NuMicro® Family

Arm® Cortex®-A35-based Microcontroller

NuMicro® Family
MA35D1 OP-TEE
User Manual

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

Table of Contents

1 OVERVIEW

Nuvoton MA35D1 adopts Linaro OP-TEE solution to provide secure-world service for non-secure world Linux applications. OP-TEE is a Trusted Execution Environment (TEE) designed as companion to a non-secure Linux kernel. Detail functions about OP-TEE please refer to https://optee.readthedocs.io/en/latest/ .

In MA35D1, some of the hardware components can be configured to work under secured or non-secured mode, such as TRNG, Key Store, and Cryptographic Accelerator. If the hardware component is configured as non-secured, it can be directly controlled by Linux kernel drivers. The MA35D1 Linux BSP provides corresponding kernel drivers. Once the hardware component is configured as secured, Linux will not be able to access it because Linux is running in non-secure world. To access the secured hardware component can only be accomplished by the help of OP-TEE.

The MA35D1 implemented three PTAs (Pseudo Trusted Application) in OP-TEE Nuvoton platform implementation. These are TRNG PTA, KS PTA, and Crypto PTA. They are built-in to OP-TEE core image. The corresponding implementation in Linux kernel are in TRNG driver (drivers/char/hwrng/ma35d1_trng.c), Key Store driver (drivers/misc/ma35d1_ks.c), and Crypto driver (drivers/crypto/platform/nuvoton). On the initialization of Linux kernel, the Linux tee/optee driver will enumerate devices of OP-TEE and these PTAs should be enumerated at this time.

Linux OP-TEE client driver can create context, session, and invoke OP-TEE functions to request OP-TEE PTA service. The communication will be accomplished via SMC call. Thus, the application running at non-secure world (Linux) can get service from secured IP running in secured world (OP-TEE PTA).

2 OP-TEE CONFIGURATION

The MA35D1 platform dependent implementations of OP-TEE are gathered in OP-TEE OS core/arch/arm/plat-nuvoton folder.

2.1 conf.mk

“conf.mk” contains platform specific memory arrangement, CPU related definition, and vendor defined macros.

“PLATFORM_FLAVOR” claims vendor specific platform.

```
PLATFORM_FLAVOR ?= MA35D1
```

The following line specifies MA35D1 to be dual-core.

```
$(call force,CFG_TEE_CORE_NB_CORE,2)
```

The following lines specifies trust-zone DRAM start address and size for OP-TEE. This trust zone DRAM area is used by OP-TEE OS and to load TA (Trusted Application) for execution on run-time.

```
CFG_TZDRAM_START ?= 0x8f000000
CFG_TZDRAM_SIZE ?=  0x00700000
```

The following lines specifies start address and size of share memory. This share memory is shared by Linux kernel and OP-TEE kernel. Linux client driver and client application can request OP-TEE to allocate a block of share memory, and send/receive data to/from OP-TEE.

```
CFG_SHMEM_START ?=  0x8ff00000
CFG_SHMEM_SIZE ?=   0x00100000
```

3 OP-TEE FEATURES

The MA35D1 platform port of OP-TEE keeps all native support function of OP-TEE.

- TEE Client API Specification v1.0
- TEE Internal API Specification v1.1

3.1 MA35D1 platform port of OP-TEE

The MA35D1 supports hardware Cryptographic Accelerator, Key Store, and TRNG. In OP-TEE core, the MA35D1 provides pseudo trusted applications (pseudo TAs) to achieve support for these hardware components.

3.1.1 MA35D1 Pseudo Trusted Application

The MA35D1 pseudo TAs provide service to corresponding Linux client drivers via SMC call. OP-TEE has defined the communication path between Linux tee driver. MA35D1 Crypto driver, Key Store driver, and TRNG driver register to the Linux tee driver of OP-TEE. Thus, these drivers can use the services provided by tee driver to issue request to pseudo TAs.

3.1.2 Crypto Pseudo TA

Crypto pseudo TA supports AES, SHA, ECC, and RSA. MA35D1 Linux crypto driver can work in OP-TEE mode, which is determined by device tree “optee_nuvoton” setting. If “optee_nuvoton” is “yes”, the crypto driver will execute AES, SHA, ECC, and RSA in way of OP-TEE client driver requesting cryptographic service to corresponding PTA in OP-TEE core.

The Linux crypto driver allocates a share memory block from OP-TEE, and send the shadow mapping of MA35D1 cryptographic accelerator control registers to crypto PTA via the share memory.

The crypto pseudo TA supports:

- AES 128/192/256 bits ECB/CBC/CTR/CFB/CCM/GCM mode encrypt/decrypt
- SHA 128/224/256/384/512
- ECC point multiplication
- RSA encrypt/decrypt
- SM2/SM3/SM4

3.1.3 TRNG Pseudo TA

TRNG pseudo TA supports generating true random numbers with MA35D1 hardware TRNG. MA35D1 Linux TRNG driver can work in OP-TEE mode, which is determined by device tree “optee_nuvoton” setting. If “optee_nuvoton” is “yes”, the TRNG driver will execute random number generation in way of OP-TEE client driver requesting TRNG service to corresponding PTA in OP-TEE core.

