MMC overview

Contents

Stable: 14.05.2020 - 09:26 / Revision: 14.05.2020 - 09:25

Stable: 14.03.2020 - 09:23 / Revision: 14.03.2020 - 09:23

A quality version of this page, approved on *16 September 2019*, was based off this revision.

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

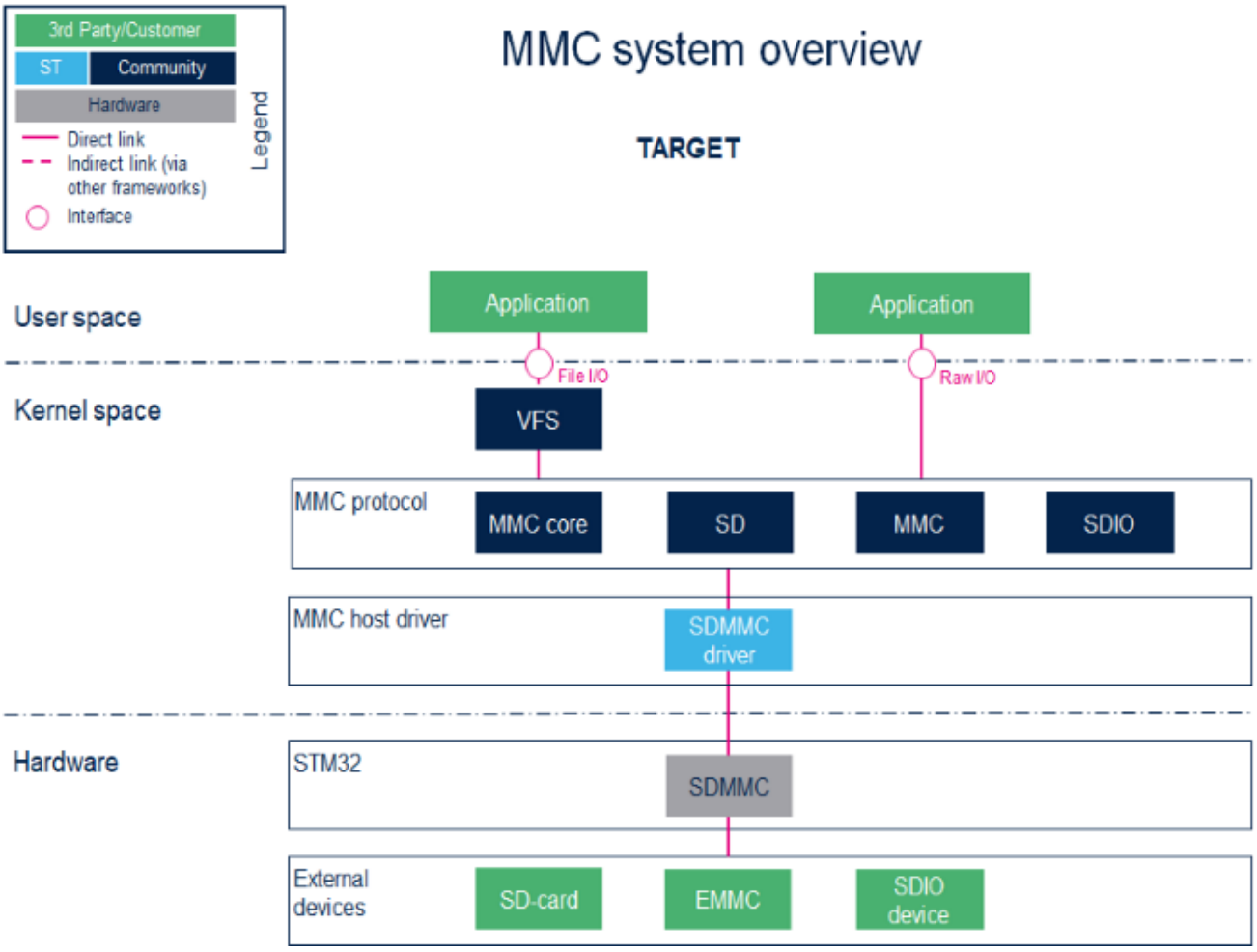
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 19.03.2021 - 13:36 / Revision: 19.03.2021 - 13:36

