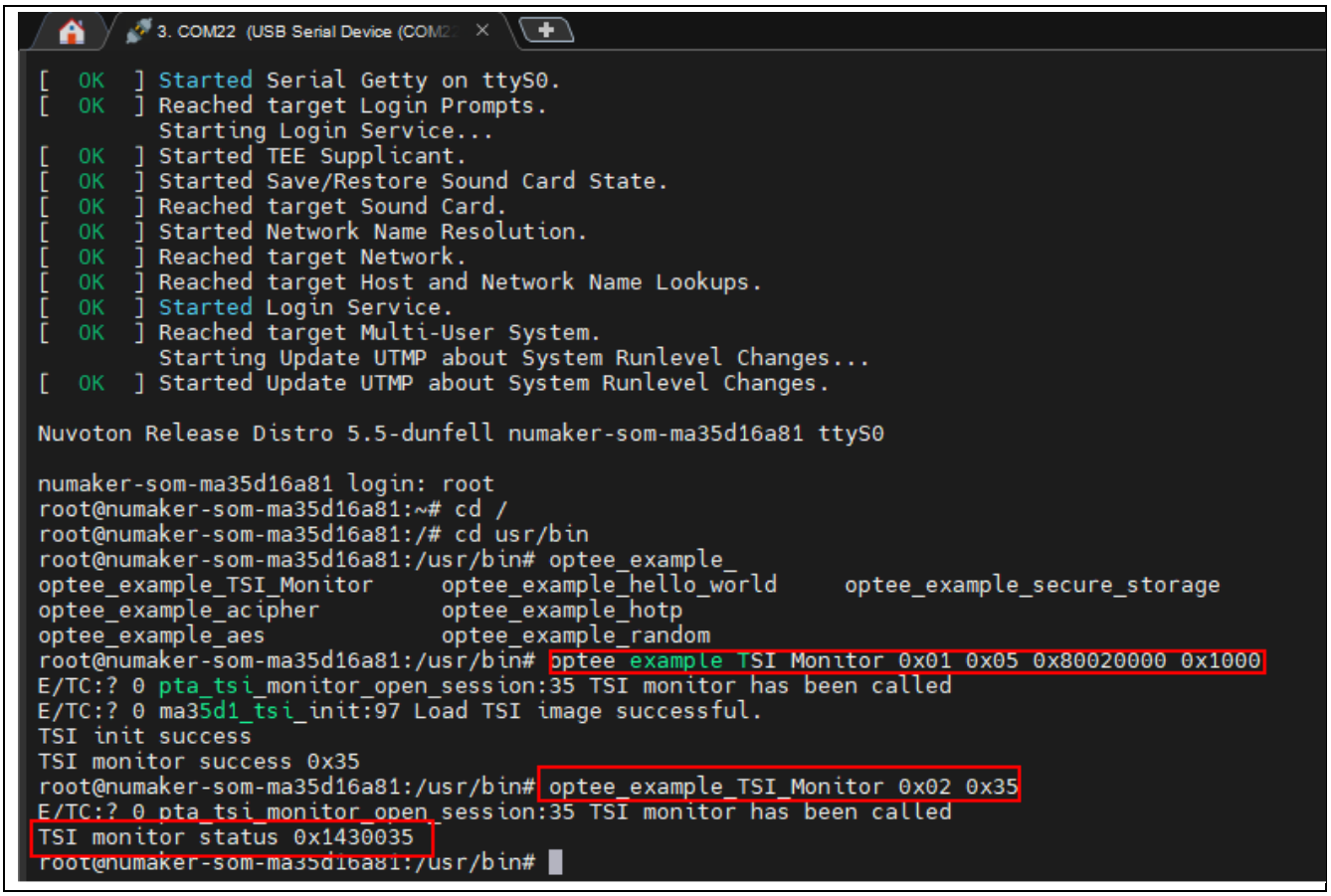


圖 1-3 範例執行結果

2. 代碼介紹

TSI Monitor 範例拆成兩部分:

- optee_example

運行於 Linux 端的應用程序，又稱作 Client Application (CA)，CA 透過 OP-TEE driver 提供的接口 API，實現相關的安全操作，CA 中的每個應用命令在 OP-TEE OS 端都有對應的操作，實際的操作工作都是在 TEE OS 或者應用層 Trusted Application (TA) 完成，再返回結果通知 CA 。

- optee_os_ma35d1

架構同樣分成 OS 層以及應用層，OS 處理來自 Linux 的相關安全功能調用，應用層為實際應用 Trusted Application，主要實現如加解密，安全存儲以及與底層的安全屬性硬體進行互動。

