MTD overview

Contents

1. MTD overview 3
2. FMC internal peripheral 12
3. QUADSPI internal peripheral 22
4. Menuconfig or how to configure kernel 32
5. STM32CubeMX 42
6. FMC device tree configuration 52
7. QUADSPI device tree configuration 62
8. How to support UBIFS through MTD 72
9. How to use the kernel dynamic debug 82

Stable: 14.05.2020 - 09:13 / Revision: 14.05.2020 - 09:10

Stable: 14.05.2020 - 09:13 / Revision: 14.05.2020 - 09:13

A quality version of this page, accepted on *14 May 2020*, was based off this revision.

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 4
2 System overview 5
2.1 Component description 5
2.2 API description 5
3 Configuration 6
3.1 Kernel configuration 6
3.1.1 SLC NAND Flash memory 6
3.1.2 SPI NOR/NAND Flash memory 6
3.2 Device tree configuration 6
3.2.1 NAND Flash memory 6
3.2.2 SPI NOR/NAND Flash memory 6
4 How to use the framework 7
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 To go further 11
8 References 12

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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

Stable: 19.11.2020 - 10:48 / Revision: 12.11.2020 - 09:10

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 14
2 System overview 15
2.1 Component description 15
2.2 API description 15
3 Configuration 16
3.1 Kernel configuration 16
3.1.1 SLC NAND Flash memory 16
3.1.2 SPI NOR/NAND Flash memory 16
3.2 Device tree configuration 16
3.2.1 NAND Flash memory 16
3.2.2 SPI NOR/NAND Flash memory 16

4 How to use the framework 17
5 How to trace and debug the framework 19
5.1 How to monitor 19
5.2 How to trace 19
6 Source code location 20
7 To go further 21
8 References 22

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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

Stable: 09.10.2019 - 12:44 / Revision: 16.09.2019 - 12:51

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 24
2 System overview 25
2.1 Component description 25
2.2 API description 25
3 Configuration 26
3.1 Kernel configuration 26
3.1.1 SLC NAND Flash memory 26
3.1.2 SPI NOR/NAND Flash memory 26
3.2 Device tree configuration 26
3.2.1 NAND Flash memory 26
3.2.2 SPI NOR/NAND Flash memory 26

4 How to use the framework 27
5 How to trace and debug the framework 29
5.1 How to monitor 29
5.2 How to trace 29
6 Source code location 30
7 To go further 31
8 References 32

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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

Stable: Not stable / Revision: 19.01.2021 - 10:34

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 34
2 System overview 35
2.1 Component description 35
2.2 API description 35
3 Configuration 36
3.1 Kernel configuration 36
3.1.1 SLC NAND Flash memory 36
3.1.2 SPI NOR/NAND Flash memory 36
3.2 Device tree configuration 36
3.2.1 NAND Flash memory 36
3.2.2 SPI NOR/NAND Flash memory 36

4 How to use the framework 37
5 How to trace and debug the framework 39
5.1 How to monitor 39
5.2 How to trace 39
6 Source code location 40
7 To go further 41
8 References 42

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

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 Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 44
2 System overview 45
2.1 Component description 45
2.2 API description 45
3 Configuration 46
3.1 Kernel configuration 46
3.1.1 SLC NAND Flash memory 46
3.1.2 SPI NOR/NAND Flash memory 46
3.2 Device tree configuration 46
3.2.1 NAND Flash memory 46
3.2.2 SPI NOR/NAND Flash memory 46

4 How to use the framework 47
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 To go further 51
8 References 52

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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

Stable: 19.11.2020 - 10:48 / Revision: 13.11.2020 - 09:41

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 54
2 System overview 55
2.1 Component description 55
2.2 API description 55
3 Configuration 56
3.1 Kernel configuration 56
3.1.1 SLC NAND Flash memory 56
3.1.2 SPI NOR/NAND Flash memory 56
3.2 Device tree configuration 56
3.2.1 NAND Flash memory 56
3.2.2 SPI NOR/NAND Flash memory 56

4 How to use the framework 57
5 How to trace and debug the framework 59
5.1 How to monitor 59
5.2 How to trace 59
6 Source code location 60
7 To go further 61
8 References 62

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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

