



---

## MMC overview



---

## Contents

---

1. MMC overview .....	3
2. How to support EXT4 through MMC .....	11
3. How to use the kernel dynamic debug .....	19
4. Menuconfig or how to configure kernel .....	27
5. SDMMC device tree configuration .....	35
6. SDMMC internal peripheral .....	43
7. STM32CubeMX .....	51
8. WLAN overview .....	59



A quality version of this page, approved on 15 April 2020, was based off this revision.

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

## Contents

1 Framework purpose .....	4
2 System overview .....	5
2.1 Component description .....	5
2.2 API description .....	6
3 Configuration .....	7
3.1 Kernel configuration .....	7
3.2 Device tree configuration .....	7
4 How to use the framework .....	8
5 How to trace and debug the framework .....	9
5.1 How to monitor .....	9
5.2 How to trace .....	9
6 Source code location .....	10
7 References .....	11



---

## 1 Framework purpose

---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	12
2 System overview .....	13
2.1 Component description .....	13
2.2 API description .....	14
3 Configuration .....	15
3.1 Kernel configuration .....	15
3.2 Device tree configuration .....	15
4 How to use the framework .....	16
5 How to trace and debug the framework .....	17
5.1 How to monitor .....	17
5.2 How to trace .....	17
6 Source code location .....	18
7 References .....	19



---

## 1 Framework purpose

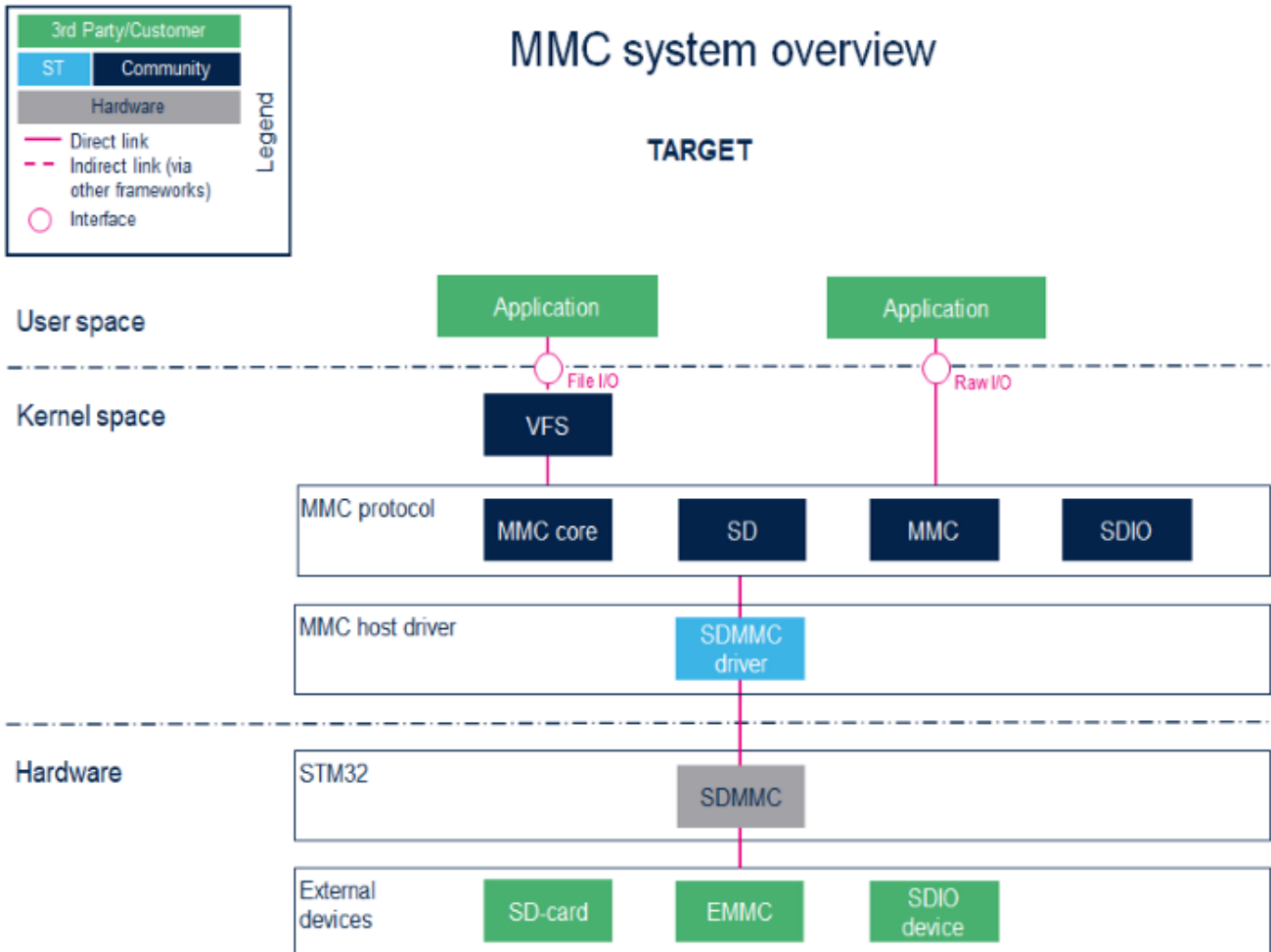
---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	20
2 System overview .....	21
2.1 Component description .....	21
2.2 API description .....	22
3 Configuration .....	23
3.1 Kernel configuration .....	23
3.2 Device tree configuration .....	23
4 How to use the framework .....	24
5 How to trace and debug the framework .....	25
5.1 How to monitor .....	25
5.2 How to trace .....	25
6 Source code location .....	26
7 References .....	27