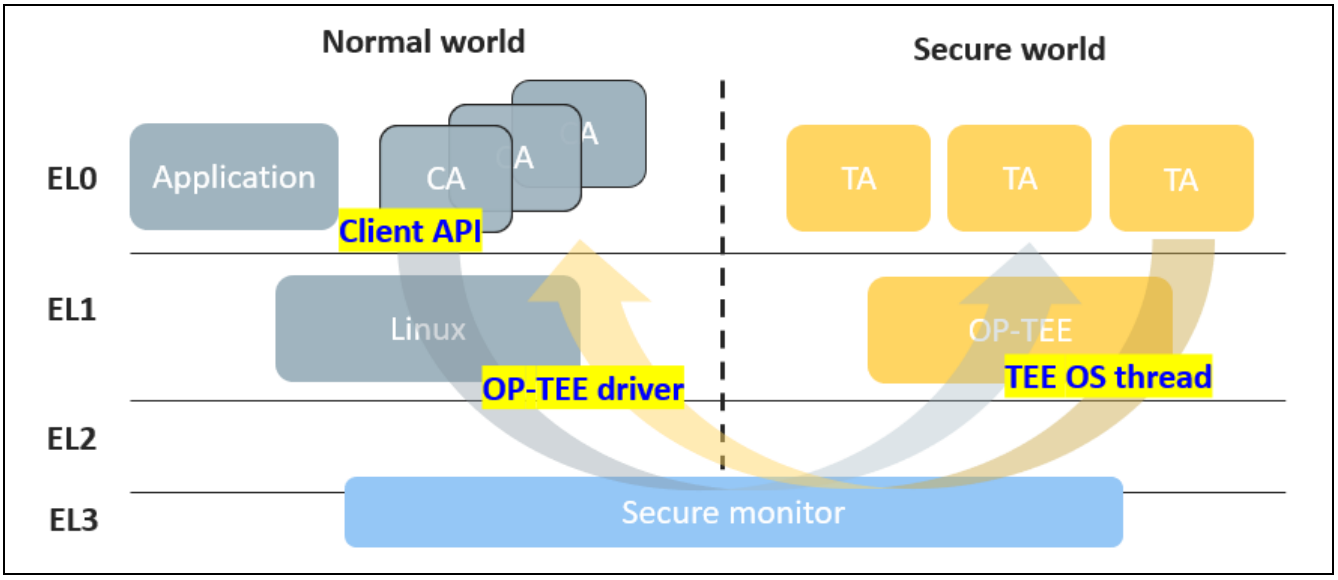


圖 2-1 OP-TEE 軟體架構

第一部份為 optee_example。程式碼一開始呼叫 Prepare_tee_session 確認是否成功建立 TSI monitor 會話，再透過 sscanf 讀取應用參數，來確認實際要執行的操作。代碼位於 *main.c*。

```
int main(int argc, char *argv[])
{
    struct test_ctx ctx;
    uint32_t monitor_argv1,monitor_argv2,monitor_argv3,monitor_argv4;

    /* Open tee application */
    Prepare_tee_session(&ctx);
```

```
    /*
     * argv[1]: TSI monitor command
     * argv[2]:
     * For TSI monitor set, it represents monitor time interval
     *     unit: 1 second, Maximum is 255
     * For TSI monitor status/ TSI monitor close, it represents monitor ID
     * argv[3]: Base Address
     * argv[4]: Monitor size, Maximum is 32 MB (0x2000000)
     */

    sscanf (argv[1], "%x", &monitor_argv1);

    /*
     * Check tsi operation:
     * Monitor_Set
     * Monitor_Status
     * Monitor_Close
     */
    switch (monitor_argv1)
    {
    case monitor_set:
         sscanf (argv[2], "%x", &monitor_argv2);
         sscanf (argv[3], "%x", &monitor_argv3);
         sscanf (argv[4], "%x", &monitor_argv4);
         Monitor_Set(&ctx, monitor_argv2, monitor_argv3, monitor_argv4);
         break;

    case monitor_status:
         sscanf (argv[2], "%x", &monitor_argv2);
         Monitor_Status(&ctx, monitor_argv2);
         break;

    case monitor_close:
         sscanf (argv[2], "%x", &monitor_argv2);
         Monitor_Close(&ctx, monitor_argv2);
         break;

    default:
         break;
    }

    /*Close tee application*/
    Terminate_tee_session(&ctx);

    return 0;
}
```

