MA35D1 使用 OP-TEE 进行 TSI 安全监控

NuMicro® 64 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码透过 OP-TEE 实现使用 MA35D1 的协同处理器 TSI 做内存区块的安全监控功能
BSP 版本	Linux-5.10.x
开发平台	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

本范例代码使用 MA35D1 内的 TSI 进行特定内存区段的完整性检测。

TSI 全名为 Trusted Secure Island，为 MA35D1 SoC 的一部分，旨在提供安全的加密和密钥储存等服务。在 MA35D1 中，TSI 是一个独立的子系统，并且与 Cortex®-A35 主系统各自独立运行。MA35D1 Cortex-A35 上运行的程序访问 TSI 的唯一途径是通过 Wormhole1，这是一个安全属性的硬件组件，提供 Cortex-A35 主系统与 TSI 子系统之间双向通信。TSI 以被动方式运作，始终等待来自 Cortex-A35 的请求。它会在一个无限循环中运行，接收请求、执行请求，并通过 Wormhole1 回应 ACK。为了节省电力，如果在一段时间内未接收到请求，TSI 本身会自动进入省电模式。

OP-TEE（Open Portable Trusted Execution Environment）是一个开源的 Trusted Execution Environment（TEE, https://optee.readthedocs.io/en/latest/general/about.html），由 Linaro 主导开发，专为 ARM TrustZone® 技术设计。其目的是在嵌入式系统中提供一个安全的执行环境，用以执行敏感的应用程序和处理敏感的数据。有如下的应用场景:

- 数字签名与验证：OP-TEE 可用于保护密钥操作，确保数字签名的私钥只在安全环境中使用，防止被泄露。
- 数据加密与解密：通过 TEE 来处理加密和解密操作，保障敏感数据的安全性。
- 身份验证：使用 TEE 来存储和验证身份凭证，防止凭证被窃取或篡改。
- 支付系统：TEE 可以保护金融交易的敏感数据，防止交易过程中的安全漏洞。

本范例程序根据 OP-TEE 建立一支在 Linux 端的应用程序 (Client Application)，透过该应用程序向安全世界 (OP-TEE OS) 中的安全程序 (Trusted Application) 发送请求，再由 TA 呼叫 TSI 进行内存的完整性检验。TSI 可存取 MA35D1 的所有内存空间，透过硬件加密加速器的 SHA-256，定期计算要监控内存区的哈希值，来确保该内存空间没有被窜改。区段的起始位置以及大小可由用户自行设定，最多可以监测四段区域。由于 TSI 是一个独立的子系统，因此可以确保其计算的独立性及可靠性。

1.1 原理

Cortex-A35 只能通过 TrustZone Secure (TZS) 模式下访问 TSI。由于 Linux 属于运行在非安全世界的 OS，无权访问 Wormhole1 和 TSI，必须有一个在安全模式下运行的操作系统，例如 OP-TEE，来帮助 Linux 访问 TSI。MA35D1 的 Linux BSP 支持 OP-TEE 的 Crypto、Key Store 和 TRNG 客户端驱动程序，以协助 Linux 访问安全世界的 TSI。示意图可参考如图 1-1:

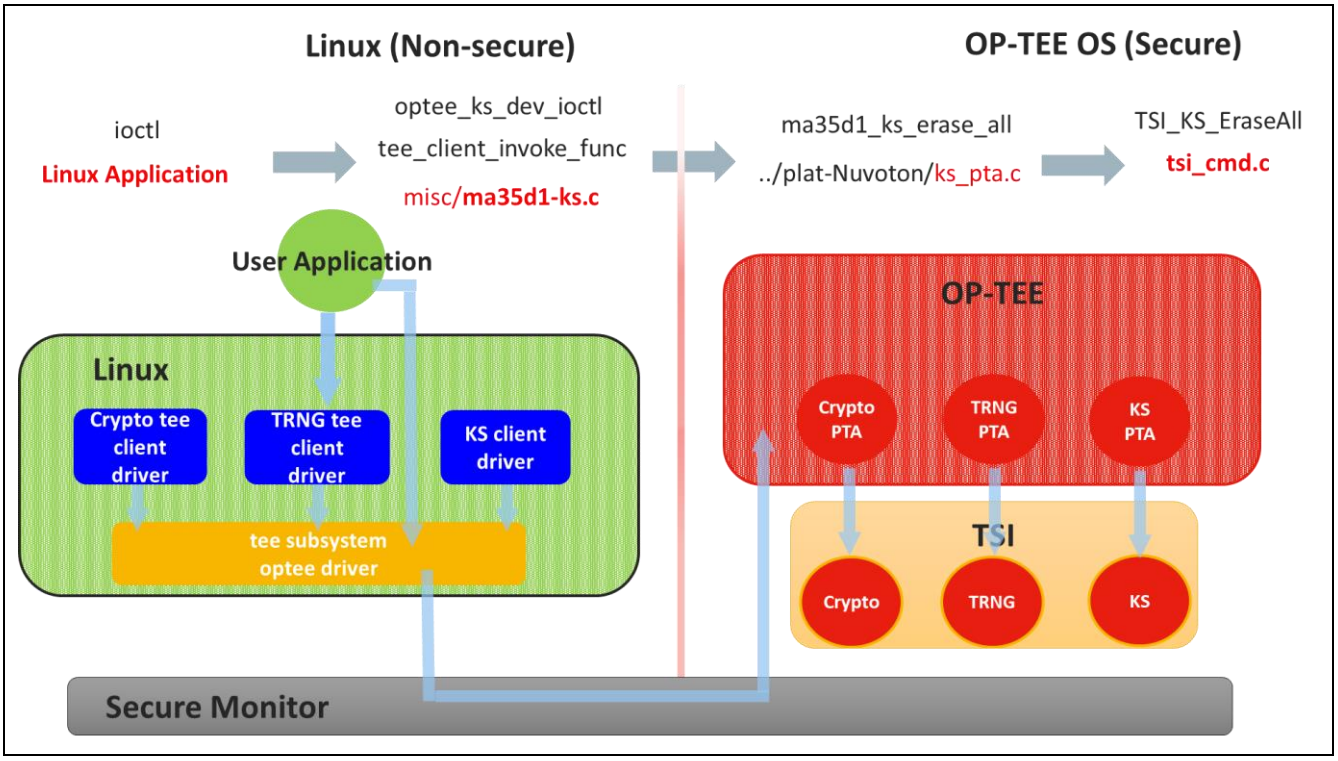


图 1-1 Linux 与 OP-TEE 沟通示意图

MA35D1 的 Wormhole1 是 TrustZone Secure (TZS) ，只有在安全模式下运行的程序才能访问 Wormhole1。A35 透过 Secure Monitor 切换安全属性 (NS->S)，进到安全模式下运行 OP-TEE OS，而 Linux 核心则在非安全模式下运行。MA35D1 Linux 核心中的 Crypto、TRNG 和 Key Store 驱动程序作为 OP-TEE 应用端驱动实现，需要 OP-TEE 帮助访问 TSI。TSI 从 Wormhole1 接收请求，执行该请求，并通过 ACK 回应执行结果，如图 1-2 所示。