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

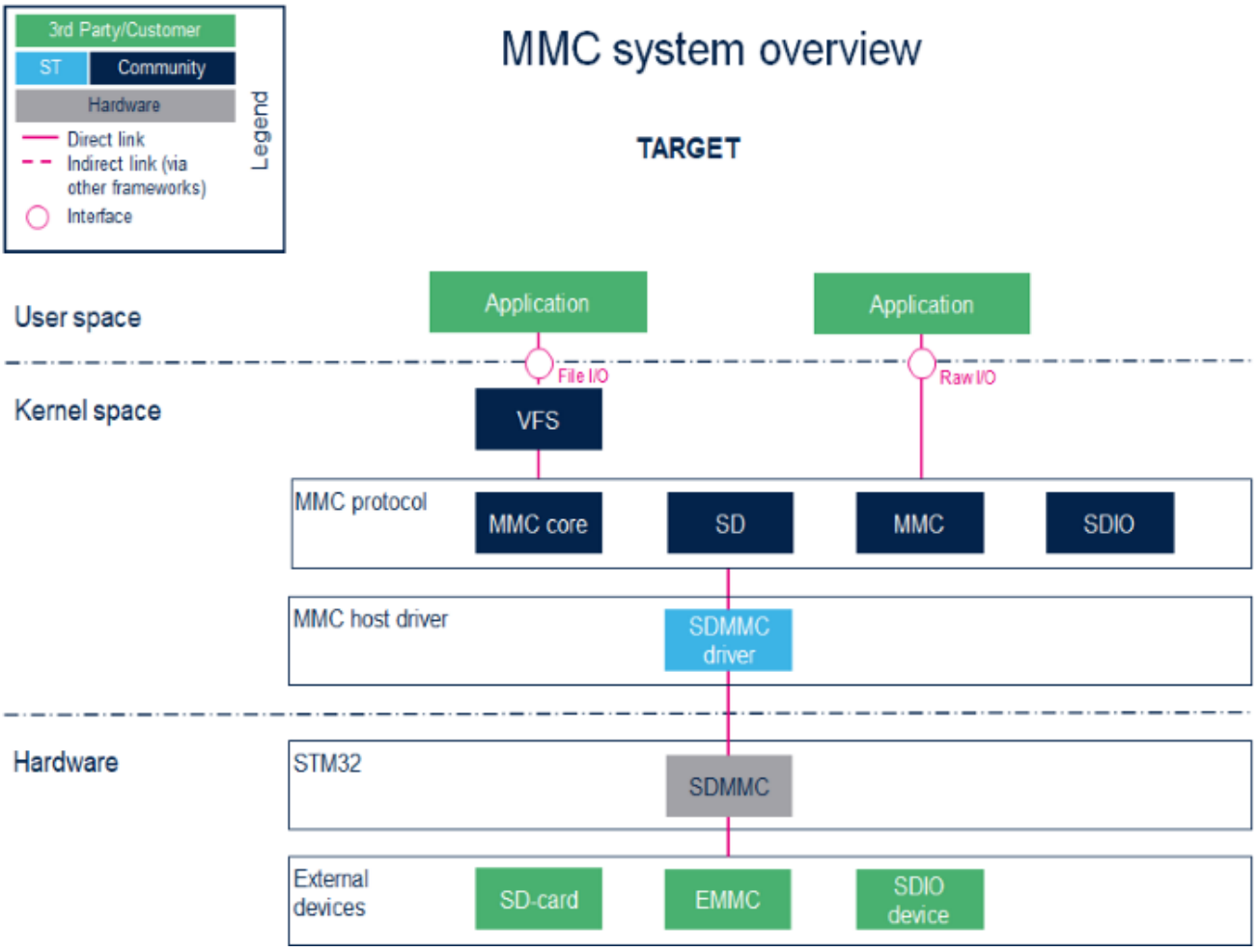
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 02.11.2020 - 10:48 / Revision: 19.10.2020 - 12:09