相關操作 API 如下，提供監控設定(Monitor_Set)，狀態回報(Monitor_Status)以及監控關閉(Monitor_Close)等操作。

```
void Monitor_Set(struct test_ctx *ctx, uint32_t interval, uint32_t baseaddress, uint32_t
size)
{
    TEEC_Operation op;
    TEEC_Result res;
```

```
    uint32_t err_origin;
    /* Clear the TEEC_Operation structure */
    memset(&op, 0, sizeof(op));

    /*
     * Invokes a Command within the specified Session.
     * Command: TSI initial
     */
    res = TEEC_InvokeCommand(&ctx->sess, TA_TSI_INIT, &op,
              &err_origin);
    if (res != TEEC_SUCCESS)
         printf("TSI init fail\n");
    else
         printf("TSI init success\n");


    /* Set TSI monitor parameter */
    memset(&op, 0, sizeof(op));
    op.paramTypes = TEEC_PARAM_TYPES(TEEC_VALUE_INPUT, TEEC_VALUE_OUTPUT,
TEEC_VALUE_INPUT, TEEC_NONE);
    op.params[0].value.a = interval;
    op.params[0].value.b = baseaddress;
    op.params[2].value.a = size;

    /*
     * Invokes a Command within the specified Session.
     * Command: TA_TSI_MONITOR_SET
     */
     res = TEEC_InvokeCommand(&ctx->sess, TA_TSI_MONITOR_SET, &op,
     &err_origin);
     if (res != TEEC_SUCCESS)
            errx(1, "TEEC_InvokeCommand failed with code 0x%x origin 0x%x",
            res, err_origin);
      else
         printf("TSI monitor success%x\n", op.params[1].value.a );

}

void Monitor_Status(struct test_ctx *ctx, uint32_t id)
{
    TEEC_Operation op;
    TEEC_Result res;
    uint32_t err_origin;
    /* Clear the TEEC_Operation structure */
    memset(&op, 0, sizeof(op));
    /* Set TSI monitor parameter */
    op.paramTypes = TEEC_PARAM_TYPES(TEEC_VALUE_INPUT, TEEC_VALUE_OUTPUT,
                     TEEC_NONE, TEEC_NONE);
    op.params[0].value.a = id;            /* Monitor ID */
    /*
     * Invokes a Command within the specified Session.
     * Command: TA_TSI_MONITOR_STATUS
     */
     res = TEEC_InvokeCommand(&ctx->sess, TA_TSI_MONITOR_STATUS, &op,
     &err_origin);
         printf("TSI monitor status%x\n",op.params[1].value.a);
```

```
}

void Monitor_Close(struct test_ctx *ctx, uint32_t id)
{
    TEEC_Operation op;
    TEEC_Result res;
    uint32_t err_origin;
    /* Close TSI monitor 1 status */
    memset(&op, 0, sizeof(op));
    op.paramTypes = TEEC_PARAM_TYPES(TEEC_VALUE_INPUT, TEEC_VALUE_OUTPUT,
                    TEEC_NONE, TEEC_NONE);
    op.params[0].value.a = id;            /* Monitor ID */
    /*
     * Invokes a Command within the specified Session.
     * Command: TA_TSI_MONITOR_CLOSE
     */
     res = TEEC_InvokeCommand(&sess, TA_TSI_MONITOR_CLOSE, &op,
     &err_origin);
     if (res != TEEC_SUCCESS)
            errx(1, "TEEC_InvokeCommand failed with code 0x%x origin 0x%x",
            res, err_origin);
      else
         printf("TSI Close status%x\n",op.params[1].value.a);
}
```