Stable: 14.05.2020 - 07:08 / Revision: 14.05.2020 - 07:07

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 64
2 System overview 65
2.1 Component description 65
2.2 API description 65
3 Configuration 66
3.1 Kernel configuration 66
3.1.1 SLC NAND Flash memory 66
3.1.2 SPI NOR/NAND Flash memory 66
3.2 Device tree configuration 66
3.2.1 NAND Flash memory 66
3.2.2 SPI NOR/NAND Flash memory 66

4 How to use the framework 67
5 How to trace and debug the framework 69
5.1 How to monitor 69
5.2 How to trace 69
6 Source code location 70
7 To go further 71
8 References 72

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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

Stable: 03.02.2020 - 08:05 / Revision: 03.02.2020 - 08:03

The Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 74
2 System overview 75
2.1 Component description 75
2.2 API description 75
3 Configuration 76
3.1 Kernel configuration 76
3.1.1 SLC NAND Flash memory 76
3.1.2 SPI NOR/NAND Flash memory 76
3.2 Device tree configuration 76
3.2.1 NAND Flash memory 76
3.2.2 SPI NOR/NAND Flash memory 76

4 How to use the framework 77
5 How to trace and debug the framework 79
5.1 How to monitor 79
5.2 How to trace 79
6 Source code location 80
7 To go further 81
8 References 82

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

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 Linux®MTD (Memory Technology Device) subsystem provides an abstraction layer for raw Flash memories. It makes it possible to use the same API when working with different Flash types and technologies, e.g. SLC NAND, SPI NOR, ...

Contents

1 Framework purpose 84
2 System overview 85
2.1 Component description 85
2.2 API description 85
3 Configuration 86
3.1 Kernel configuration 86
3.1.1 SLC NAND Flash memory 86
3.1.2 SPI NOR/NAND Flash memory 86
3.2 Device tree configuration 86
3.2.1 NAND Flash memory 86
3.2.2 SPI NOR/NAND Flash memory 86

4 How to use the framework 87
5 How to trace and debug the framework 89
5.1 How to monitor 89
5.2 How to trace 89
6 Source code location 90
7 To go further 91
8 References 92

1 Framework purpose

The purpose of this article is to introduce the MTD Linux subsystem:

- General information
- Main components/stakeholders
- How to use the MTD API

2 System overview

File:MTD v1.1.0.png

2.1 Component description

- User space applications that perform **file I/O** need to view the Flash memory as if it was a disk, whereas programs that wish to accomplish **raw I/O** access the memory as if it was a character device.
- **VFS** (Kernel space)

Virtual File System. Please refer to the VFS documentation [1].

- **mtdchar** (Kernel space)

Usually referred to as /dev/mtdX. For MTD character devices, please refer to the MTD overview documentation [2].

- **mtdblock** (Kernel space)

Usually referred to as /dev/mtdblockX. Do not use mtdblock unless you know exactly what you are doing. For MTD block devices, please refer to the MTD block documentation [3].

- **JFFS2** (Kernel space)

Journally Flash File System. Please refer to the MTD JFFS2 documentation [4].

- **UBI** (Kernel space)

Unsorted Block Images. Please refer to the MTD UBI documentation [5].

- **UBIFS** (Kernel space)

UBI File System. Please refer to the MTD UBIFS documentation [6].

- **MTD core** (Kernel space)

The MTD core provides an abstraction layer for raw Flash memories.

- **Raw NAND subsystem** (Kernel space)

The Raw NAND protocol is used in the MTD subsystem for interfacing NAND Flash memories.

- **SPI-MEM subsystem** (Kernel space)

The SPI-MEM protocol is used in the MTD subsystem for interfacing all kinds of SPI memories (NORs, NANDs)

- **SPI-NAND subsystem** (Kernel space)

The SPI-NAND protocol is used in the MTD subsystem for interfacing SPI NAND Flash memories.

- **SPI-NOR subsystem** (Kernel space)

The SPI-NOR protocol is used in the MTD subsystem for interfacing SPI NOR Flash memories.

- **FMC driver** (Kernel space) / **FMC** (Hardware)

Please refer to the FMC internal peripheral.

- **QUADSPI driver** (Kernel space) / **QUADSPI** (Hardware)

Please refer to the QUADSPI internal peripheral.

2.2 API description

For the Linux MTD API description, please refer to the MTD API documentation [7].