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

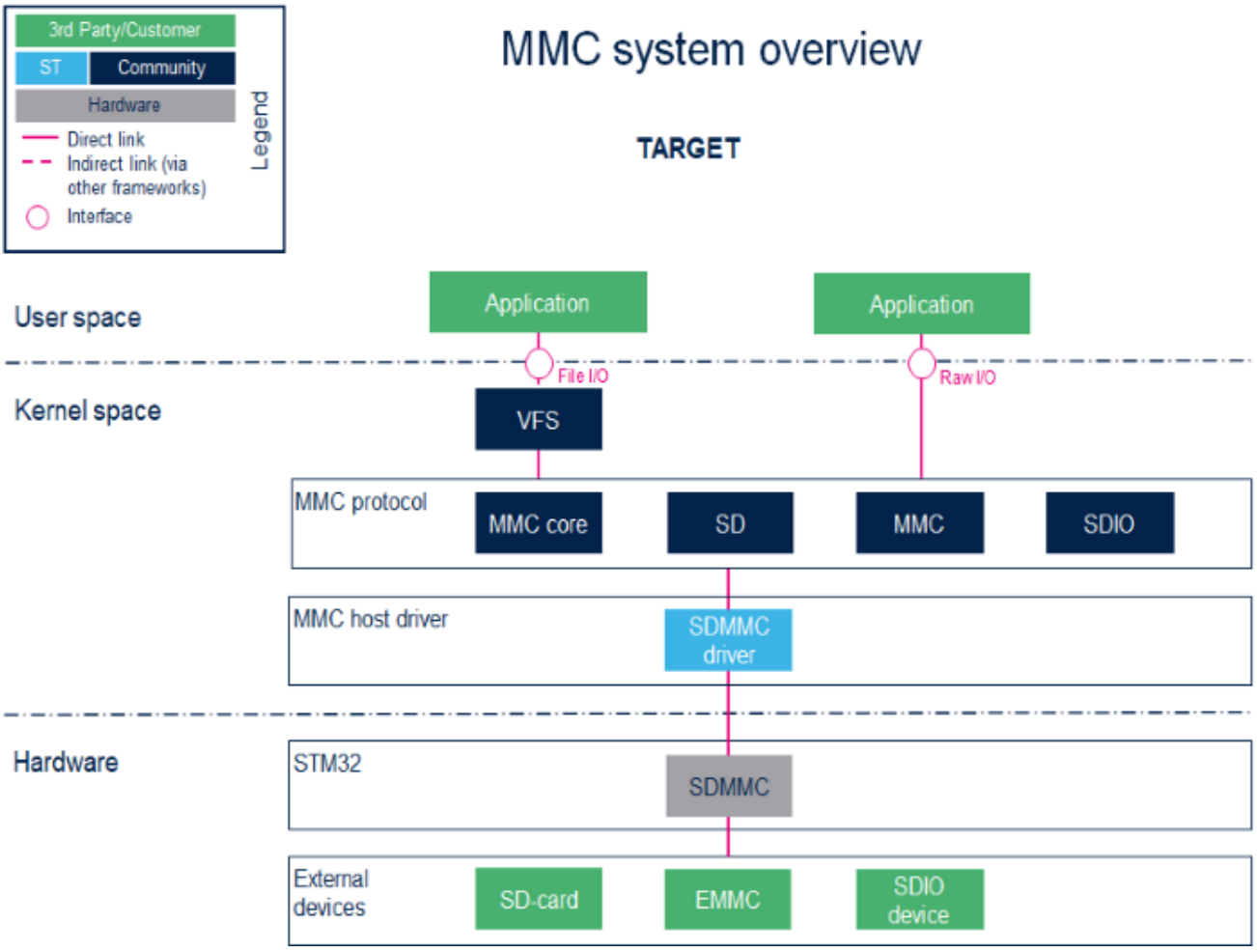
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 31.03.2021 - 08:47 / Revision: 26.03.2021 - 08:44