CA 的操作在 optee-os-ma35d1 底下的 TSI_Monitor_pta.c 都有對應的功能，透過 pta 呼叫 TSI 進行一系列的操作，並返回結果給 CA。

```
// SPDX-License-Identifier: GPL-2.0
/*
 * Copyright (C) 2020 Nuvoton Technology Corp. All rights reserved.
 *
 */

#include <kernel/pseudo_ta.h>
#include <mm/core_memprot.h>
#include <platform_config.h>
#include <io.h>
#include <platform_config.h>
#include <tsi_cmd.h>
#include <TSI_Monitor_ta.h>

#define PTA_NAME "tsi_monitor.pta"

/*
 * Open TSI monitor session
 */

TEE_Result pta_tsi_monitor_open_session(uint32_t param_types,
     TEE_Param __maybe_unused params[4],
     void __maybe_unused **sess_ctx)
{
    uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_NONE,
```

```
                        TEE_PARAM_TYPE_NONE,
                        TEE_PARAM_TYPE_NONE,
                        TEE_PARAM_TYPE_NONE);

    EMSG("TSI monitor has been called");

    if (param_types != exp_param_types)
         return TEE_ERROR_BAD_PARAMETERS;

    /* Unused parameters */
    (void)&params;
    (void)&sess_ctx;

    /* If return value != TEE_SUCCESS the session will not be created. */
    return TEE_SUCCESS;
}

/*
 * Close TSI monitor session
 */
void pta_tsi_monitor_close_session(void __maybe_unused *sess_ctx)
{
    (void)&sess_ctx; /* Unused parameter */
    IMSG("Goodbye!\n");
}

/*
 * TSI initialization
 */

static TEE_Result tsi_init(void)
{
    vaddr_t sys_base = core_mmu_get_va(SYS_BASE, MEM_AREA_IO_SEC);
    vaddr_t tsi_base = core_mmu_get_va(TSI_BASE, MEM_AREA_IO_SEC);
     if (!(io_read32(sys_base + SYS_CHIPCFG) & TSIEN))
         return ma35d1_tsi_init();
}

/*
 * Send monitor command to TSI
 */

static TEE_Result tsi_monitor_set(uint32_t types,
                      TEE_Param params[TEE_NUM_PARAMS])
{
    vaddr_t sys_base = core_mmu_get_va(SYS_BASE, MEM_AREA_IO_SEC);
    vaddr_t tsi_base = core_mmu_get_va(TSI_BASE, MEM_AREA_IO_SEC);
    int ret;
    uint32_t *status_buff;
    uint32_t size;

    /* Check types of parameters */
    if (types != TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INPUT,
                      TEE_PARAM_TYPE_VALUE_OUTPUT,
                      TEE_PARAM_TYPE_VALUE_INPUT,
                      TEE_PARAM_TYPE_NONE)) {
```

```
        EMSG("bad parameters types: 0x%" PRIx32, types);
        return TEE_ERROR_BAD_PARAMETERS;
    }

    /* Check TSI is enable and do monitor set*/
    if (!(io_read32(sys_base + SYS_CHIPCFG) & TSIEN)) {
        ret = TSI_Monitor_Set(params[0].value.a,        /* Interval      */
            params[0].value.b,                          /* Base address  */
            params[2].value.a,                          /* Size */
            &params[1].value.a                          /* Status return */
            );
        if (ret != ST_SUCCESS) {
              EMSG("TSI Monitor Set failed! 0x%x\n", ret);
            return ret;
        }
        return TEE_SUCCESS;
    }


    return 0x00000002;
}

/*
 * Send status check command to TSI
 */
static TEE_Result tsi_monitor_status(uint32_t types,
                     TEE_Param params[TEE_NUM_PARAMS])
{
    vaddr_t sys_base = core_mmu_get_va(SYS_BASE, MEM_AREA_IO_SEC);
    vaddr_t tsi_base = core_mmu_get_va(TSI_BASE, MEM_AREA_IO_SEC);
    int ret;

    /* Check types of parameters*/
    if (types != TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INPUT,
                    TEE_PARAM_TYPE_VALUE_OUTPUT,
                     TEE_PARAM_TYPE_NONE,
                     TEE_PARAM_TYPE_NONE)) {
        EMSG("bad parameters types: 0x%" PRIx32, types);
        return TEE_ERROR_BAD_PARAMETERS;
    }


    /* Check TSI is enable and do monitor set */
    if (!(io_read32(sys_base + SYS_CHIPCFG) & TSIEN)) {
        ret = TSI_Monitor_Status(params[0].value.a,       /* Monitor ID */
            &params[1].value.a);                          /* Status return*/
        if (ret != ST_SUCCESS) {
              EMSG("TSI Monitor Status fail");
            return ret;
        }
        return TEE_SUCCESS;
    }


    return 0x00000002;
}
```

```
/*
 * Send monitor close command to TSI
 */
static TEE_Result tsi_monitor_close(uint32_t types,
                    TEE_Param params[TEE_NUM_PARAMS])
{
    vaddr_t sys_base = core_mmu_get_va(SYS_BASE, MEM_AREA_IO_SEC);
    vaddr_t tsi_base = core_mmu_get_va(TSI_BASE, MEM_AREA_IO_SEC);
    int ret;

    /* Check types of parameters */
    if (types != TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INPUT,
                   TEE_PARAM_TYPE_VALUE_OUTPUT,
                    TEE_PARAM_TYPE_NONE,
                    TEE_PARAM_TYPE_NONE)) {
        EMSG("bad parameters types: 0x%" PRIx32, types);
        return TEE_ERROR_BAD_PARAMETERS;
    }

    /* Close monitor */
    if (!(io_read32(sys_base + SYS_CHIPCFG) & TSIEN)) {
        ret = TSI_Monitor_Close(params[0].value.a,       /* Monitor ID */
            &params[1].value.a);                         /* Status return*/
        if (ret != ST_SUCCESS) {
              EMSG("TSI Monitor Close fail");
            return ret;
        }
        return TEE_SUCCESS;
    }


    return 0x00000002;
}



static TEE_Result pta_tsi_monitor_invoke_command(void *pSessionContext __unused,
              uint32_t nCommandID, uint32_t nParamTypes,
              TEE_Param pParams[TEE_NUM_PARAMS])
{
    switch (nCommandID) {
    case TA_TSI_INIT:
            return tsi_init();
    case TA_TSI_MONITOR_SET:
            return tsi_monitor_set(nParamTypes, pParams);
    case TA_TSI_MONITOR_STATUS:
            return tsi_monitor_status(nParamTypes, pParams);
    case TA_TSI_MONITOR_CLOSE:
            return tsi_monitor_close(nParamTypes, pParams);

    }

    return TEE_ERROR_NOT_IMPLEMENTED;
}
```

```
pseudo_ta_register(.uuid = PTA_TSI_MONITOR_UUID, .name = PTA_NAME,
        .flags = PTA_DEFAULT_FLAGS | TA_FLAG_DEVICE_ENUM,
        .open_session_entry_point = pta_tsi_monitor_open_session,
        .close_session_entry_point = pta_tsi_monitor_close_session,
        .invoke_command_entry_point = pta_tsi_monitor_invoke_command);
```