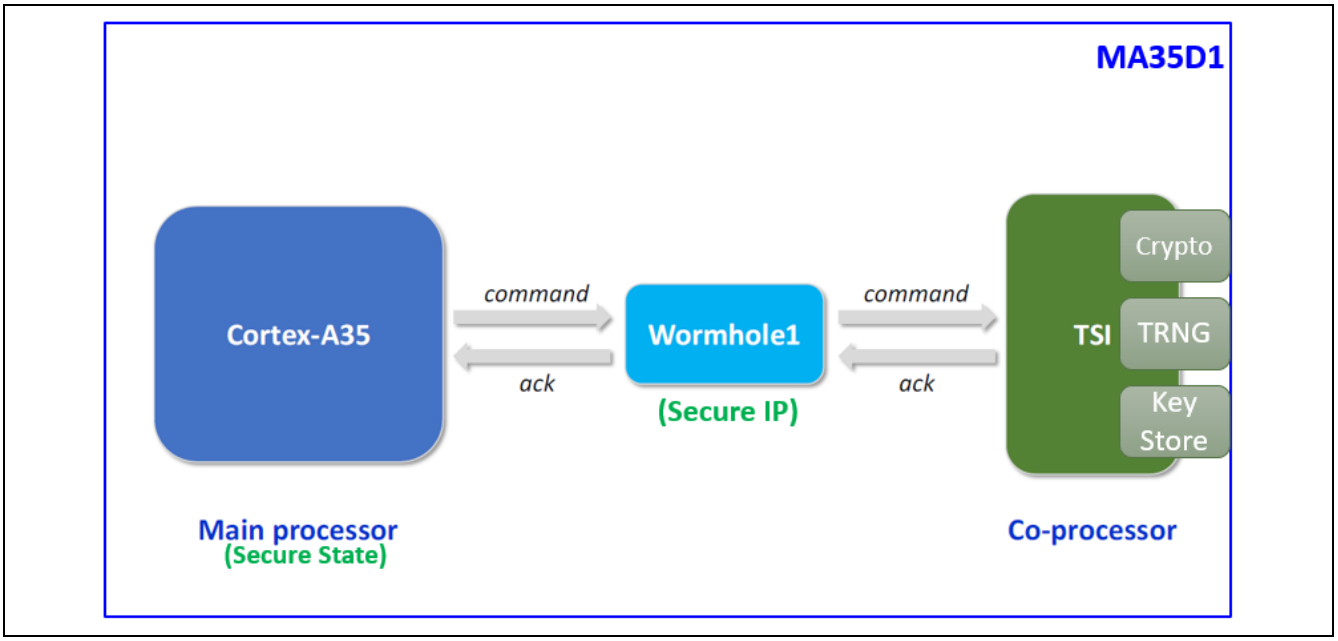


图 1-2 A35 与 TSI 通信示意图

TSI 支持 TSI_Monitor_Set、TSI_Monitor_Status 和 TSI_Monitor_Close 命令，这些命令允许 TSI 监控指定记忆区域的完整性。TSI 最多同时支持四个监控器运行。

MA35D1 使用 TSI_Monitor_Set 命令来指定要监控的记忆区域以及执行记忆检查的时间间隔。TSI 通过 SHA 硬件加速器检查并比较记忆区域。MA35D1 可以随时使用 TSI_Monitor_Status 来检查内存完整性状态。监控器会定期检查内存，直到接收到 TSI_Monitor_Close 命令为止。

- TSI_Monitor_Set

  设定要监控的内存起始位置、大小(最大:32 MB) 以及时间间隔。

- TSI_Monitor_Status

  请求 TSI 回报该监控段的状态。

- TSI_Monitor_Close

  关闭该监控段的定期检验。

1.2 执行结果

本范例代码执行结果如图 1-3 所示。MA35D1 启动成功后，进到 usr/bin 下可以看到 OP-TEE 相关范例，除 optee_example_TSI_Monitor 外皆为 OP-TEE 预设提供。执行 optee_example_TSI_Monitor，第一次带入 TSI_Monitor_Set 相关参数

(optee_example_TSI_Monitor *cmd set, time interval, base address, size*) ，设定成功后会看到 TSI 回传 ID (0x35) ，当执行 TSI_Monitor_Status 以及 TSI_Monitor_Close 时，该 ID 值须作为参数带入已成功执行范例。

第二次执行 optee_example_TSI_Monitor 范例程序，带入 TSI_Monitor_Status 相关参数 (optee_example_TSI_Monitor *cmd status, ID*)，执行成功后会看到 TSI 回传该监控区域的状态 (0x14300035)，其中 143 为 0x0143，表示输入的命令为 TSI_Monitor_Status ，00 为 0X00 表示无异常状态，35 则为该监控区域对应的表示 ID 0x35。