3 Configuration

3.1 Kernel configuration

MTD is activated by default in ST deliveries. Nevertheless, if a specific configuration is needed, this section indicates how MTD can be activated/deactivated in the kernel.

Activate MTD in the kernel configuration with the Linux Menuconfig tool: Menuconfig or how to configure kernel.

3.1.1 SLC NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        <*> RAW/Parallel NAND Device Support --->
            <*> Support for NAND controller on STM32MP Socs.
```

3.1.2 SPI NOR/NAND Flash memory

```
[*] Device Drivers --->
    <*> Memory Technology Device (MTD) support --->
        Self-contained MTD device drivers --->
            <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*> SPI NAND device Support
        <*> SPI-NOR device support
    <*> SPI support --->
        -*- SPI memory extension
        <*> STMicroelectronics STM32 QUAD SPI controller
```

3.2 Device tree configuration

The DT configuration can be done thanks to STM32CubeMX.

3.2.1 NAND Flash memory

Please refer to the FMC device tree configuration.

3.2.2 SPI NOR/NAND Flash memory

Please refer to the QUADSPI device tree configuration.

4 How to use the framework

A file system, which handles read/write/erase operations, can be used over the MTD Framework. Please refer to the UBIFS support through MTD.

You can also interact with the MTD subsystem using the MTD utilities. The MTD utilities[8] are a set of tools that can be used to perform operations on Flash memories through the MTD character interface.

The most common utilities used are:

- mtdinfo
- flash_erase
- flashcp
- nandwrite
- nanddump

```
root:~# mtdinfo -a
Count of MTD devices:           9
Present MTD devices:            mtd0, mtd1, mtd2, mtd3, mtd4, mtd5, mtd6, mtd7, mtd8
Sysfs interface supported:      yes

mtd0
Name:                           fsbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:0
Bad blocks are allowed:         true
Device is writable:             true

mtd1
Name:                           ssbl
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          8 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:2
Bad blocks are allowed:         true
Device is writable:             true

mtd2
Name:                           UBI
Type:                           nand
Eraseblock size:                262144 bytes, 256.0 KiB
Amount of eraseblocks:          4078 (1069023232 bytes, 1019.5 MiB)
Minimum input/output unit size: 4096 bytes
Sub-page size:                  4096 bytes
OOB size:                       224 bytes
Character device major/minor:   90:4
Bad blocks are allowed:         true
Device is writable:             true

mtd3
Name:                           fsbl1
```

```
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:6
Bad blocks are allowed:         false
Device is writable:             true

mtd4
Name:                           fsbl2
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:8
Bad blocks are allowed:         false
Device is writable:             true

mtd5
Name:                           ssbl
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          32 (2097152 bytes, 2.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:10
Bad blocks are allowed:         false
Device is writable:             true

mtd6
Name:                           logo
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          4 (262144 bytes, 256.0 KiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:12
Bad blocks are allowed:         false
Device is writable:             true

mtd7
Name:                           nor_user
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          980 (64225280 bytes, 61.2 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:14
Bad blocks are allowed:         false
Device is writable:             true

mtd8
Name:                           58003000.qspi
Type:                           nor
Eraseblock size:                65536 bytes, 64.0 KiB
Amount of eraseblocks:          1024 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:16
Bad blocks are allowed:         false
Device is writable:             true
```

5 How to trace and debug the framework

5.1 How to monitor

The sysfs interface provides detail information on each mtd device.

```
root:~# cat /sys/class/mtd/mtd0/name
fsbl
root:~# cat /sys/class/mtd/mtd0/type
nand
root:~# cat /sys/class/mtd/mtd0/erasesize
262144
root:~# cat /sys/class/mtd/mtd0/ecc_strength
8
root:~# cat /sys/class/mtd/mtd0/bad_blocks
0
root:~# cat /sys/class/mtd/mtd0/ecc_failures
0
```

5.2 How to trace

A detail dynamic trace is available here How to use the kernel dynamic debug.

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

6 Source code location

The MTD framework is here .

7 To go further

Please refer to the MTD FAQs documentation [9].

8 References

Please refer to the following links for full description:

- VFS
- MTD overview
- MTD block
- MTD JFFS2
- MTD UBI
- MTD UBIFS
- MTD API
- MTD utils
- MTD FAQs

Memory Technology Device

Application programming interface

Single-Level Cell is a kind of NAND flash

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Virtual File System

Device Tree

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