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

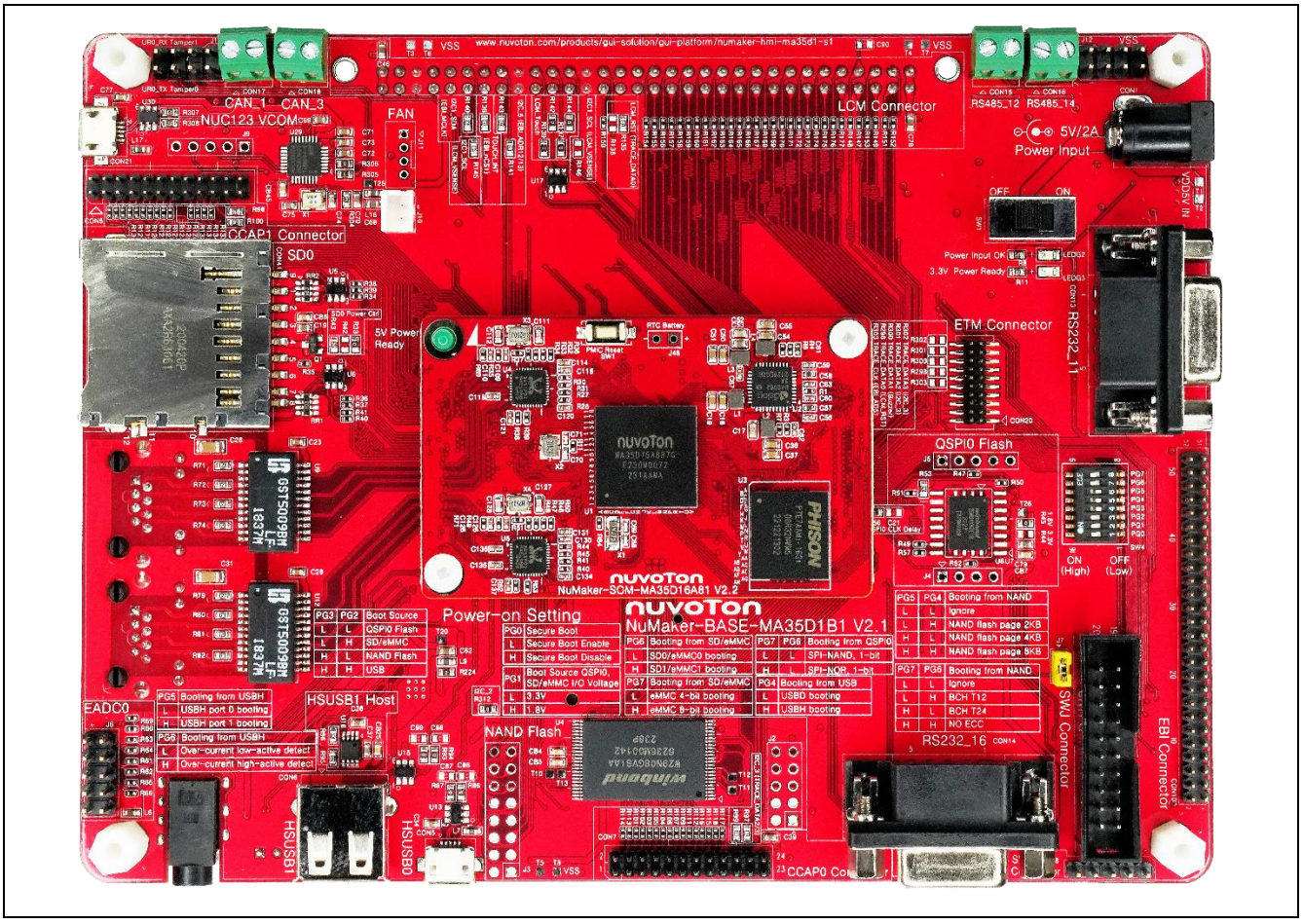


圖 3-1 NuMaker-HMI-MA35D1 V 1.0

4. 目錄資訊

目錄結構如下圖所示。

EC_MA35D1_TSI_Monitor_through_OP-TEE_V1.00	
optee_os_ma35d1	Trusted application header and source files
optee_example	Client application header and source files