3.1.4 KS Pseudo TA

KS pseudo TA supports writing and reading keys from MA35D1 Key Store SRAM. It does not support writing or reading Key Store OTP keys. MA35D1 Linux KS driver can work in OP-TEE mode, which is determined by device tree “optee_nuvoton” setting. If “optee_nuvoton” is “yes”, the KS driver will execute Key Store operations in way of OP-TEE client driver requesting service to corresponding KS PTA in OP-TEE core.

The KS pseudo TA supports:

- Key Store SRAM key read/write
- Key Store SRAM key erase
- Get remaining space of Key Store SRAM

- Read OTP data
- Read Key Store OTP keys

4 BUILD OP-TEE

OP-TEE working environment is composed of OP-TEE OS in secured world and OP-TEE client in normal world. While booting MA35D1, the IBR (Internal Boot ROM) loads TF-A (ARM Trusted Firmware) and subsequently TF-A will load OP-TEE OS and u-boot, and finally u-boot will load Linux kernel. After the boot process, OP-TEE OS will reside in the secured world. The Crypto, TRNG, and KS pseudo TAs, as parts of OP-TEE OS will also reside in the secure world.

On Linux run-time, OP-TEE OS provide services such as cryptographic and random number generation. Linux kernel can request service to OP-TEE OS via SMC call. For SMC, refer to https://developer.arm.com/documentation/den0028/latest/. In Linux, the OP-TEE service request is executed by TEE driver, which can be found in Linux kernel folder drivers/tee/optee.

To run an OP-TEE application on Linux, besides of OP-TEE OS and Linux optee driver, the client API library and tee-supplicant are also required.

The tee-supplicant is a helper process, which can help OP-TEE OS to access storages of Linux world. The tee-supplicant can help OP-TEE to load the TA (Trusted Application) from storage in a secure way. Generally, OP-TEE trusted applications are encrypted and stored in storages of Linux world. Once the OP-TEE client request service to the TA, OP-TEE OS will ask tee-supplicant to help load the TA.

OP-TEE has support for GlobalPlatform TEE Client API Specification v1.0. User application can just invoke the client API to complete the work he wants to achieve.

The chapter presents how to build the OP-TEE OS, client API, and user application.

4.1 Build OP-TEE with Yocto

In MA35D1 yocto release, build OP-TEE is simple. Please first find the local.conf in the yocto/mybuild/conf folder, where “mybuild” is the folder name specified on creating the yocto build environment. Open local.conf with a text editor and insert the following line to it

```
MACHINE_FEATURES_append = "optee"
```

This will add OP-TEE stuff into the yocto build, including optee-os-ma35d1, optee-client, and optee-examples. And the kernel device tree will also include the OP-TEE driver. Then re-build yocto can have OP-TEE included.

After MA35D1 Linux startup, device *tee0* and *teepriv0* should be presented if OP-TEE is successfully included.

```
root@numaker-som-ma35d16a81:~# ls /dev/tee*
/dev/tee0      /dev/teepriv0
```

And *tee-supplicant* should have been running in background.

```
root@numaker-som-ma35d16a81:~# ps
…
237 root      2332 S    /usr/bin/tee-supplicant
```

Several OP-TEE examples can be found in /usr/bin and can be executed directly.

```
root@numaker-som-ma35d16a81:/usr/bin# ls optee*
optee_example_acipher         optee_example_hello_world     optee_example_random
optee_example_aes             optee_example_hotp
optee_example_secure_storage
root@numaker-som-ma35d16a81:/usr/bin# optee_example_hello_world
Invoking TA to increment 42
TA incremented value to 43
```

```
root@numaker-som-ma35d16a81:/usr/bin#
```

4.2 Build OP-TEE with Buildroot

In MA35D1 buildroot release, build OP-TEE is very simple. Select the nuvoton_ma35d1_xxx_security_defconfig, where 'xxx' is the board name. These nuvoton_ma35d1_xxx_security_defconfig have included OP-TEE support.

In menuconfig, check the following packages:

```
Target packages   --->
    Security   --->
        -*- optee-client
        [*] optee-examples
```

After MA35D1 Linux startup, device *tee0* and *teepriv0* should be presented.

```
# ls /dev/tee*
/dev/tee0      /dev/teepriv0
#
```

And *tee-supplicant* should have been running in background.

```
# ps
…
196 root     /usr/sbin/tee-supplicant -d /dev/teepriv0
```

Several OP-TEE examples can be found in /usr/bin and can be executed directly.

```
# ls /usr/bin/opt*
/usr/bin/optee_example_acipher         /usr/bin/optee_example_hotp
/usr/bin/optee_example_aes             /usr/bin/optee_example_random
/usr/bin/optee_example_hello_world     /usr/bin/optee_example_secure_storage
# /usr/bin/optee_example_hello_world
Invoking TA to increment 42
TA incremented value to 43
#
```

4.3 Build OP-TEE Standalone

The OP-TEE OS, client API library, and examples can be built independently. But it's always suggested to use yocto or buildroot.