---

## 1 Framework purpose

---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	28
2 System overview .....	29
2.1 Component description .....	29
2.2 API description .....	30
3 Configuration .....	31
3.1 Kernel configuration .....	31
3.2 Device tree configuration .....	31
4 How to use the framework .....	32
5 How to trace and debug the framework .....	33
5.1 How to monitor .....	33
5.2 How to trace .....	33
6 Source code location .....	34
7 References .....	35



---

## 1 Framework purpose

---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	36
2 System overview .....	37
2.1 Component description .....	37
2.2 API description .....	38
3 Configuration .....	39
3.1 Kernel configuration .....	39
3.2 Device tree configuration .....	39
4 How to use the framework .....	40
5 How to trace and debug the framework .....	41
5.1 How to monitor .....	41
5.2 How to trace .....	41
6 Source code location .....	42
7 References .....	43



---

## 1 Framework purpose

---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	44
2 System overview .....	45
2.1 Component description .....	45
2.2 API description .....	46
3 Configuration .....	47
3.1 Kernel configuration .....	47
3.2 Device tree configuration .....	47
4 How to use the framework .....	48
5 How to trace and debug the framework .....	49
5.1 How to monitor .....	49
5.2 How to trace .....	49
6 Source code location .....	50
7 References .....	51



---

## 1 Framework purpose

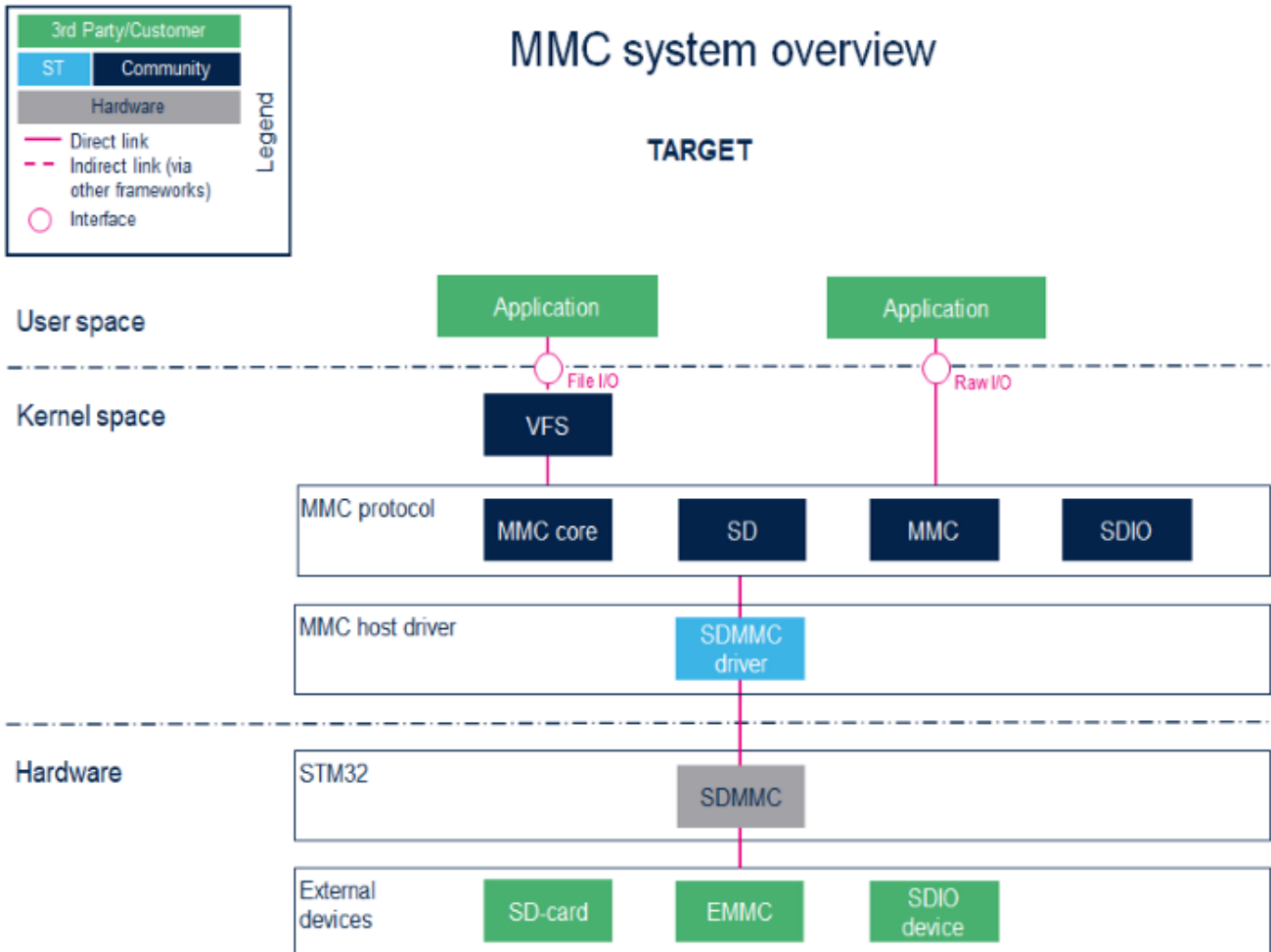
---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	52
2 System overview .....	53
2.1 Component description .....	53
2.2 API description .....	54
3 Configuration .....	55
3.1 Kernel configuration .....	55
3.2 Device tree configuration .....	55
4 How to use the framework .....	56
5 How to trace and debug the framework .....	57
5.1 How to monitor .....	57
5.2 How to trace .....	57
6 Source code location .....	58
7 References .....	59



---

## 1 Framework purpose

---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for eMMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

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

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

### Contents

1 Framework purpose .....	60
2 System overview .....	61
2.1 Component description .....	61
2.2 API description .....	62
3 Configuration .....	63
3.1 Kernel configuration .....	63
3.2 Device tree configuration .....	63
4 How to use the framework .....	64
5 How to trace and debug the framework .....	65
5.1 How to monitor .....	65
5.2 How to trace .....	65
6 Source code location .....	66
7 References .....	67



---

## 1 Framework purpose

---

The purpose of this article is to introduce the MMC Linux<sup>®</sup> 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<sup>[1]</sup>.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation<sup>[2]</sup>.

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

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

- **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<sup>®</sup> 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](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

MultimediaCard

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Secure digital

Virtual File System

Secure digital input/output

Application programming interface

SDIO is an SD-size card with extended input/output functions

former spelling for e•MMC ('e' in italic)

Device Tree

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)