表 4-1 目錄資訊

5. 範例程式執行

MA35D1 映像預設會編譯 optee-os-ma35d1 以及 optee-examples，範例要能成功執行必須修改 optee-os-ma35d1 內相關 source code，以及在 optee-examples 底下新增該 Linux 端的應用程序，以下提供 Yocto 相關步驟。

1. 進入 MA35D1 Yocto 相關路徑(根據個人設定會有所不同):
   user@662f85abcbe8:~/shared/NuMaker-SOM-MA35D1/yocto/build$ 。

2. 輸入:

```
devtool modify optee-os-ma35d1
```

3. 輸入:

```
devtool modify optee-examples
```

4. 確認路徑: NuMaker-SOM-MA35D1/yocto/build/workspace/sources 底下出現 optee-examples 以及 optee-os-ma35d1 兩個資料夾。

5. 進入 optee-os-ma35d1 資料夾下的以下路徑: NuMaker-SOM-MA35D1/yocto/build/workspace/sources/optee-os-ma35d1/core/arch/arm/plat-nuvoton, 使用範例程式中 optee_os 資料夾底下的 tsi_cmd.c 和 tsi_cmd.h，以及 pta 資料夾下的 TSI_Monitor_pta.c 和 TSI_Monitor_ta.h，替換原路徑底下的對應檔案。

6. 打開 pta 資料夾內的 sub.mk，在最下方新增:

```
srcs-y += TSI_Monitor_pta.c
```

7. 重新編譯 optee-os-ma35d1，輸入：

```
bitbake optee-os-ma35d1
```

8. 確認編譯成功。

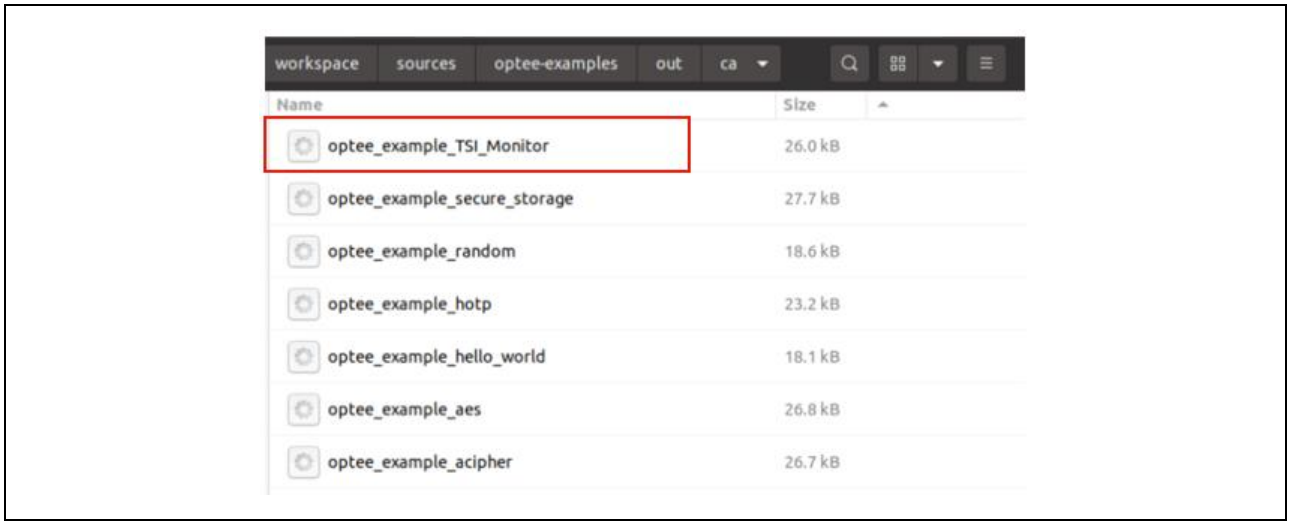
```
user@662f85abcbe8:~/shared/NuMaker-SOM-MA35D1/yocto/build$ bitbake optee-os-ma35d1
Loading cache: 100% |#########################################################################################################
Loaded 3521 entries from dependency cache.
Parsing recipes: 100% |#######################################################################################################
Parsing of 2384 .bb files complete (2382 cached, 2 parsed). 3523 targets, 405 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.46.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-20.04"
TARGET_SYS           = "aarch64-poky-linux"
MACHINE              = "numaker-som-ma35d16a81"
DISTRO               = "nvt-ma35d1-directfb"
DISTRO_VERSION       = "5.5-dunfell"
TUNE_FEATURES        = "aarch64 cortexa35 crc crypto"
TARGET_FPU           = ""
meta
meta-poky            = "HEAD:251639560dd29bd2da5e694c1d5958ca26b12d0d"
meta-oe
meta-multimedia
meta-python
meta-networking
meta-filesystems
meta-gnome           = "HEAD:430ef96fe65f62d8da995f446d5b9b093544f031"
meta-qt5             = "HEAD:b4d24d70aca75791902df5cd59a4f4a54aa4a125"
meta-ma35d1          = "HEAD:cdd0aa40ccfe84051f66c32f1d849d0331017d64"
workspace            = "<unknown>:<unknown>"
meta-virtualization  = "HEAD:89abc62b47f6f20db9d00a8ec9b2c1b6b60ac3f9"

Initialising tasks: 100% |####################################################################################################
Sstate summary: Wanted 0 Found 0 Missed 0 Current 78 (0% match, 100% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 540 tasks of which 540 didn't need to be rerun and all succeeded.
user@662f85abcbe8:~/shared/NuMaker-SOM-MA35D1/yocto/build$
```