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

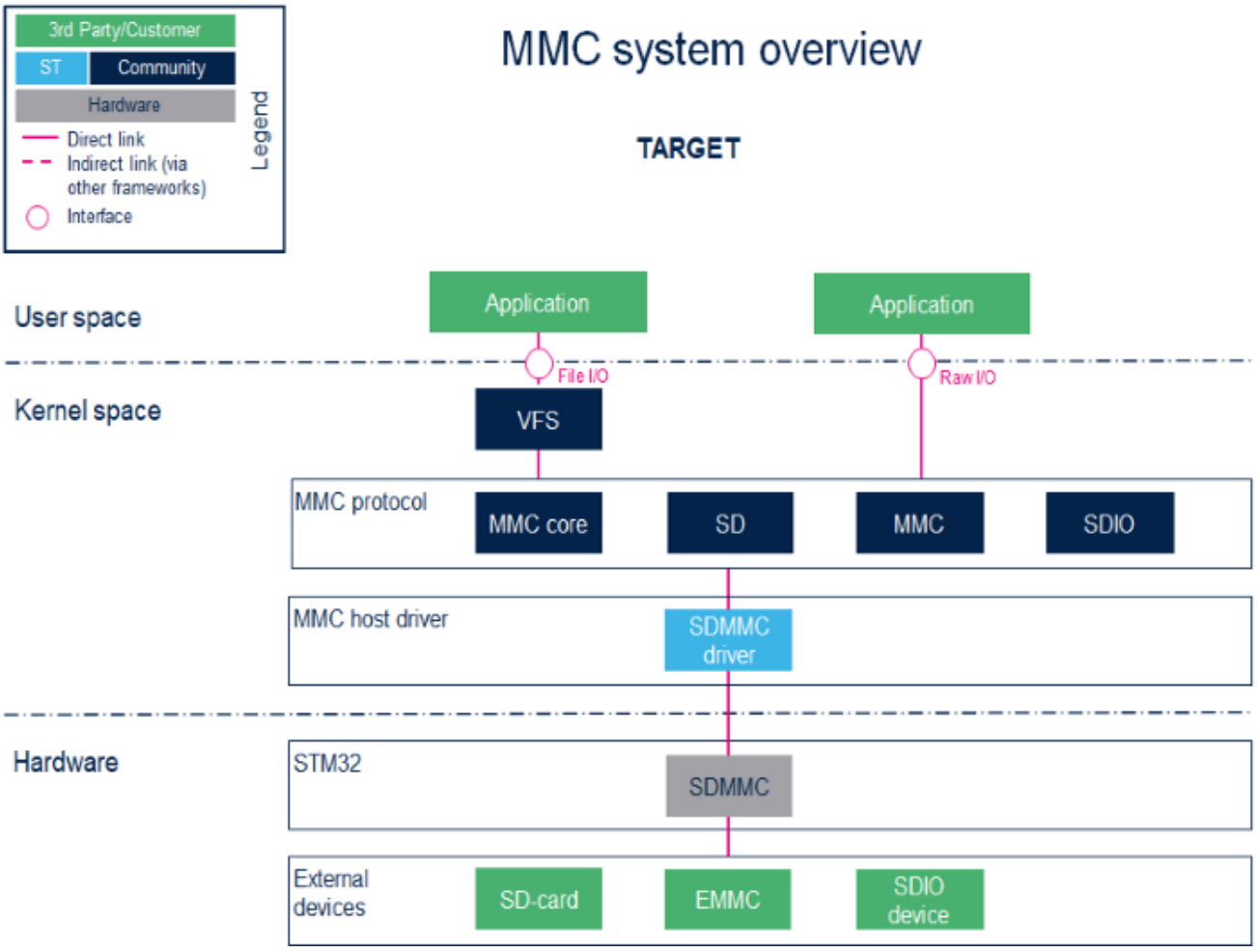
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 14.05.2020 - 07:28 / Revision: 14.05.2020 - 07:27