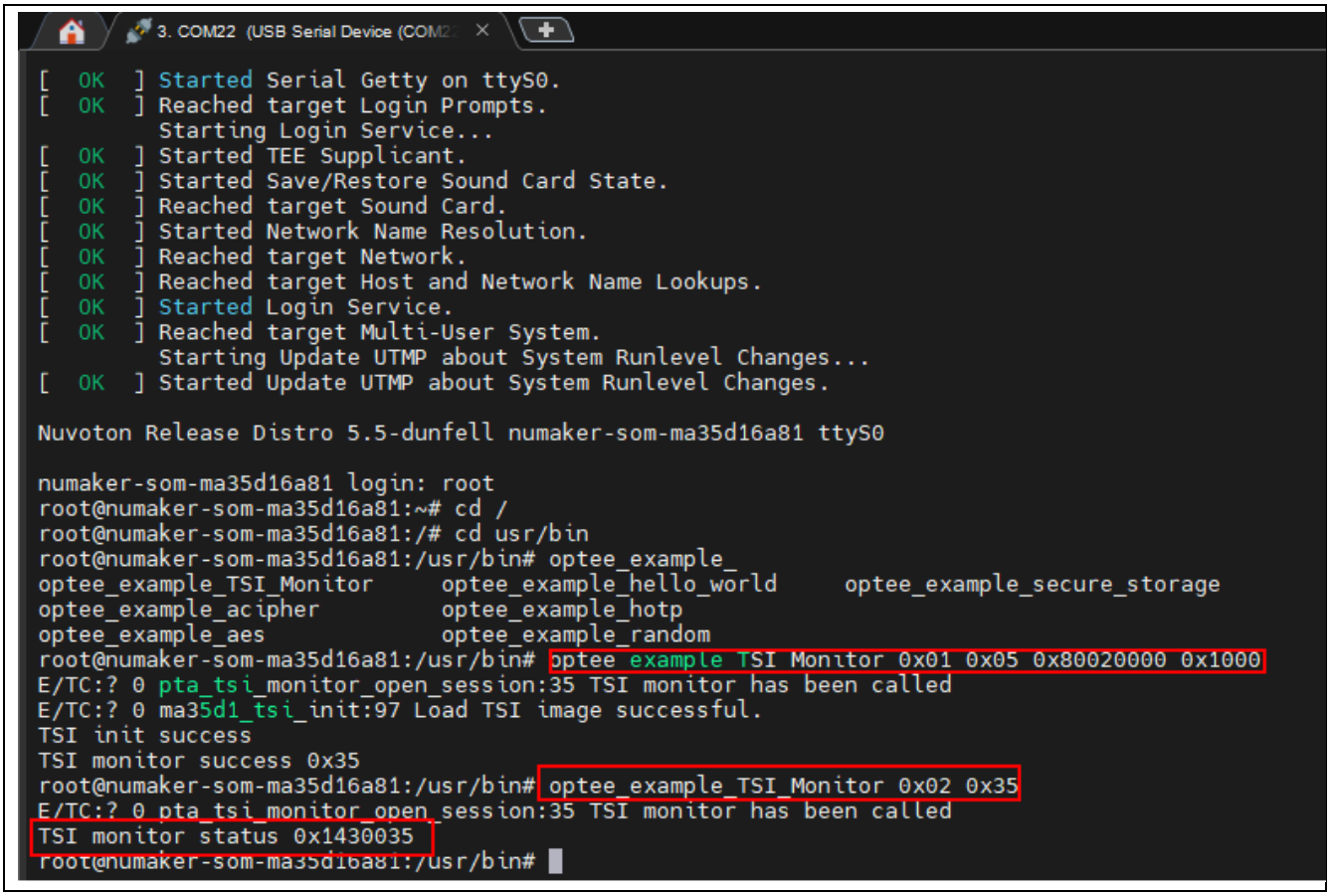


图 1-3 范例执行结果

2. 代码介绍

TSI Monitor 范例拆成两部分:

- optee_example

运行于 Linux 端的应用程序，又称作 Client Application (CA)，CA 透过 OP-TEE driver 提供的接口 API，实现相关的安全操作，CA 中的每个应用命令在 OP-TEE OS 端都有对应的操作，实际的操作工作都是在 TEE OS 或者应用层 Trusted Application (TA) 完成，再返回结果通知 CA 。

- optee_os_ma35d1

架构同样分成 OS 层以及应用层，OS 处理来自 Linux 的相关安全功能调用，应用层为实际应用 Trusted Application，主要实现如加解密，安全存储以及与底层的安全属性硬件进行互动。