9. 進入 optee-examples 資料夾下的以下路徑: NuMaker-SOM-MA35D1/yocto/build/workspace/sources/examples, 將範例程式中的 TSI_Monitor 複製到該路徑底下。

10. 重新編譯 optee-examples，輸入:

```
bitbake optee-examples
```

11. 確認編譯成功，確認資料夾 ca 底下生成範例 optee_example_TSI_Monitor。



12. 編譯 MA35D1 映像檔，輸入:

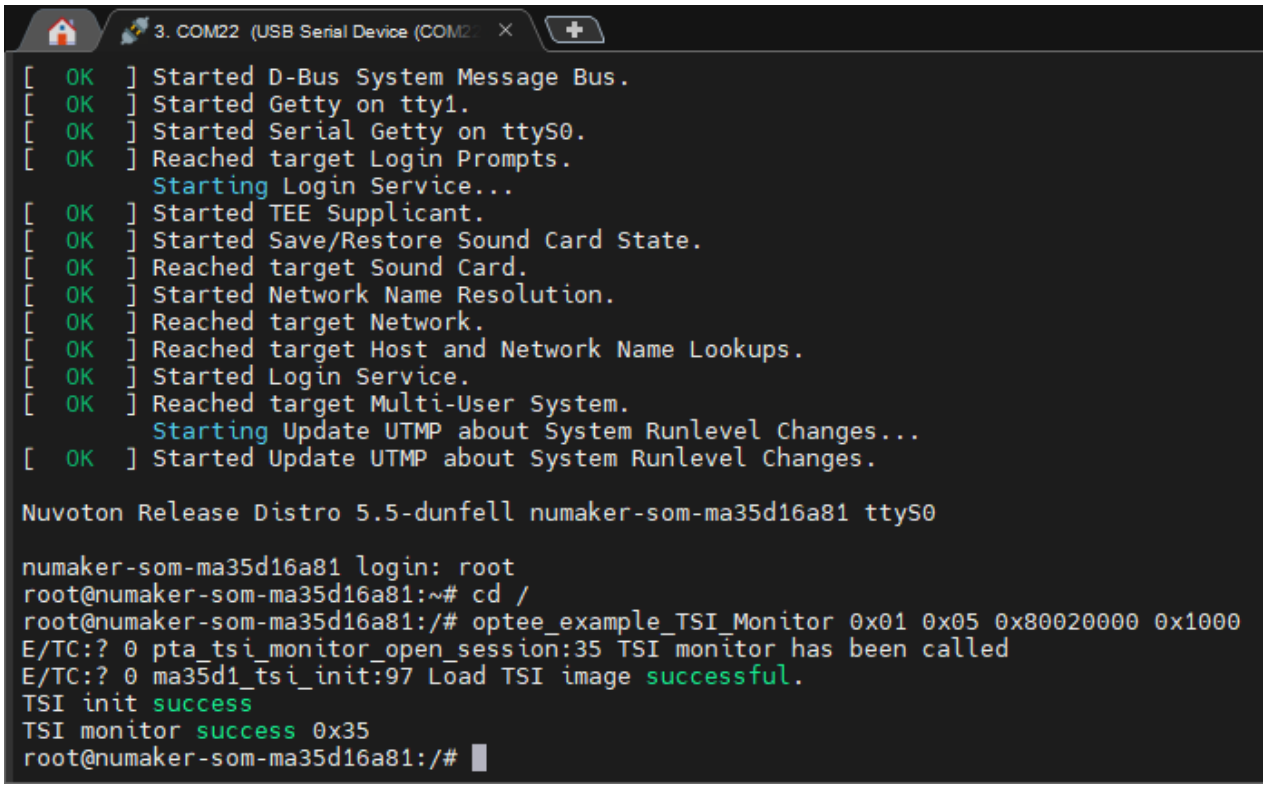
```
bitbake core-image-minimal
```

13. 編譯完成後將 core-image-minimal-numaker-som-ma35d16a81-202xxxxxxxxxx.rootfs.sdcard 燒錄至開機用的 SD 卡中。

14. 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，第一步需要先設定 monitor 區域，輸入: optee_example_TSI_Monitor 0x01 0x05 0x80020000 0x1000，各參數分別表示:

    - TSI_Monitor_Set command (0x01)
    - Monitor time interval (0x05)
    - Monitor base address (0x80020000)
    - Monitor size (0x1000)

15. 設定成功 TSI 會回傳一組 status code 0x35，為該區域的表示 ID。



16. 第二步嘗試請求 TSI 回報監控區域的狀態，輸入: optee_example_TSI_Monitor 0x02 0x35，各參數分別表示:

    - TSI_Monitor_Status command (0x02)

- Monitor ID (0x35)

設定成功 TSI 會回傳一組 status code(0x1430035)，其中 143 為 0x0143，表示輸入的命令為 TSI_Monitor_Status ，00 為 0X00 表示無異常狀態，35 則為該監控區域對應的表示 ID。

```
Nuvoton Release Distro 5.5-dunfell numaker-som-ma35d16a81 ttyS0

numaker-som-ma35d16a81 login: root
root@numaker-som-ma35d16a81:~# cd /
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x05 0x80020000 0x1000
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 ma35d1_tsi_init:97 Load TSI image successful.
TSI init success
TSI monitor success 0x35
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x02 0x35
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
TSI monitor status 0x1430035
root@numaker-som-ma35d16a81:/#
```

17. 使用 devmem2 嘗試修改監控區域的記憶體數值，先輸入: devmem2 0x80020000，確認目前數據。再輸入: devmem2 0x80020000 w 0x12345678，修改位置 0x80020000 為 0x12345678。再次請求 TSI 回報監控區域狀態，收到 0x1434335，error code 0x43，表示偵測到該記憶體區域的數據已被修改。

```
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x02 0x35
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
TSI monitor status 0x1430035
root@numaker-som-ma35d16a81:/# devmem2 0x80020000
/dev/mem opened.
Memory mapped at address 0xffff8eff2000.
Read at address  0x80020000 (0xffff8eff2000): 0xAAAAAAAA
root@numaker-som-ma35d16a81:/# devmem2 0x80020000 w 0x12345678
/dev/mem opened.
Memory mapped at address 0xffffa0ae1000.
Read at address  0x80020000 (0xffffa0ae1000): 0xAAAAAAAA
Write at address 0x80020000 (0xffffa0ae1000): 0x12345678, readback 0x12345678
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x02 0x35
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 tsi_monitor_status:136 TSI Monitor Status fail
TSI monitor status 0x1434335
root@numaker-som-ma35d16a81:/#
```

18. 第三步嘗試請求 TSI 關閉監控功能，輸入: optee_example_TSI_Monitor 0x03 0x35，各參數分別表示:

- TSI_Monitor_Close command (0x03)
- Monitor ID (0x35)

關閉監控 TSI 成功會回傳一組 status code (0x1440035)，其中 144 為 0x0144，表示輸入的命令 TSI_Monitor_Close ，00 為 0X00 表示關閉成功，35 則為該關閉區域對應的表示 ID。

```
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x03 0x35
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
TSI Close status 0x1440035
root@numaker-som-ma35d16a81:/#
```

19. 當設定 monitor 區域超過 4 組時，會收到錯誤碼 0x41(All monitors are used) 的回覆。

```
numaker-som-ma35d16a81 login: root
root@numaker-som-ma35d16a81:~# cd /
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x05 0x80020000 0x1000
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 ma35d1_tsi_init:97 Load TSI image successful.
TSI init success
TSI monitor success 0x35
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x05 0x80030000 0x1000
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 ma35d1_tsi_init:97 Load TSI image successful.
TSI init success
TSI monitor success 0x36
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x05 0x80040000 0x1000
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 ma35d1_tsi_init:97 Load TSI image successful.
TSI init success
TSI monitor success 0x37
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x05 0x80050000 0x1000
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 ma35d1_tsi_init:97 Load TSI image successful.
TSI init success
TSI monitor success 0x38
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x05 0x80060000 0x1000
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 ma35d1_tsi_init:97 Load TSI image successful.
TSI init successE/TC:? 0 tsi_monitor_set:99 TSI Monitor Set failed! 0x41

optee_example_TSI_Monitor: TEEC_InvokeCommand failed with code 0x41 origin 0x4
root@numaker-som-ma35d16a81:/#
```

20. 當輸入錯誤的 monitor id 時，會收到錯誤碼 0x42(Invalid monitor ID) 的回覆。

```
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x03 0x41
E/TC:? 0 pta_tsi_monitor_open_session:35 TSI monitor has been called
E/TC:? 0 tsi_monitor_close:170 TSI Monitor Close fail
optee_example_TSI_Monitor: TEEC_InvokeCommand failed with code 0x42 origin 0x4
root@numaker-som-ma35d16a81:/#
```

6. 修訂紀錄

Date	Revision	Description
2024.09.24	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.