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

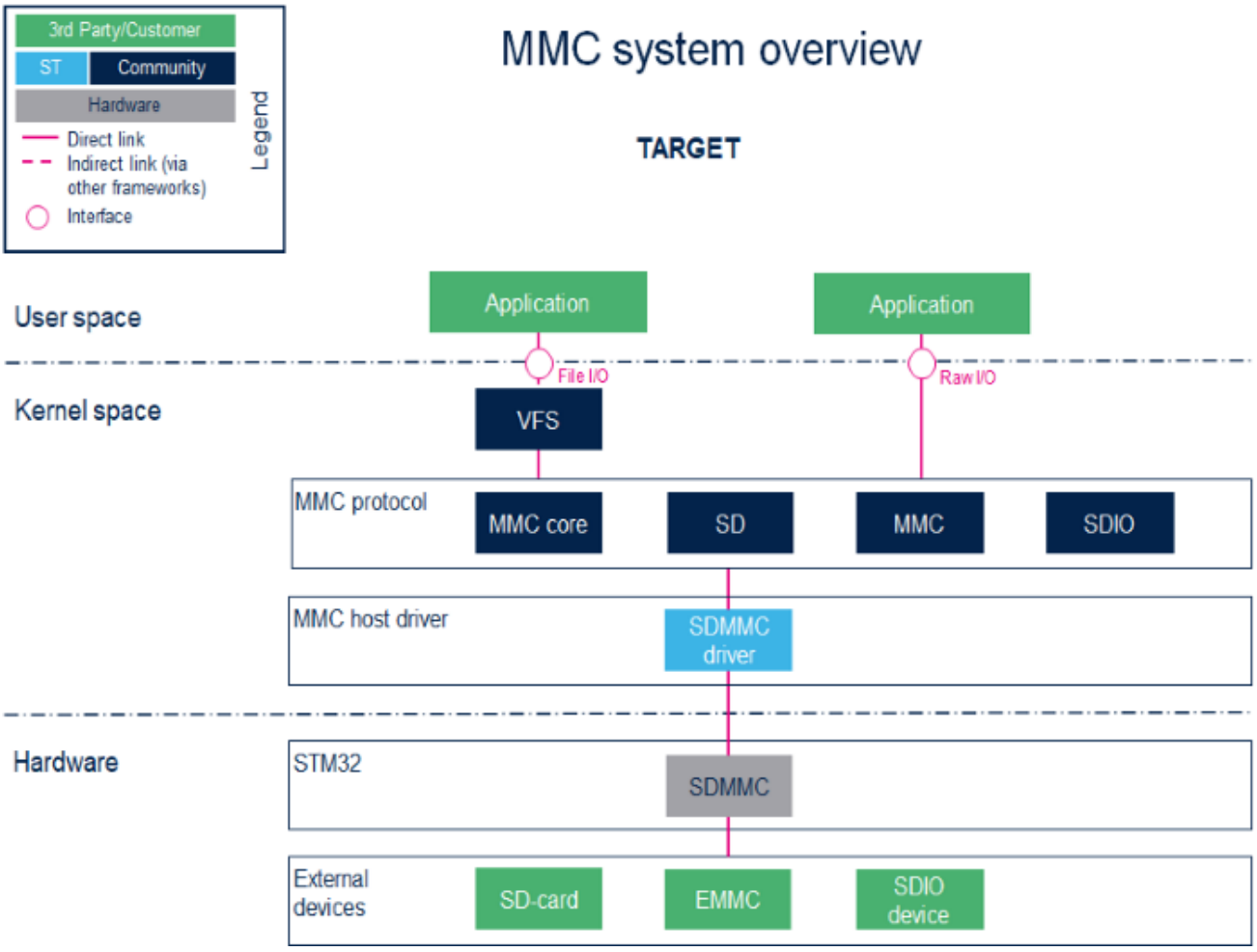
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 14.05.2020 - 07:13 / Revision: 14.05.2020 - 07:12