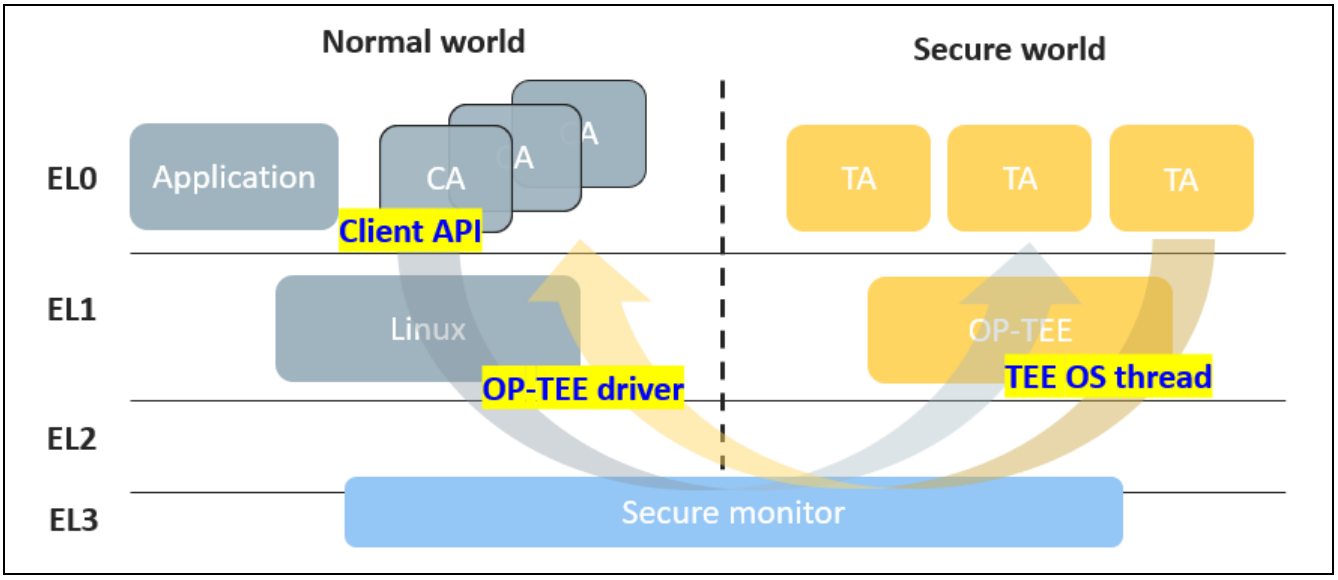


图 2-1 OP-TEE 软件架构

第一部份为 optee_example。程序代码一开始呼叫 Prepare_tee_session 确认是否成功建立 TSI monitor 会话，再透过 sscanf 读取应用参数，来确认实际要执行的操作。代码位于 *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[3], "%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;
}
```

相关操作 API 如下，提供监控设定(Monitor_Set)，状态回报(Monitor_Status)以及监控关闭(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;            // monutor 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 都有对应的功能，透过 pta 呼叫 TSI 进行一系列的操作，并返回结果给 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. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - Linux-5.10.x