The following sections introduce how to build OP-TEE OS, client API library, and examples individually and how to add them to MA35D1 Linux.

4.3.1 Build OP-TEE OS

First, install the Python package:

```
$ pip3 install pycryptodomex
```

The OP-TEE OS source can be found at https://github.com/OP-TEE/optee_os.

Download OP-TEE OS source and then use following command to build OP-TEE.

```
make CROSS_COMPILE_core=aarch64-linux-gnu- CROSS_COMPILE64=aarch64-linux-gnu-
CFG_ARM64_core=y CFG_TEE_BENCHMARK=n CFG_TEE_CORE_LOG_LEVEL=3
CROSS_COMPILE_ta_arm64=aarch64-linux-gnu- PLATFORM=nuvoton-MA35D1
```

If build success, image “tee-pager_v2.bin” and header “tee-header_v2.bin” can be found in “optee_os/out/arm-plat-nuvoton/core” folder. TF-A fiptool can create the FIP image by combine these two files with uboot.bin and TF-A firmware.

4.3.2 Build OP-TEE Client

The OP-TEE client source can be found at https://github.com/OP-TEE/optee_client.

Download OP-TEE OS source and then use following command to build OP-TEE.

```
$ make CROSS_COMPILE=aarch64-linux-gnu-
```

It will generate an “out” folder. The OP-TEE client API library and export headers can be found in “out” folder. It will also generate the tee-supplicant binary in “out/tee-supplicant” folder.

4.3.3 Build OP-TEE Example

The OP-TEE application example source can be found at https://github.com/linaro-swg/optee_examples .

You must build optee_os and optee_client first. Then export the platform and OP-TEE client API library and path:

```
$ export HOST_CROSS_COMPILE=aarch64-linux-gnu-
$ export TA_CROSS_COMPILE=aarch64-linux-gnu-
$ export TEEC_EXPORT=$(OPTEE_PATH)/optee_client/libteec
```

The OP-TEE application run in Linux should have the corresponding Trusted Application running in OP-TEE secure world. Below we show how to build the hello_world example, which can be found in “optee_examples/hello_world”. The hello_world example contains two sub-folders “host” and “ta”. “host” is the Host Application running in Linux, while “ta” is the Trusted Application running in OP-TEE.

Build the Host Application of hello_world example:

```
$ cd optee_examples/hello_world/host
$ make \
    CROSS_COMPILE=aarch64-linux-gnu- \
    TEEC_EXPORT=$(OPTEE_PATH)/optee_client/out/export/usr \
    --no-builtin-variables
```

After build completed, the Host Application image “optee_example_hello_world” can be found in “hello_world/host” folder.

Build the Trusted Application of hello_world example:

```
$ cd optee_examples/hello_world/ta
$ make \
    CROSS_COMPILE=aarch64-linux-gnu- \
    PLATFORM=nuvoton-MA35D1 \
    TA_DEV_KIT_DIR=$(OPTEE_PATH)/optee_os-v3.9.0/out/arm-plat-nuvoton/export-
ta_arm64
```

After build completed, the Trusted Application image “8aaaf200-2450-11e4-abe2-0002a5d5c51b.ta” can be found in “hello_world/ta” folder. “8aaaf200-2450-11e4-abe2-0002a5d5c51b” is the UUID of hello_world Trusted Application. The UUID is defined in “hello_world/ta/hello_world_ta.h”. When

running Host Application, it can use this UUID to request OP-TEE to launch this TA.

4.3.4 Run Example

To run an OP-TEE application, besides have the running OP-TEE OS, you should have tee-supplication, the Trusted Application, and the Host Application. The Trusted Application will be loaded by tee-supplicant, and the default path is “rootfs/lib/optee_armtz” folder.

Do the following steps. First, copy the tee-supplicant to rootfs:

```
$ cp $(OPTEE_PATH)/optee_client/out/tee-supplicant/tee-supplicant rootfs/usr/bin
```

Copy the Host Application to rootfs:

```
$ cp $(OPTEE_PATH)/optee_examples/hello_world/host/optee_example_hello_world
rootfs/usr/bin
```

Copy the Trusted Application to tee-supplicant known default folder:

```
$ cp $(OPTEE_PATH)/optee_examples/hello_world/
optee_examples/hello_world/ta/8aaaf200-2450-11e4-abe2-0002a5d5c51b.ta
rootfs/lib/optee_armtz
```

Copy the OP-TEE client API library to rootfs:

```
$ cp $(OPTEE_PATH)/optee_client/out/export/usr/lib/libteec.so.1 rootfs/lib
```

Above should include all necessary components to run the OP-TEE application. Before running the OP-TEE application, you should have the tee-supplicant running in background. Such that, it can help OP-TEE to load the Trusted Application from “rootfs/lib/optee_armtz” folder.

```
$ /usr/bin/tee-supplicant &
```

Finally, you can run the hello_world example:

```
$ /usr/bin/optee_example_hello_world
```

5 REVISION HISTORY

Date	Revision	Description
2022.03.09	1.00	Initial version.
2022.11.28	1.01	Grammar and spelling check Update document format and cover page Updated content to match with the actual status of BSP

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.