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

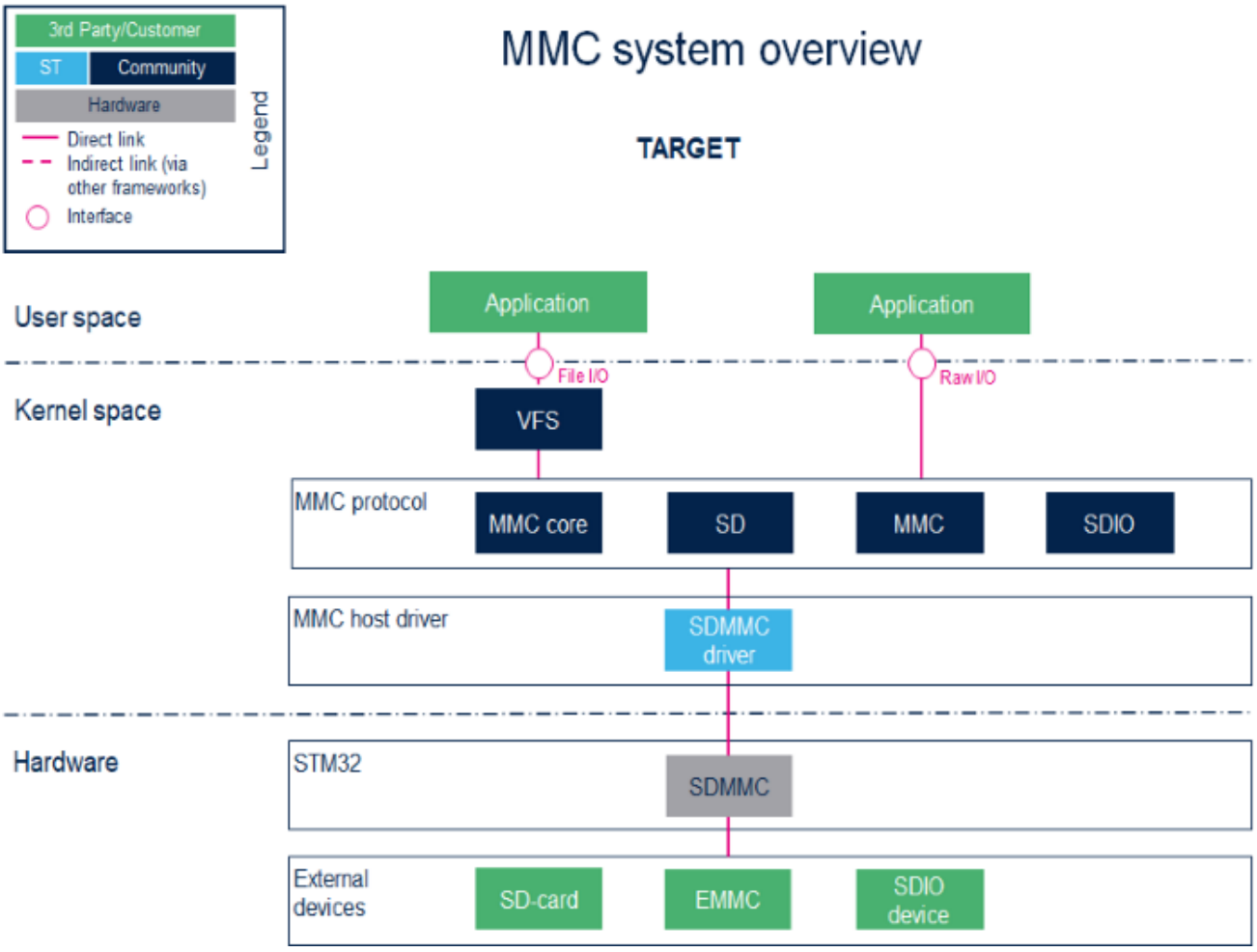
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 23.09.2020 - 13:22 / Revision: 12.06.2020 - 13:25