3.2 硬件需求

- 电路组件
  - NuMaker-HMI-MA35D1 V1.0

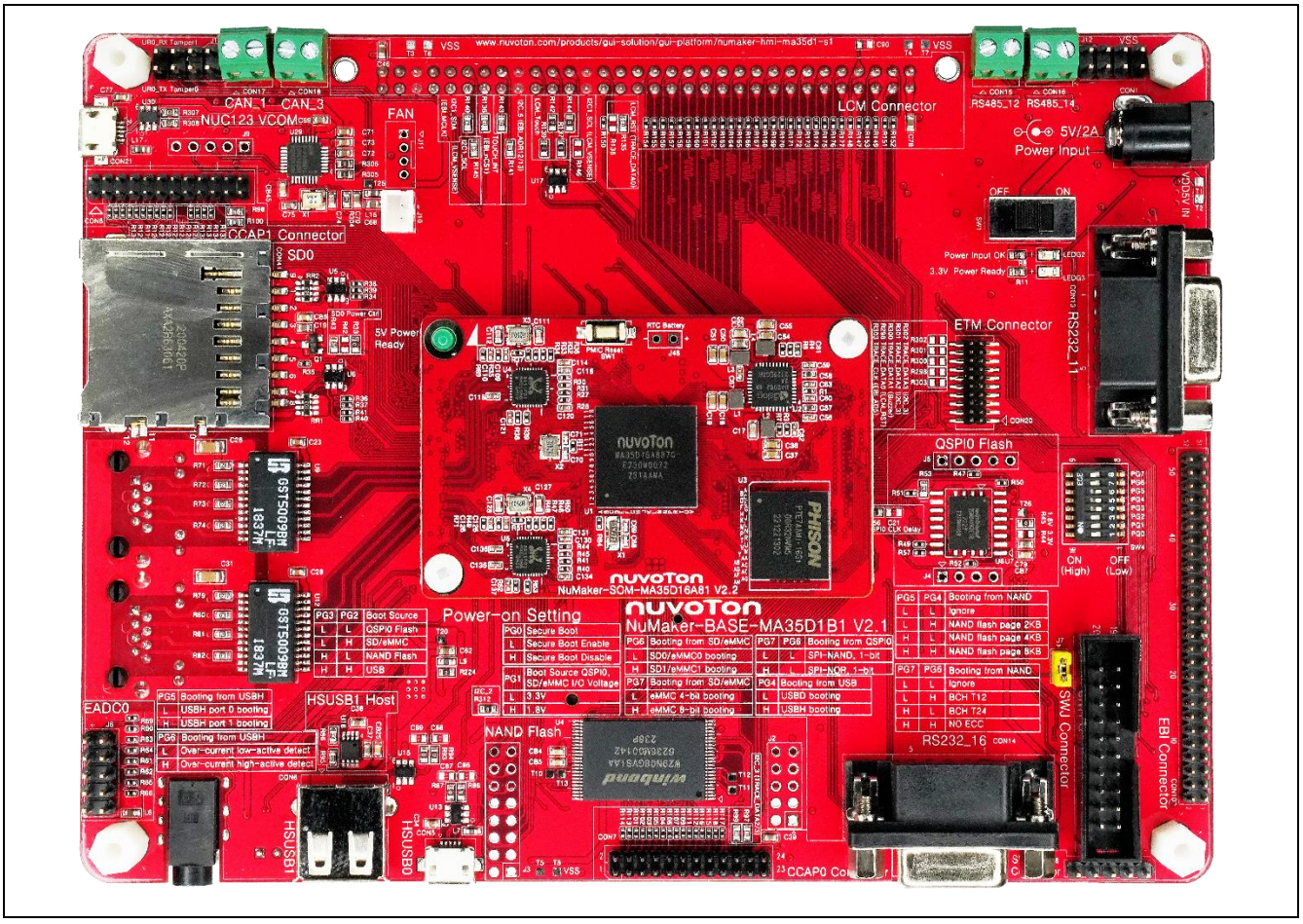


图 3-1 NuMaker-HMI-MA35D1 V 1.0

4. 目錄資訊

目录结构如下图所示。

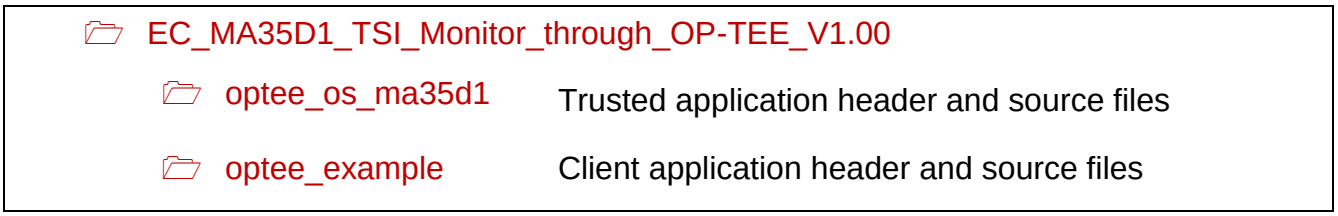


图 4-1 目录信息

5. 范例程序执行

MA35D1 映像预设会编译 optee-os-ma35d1 以及 optee-examples，范例要能成功执行必须修改 optee-os-ma35d1 内相关 source code，以及在 optee-examples 底下新增该 Linux 端的应用程序，以下提供 Yocto 相关步骤。

1. 进入 MA35D1 Yocto 相关路径(根据个人设定会有所不同):
   user@662f85abcbe8:~/shared/NuMaker-SOM-MA35D1/yocto/build$ 。
2. 输入:

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

3. 输入:

```
devtool modify optee-examples
```

4. 确认路径: NuMaker-SOM-MA35D1/yocto/build/workspace/sources 底下出现 optee-examples 以及 optee-os-ma35d1 两个文件夹。
5. 进入 optee-os-ma35d1 文件夹下的以下路径: NuMaker-SOM-MA35D1/yocto/build/workspace/sources/optee-os-ma35d1/core/arch/arm/plat-nuvoton, 使用范例程序中 optee_os 文件夹底下的 tsi_cmd.c 和 tsi_cmd.h，以及 pta 文件夹下的 TSI_Monitor_pta.c 和 TSI_Monitor_ta.h，替换原路径底下的对应档案。
6. 打开 pta 文件夹下的 sub.mk，在最下方新增:

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

7. 重新编译 optee-os-ma35d1，输入：

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

8. 确认编译成功。

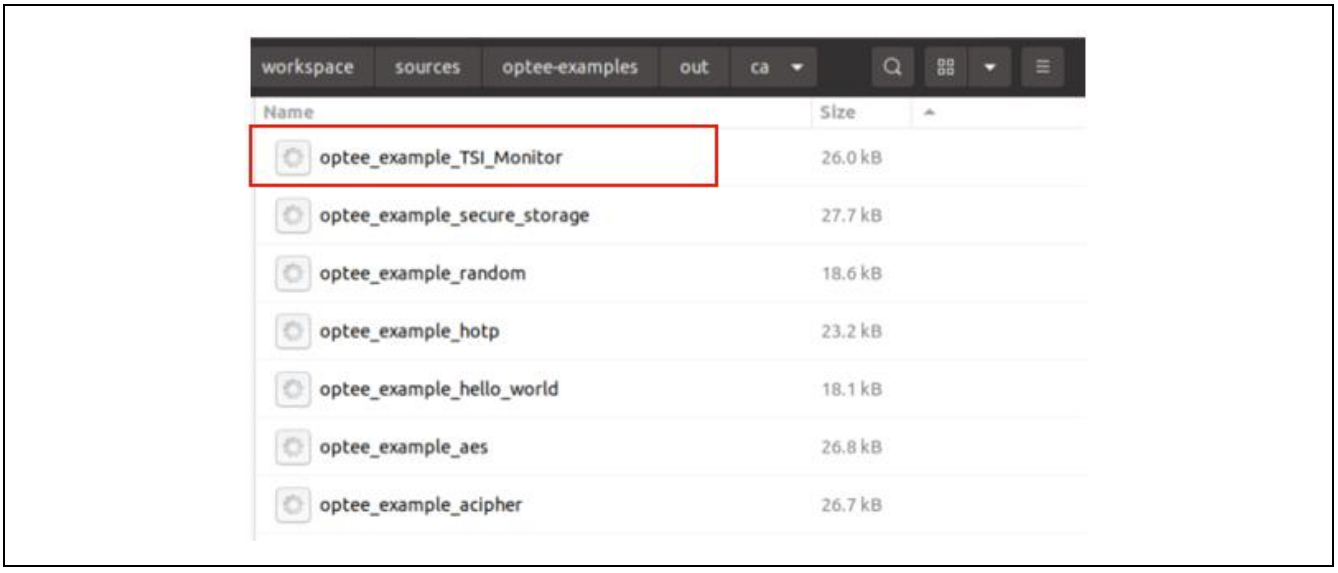
```
user@662f85abcbe8:~/shared/NuMaker-SOM-MA35D1/yocto/build$ bitbake optee-os-ma35d1
Loading cache: 100% |#########################################################################################################
Loaded 3521 entries from dependency cache.
Parsing recipes: 100% |#######################################################################################################
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. 进入 optee-examples 文件夹下的以下路径: NuMaker-SOM-MA35D1/yocto/build/workspace/sources/examples, 将范例程序中的 TSI_Monitor 复制到该路径底下。

10. 重新编译 optee-examples，输入:

```
bitbake optee-examples
```

11. 确认编译成功，确认文件夹 ca 底下生成范例 optee_example_TSI_Monitor。



12. 编译 MA35D1 映像档，输入:

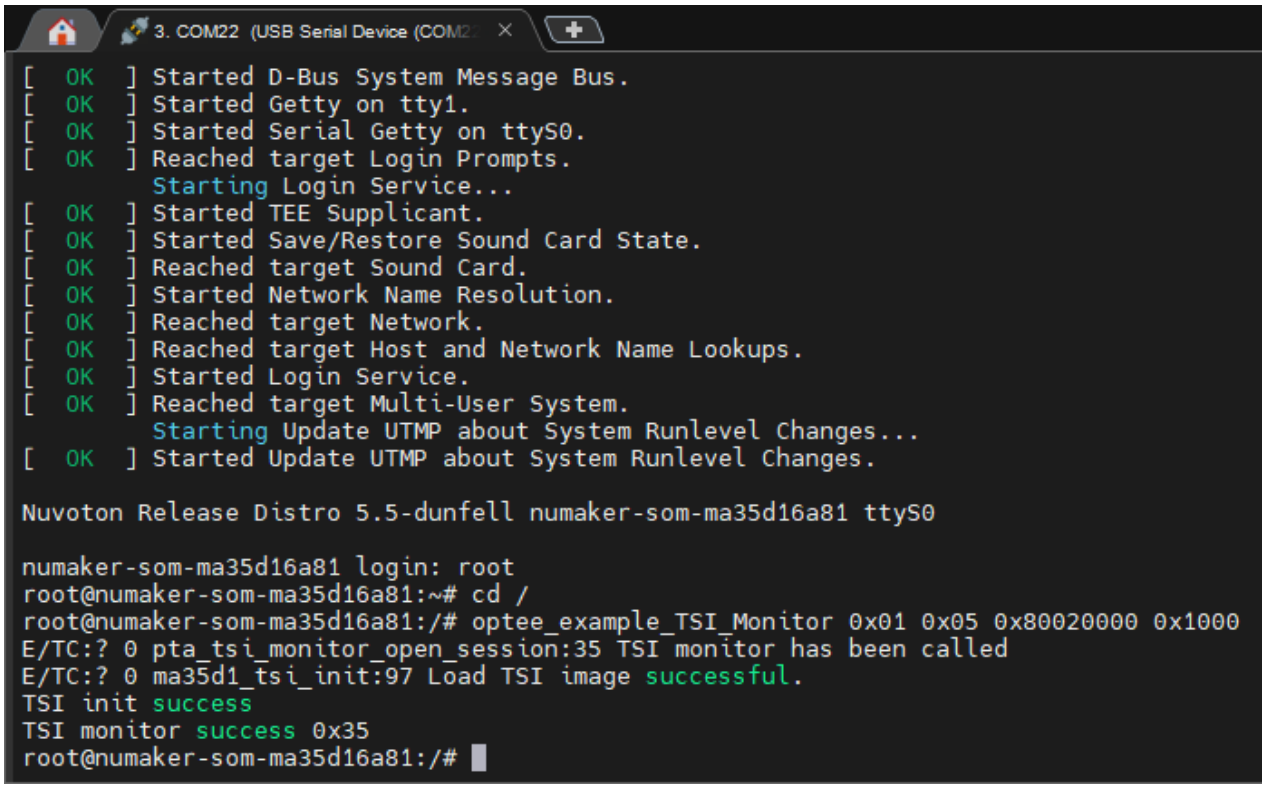
```
bitbake core-image-minimal
```

13. 编译完成后将 core-image-minimal-numaker-som-ma35d16a81-2024xxxxxxxxx.rootfs.sdcard 烧录至开机用的 SD 卡中。

14. 将开发板上的 Power-on setting 指拨开关设定为 boot from SD0，启动 MA35D1 开发板第一步需要先设定 monitor 区域，输入: optee_example_TSI_Monitor 0x01 0x05 0x80020000 0x1000，各参数分别表示:

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

15. 设定成功 TSI 会回传一组 status code 0x35，为该区域的表示 ID。



16. 第二步尝试请求 TSI 回报监控区域的状态，输入: optee_example_TSI_Monitor 0x02 0x35，各参数分别表示:

    - TSI_Monitor_Status command (0x02)

- Monitor ID (0x35)

设定成功 TSI 会回传一组 status code(0x1430035)，其中 143 为 0x0143，表示输入的命令为 TSI_Monitor_Status ，00 为 0X00 表示无异常状态，35 则为该监控区域对应的表示 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 尝试修改监控区域的内存数值，先输入: devmem2 0x80020000，确认目前数据。再输入: devmem2 0x80020000 w 0x12345678，修改位置 0x80020000 为 0x12345678。再次请求 TSI 回报监控区域状态，收到 0x1434335，error code 0x43，表示侦测到该内存区域的数据已被修改。

```
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. 第三步尝试请求 TSI 关闭监控功能，输入: optee_example_TSI_Monitor 0x03 0x35，各参数分别表示:

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

关闭监控 TSI 成功会回传一组 status code (0x1440035)，其中 144 为 0x0144，表示输入的命令 TSI_Monitor_Close ，00 为 0X00 表示关闭成功，35 则为该关闭区域对应的表示 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. 当设定 monitor 区域超过 4 组时，会收到错误码 0x41(All monitors are used) 的回复。

```
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x5 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 success35
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x5 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 success36
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x5 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 success37
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x5 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 success38
root@numaker-som-ma35d16a81:/# optee_example_TSI_Monitor 0x01 0x5 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:95 TSI Monitor Set failed! 0x41

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

20. 当输入错误的 monitor id 时，会收到错误码 0x42(Invalid monitor ID) 的回复。

```
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. 修订纪录

Date	Revision	Description
2024.09.24	1.00	初始发布。

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.