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

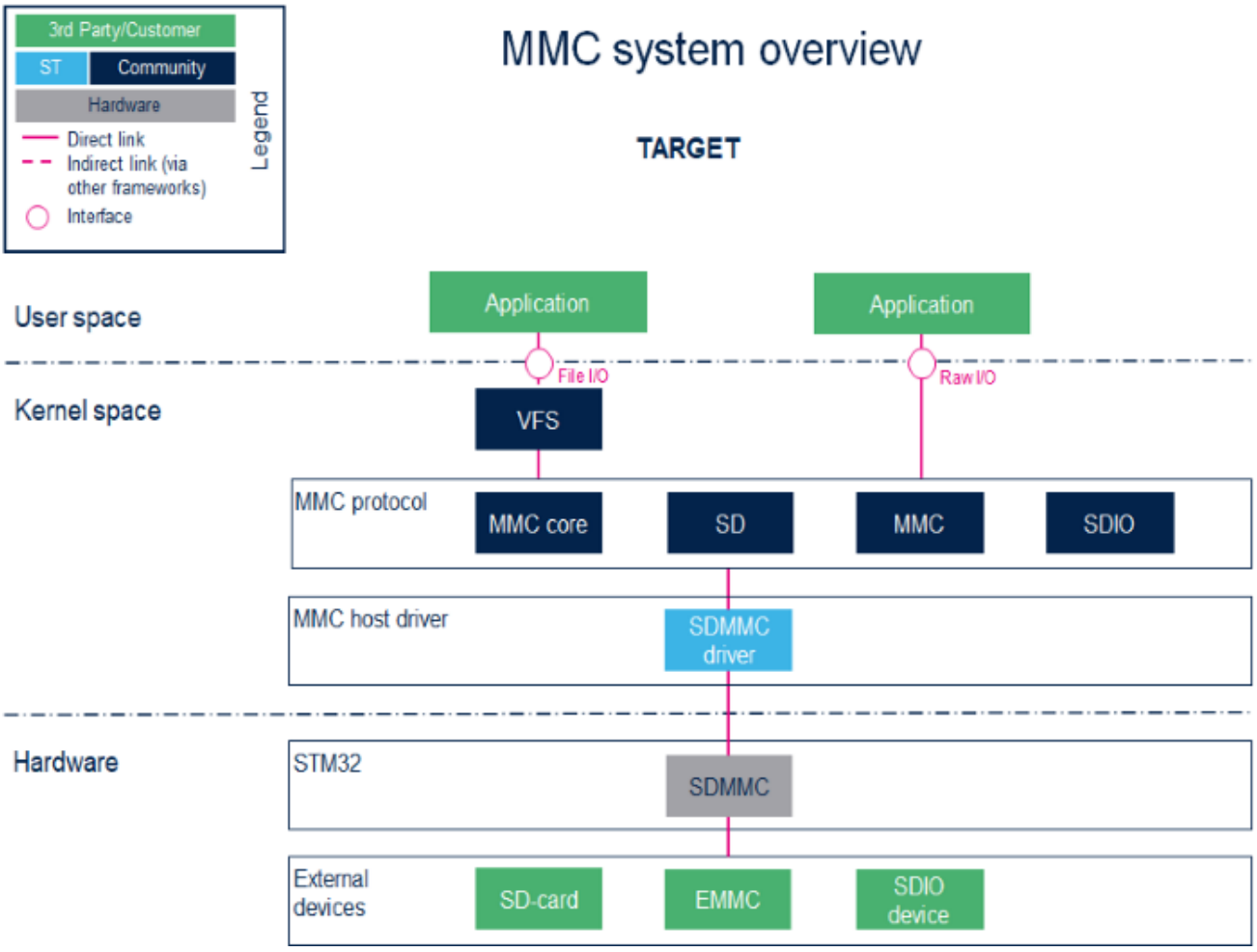
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification

Stable: 15.04.2020 - 08:28 / Revision: 15.04.2020 - 08:23

SUMMARY

The MMC (MultiMediaCard) / SD (secure digital) / SDIO (secure digital input/output) subsystem implements a standard Linux® host driver to interface with MMC / SD memory cards or SDIO cards.

Contents

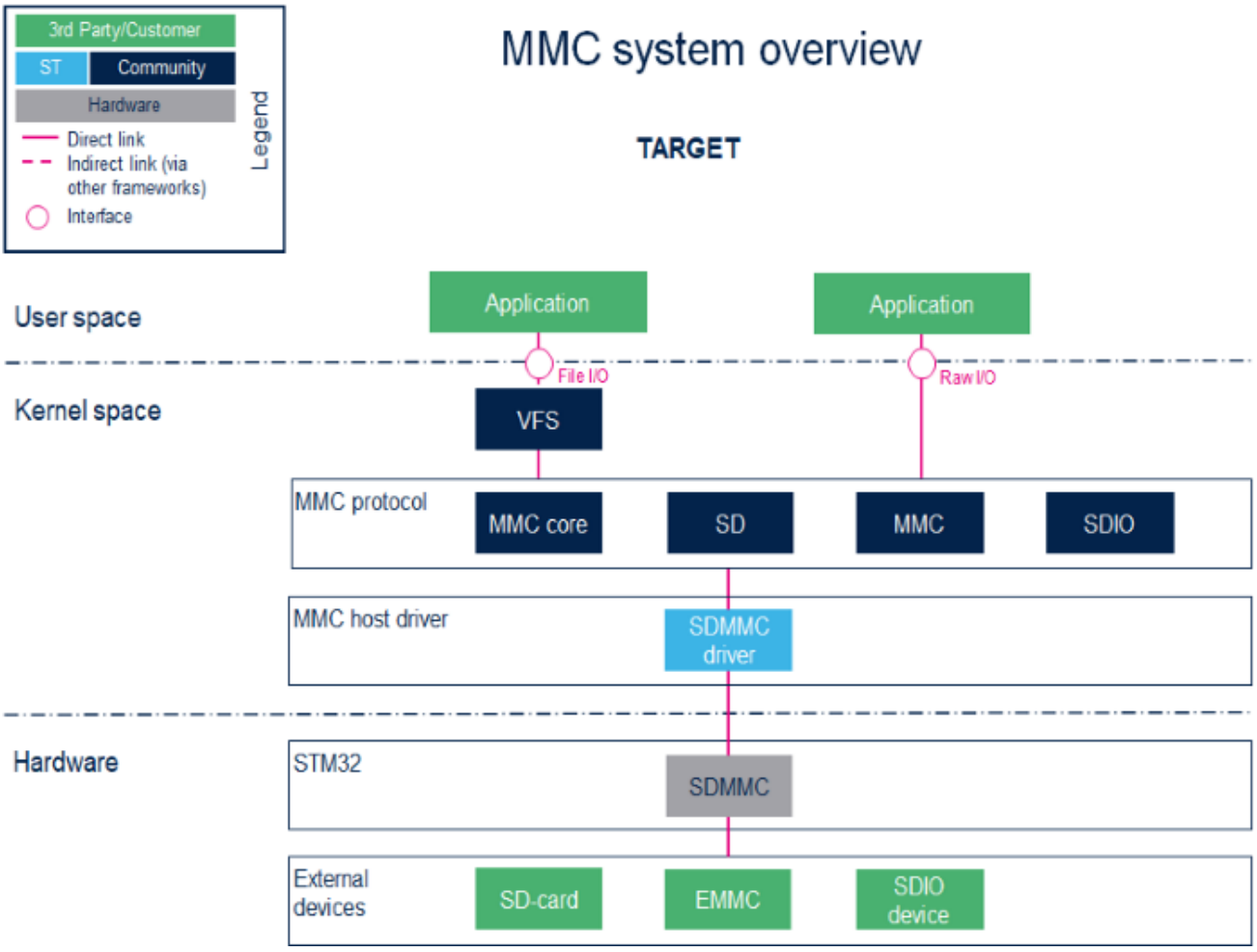
1 Framework purpose

The purpose of this article is to introduce the MMC Linux® subsystem (MMC / SD) by:

- providing general information
- describing the main components/stakeholders

The SDIO is addressed in the WLAN overview.

2 System overview



2.1 Component description

- User space applications handle **file I/O** management to view the card memory as a disk, whereas programs that perform **raw I/O** accesses see the memory as a block device[1].
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [2].

- **MMC core/SD/MMC/SDIO** (Kernel space)

The **MMC core** ensures compliance with MultiMediaCard (**MMC**)[3] / secure digital (**SD**)[4] / secure digital input/output (**SDIO**)[5].

- **SDMMC driver** (Kernel space) / **SDMMC** (hardware)

The **SDMMC driver** handles:

- the registers, the clock, the interrupt and the IDMA control.
- the communications over the bus based on command/response and data transfers.

Please refer to the SDMMC internal peripheral.

2.2 API description

The MMC core handles the file system read/write calls.

3 Configuration

3.1 Kernel configuration

The MMC framework is activated by default in ST deliveries. If a specific configuration is needed, this section indicates how the MMC framework can be activated/inactivated in the kernel.

The MMC framework can be activated in the kernel configuration via Linux® Menuconfig tool: Menuconfig or how to configure kernel

```
[*] Device Drivers
    [*] MMC/SD/SDIO card support
        <*> HW reset support for eMMC
        <*> Simple HW reset support for MMC
        <*> MMC block device driver
            (16) Number of minors per block device
        ...
        <*> ARM AMBA Multimedia Card Interface support
        [*] STMicroelectronics STM32 SDMMC Controller
```

3.2 Device tree configuration

DT configuration can be done thanks to STM32CubeMX.

Please refer to the SDMMC device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used with the MMC framework. Please refer to the EXT4 support through MMC.

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detailed information on each mmc device:

```
root:~# cat /sys/kernel/debug/mmc0/ios
clock:          50000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    2 (sd high-speed)
signal voltage: 0 (3.30 V)
driver type:    0 (driver type B)
```

5.2 How to trace

For details on dynamic trace usage, refer to How to use the kernel dynamic debug.

root:~# echo "file drivers/mmc/* +p" > /sys/kernel/debug/dynamic_debug/control

6 Source code location

The MMC framework is available here .

7 References

Please refer to the following links for a full description of the MMC framework:

- https://en.wikipedia.org/wiki/Device_file#Block_devices
- VFS
- MultiMediaCard, embedded MultiMediaCard specification
- Secure Digital, secure digital specification
- Secure Digital Input Output, Secure Digital Input Output specification