OTP

API 参考

文档版本　V2.0

发布日期　2025-08-20

前 言

概述

OTP 全称为一次性编程存储器（One-Time Programmable），是一种只能写入一次数据的固态存储器。

说明

未经特殊说明，以 XM7606V13 指代 XM7606 系列所有芯片。以 XM7206V11A 指代 XM7206 系列所有芯片。

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。

符号	说明
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2025-04-01	V1.0	首次发布。
2025-08-20	V2.0	更新 XM7206V11A 芯片支持的 OTP 烧写接口、数据类型、proc 信息说明。

目 录

1 概述

1.1 概述

OTP 全称为一次性编程存储器（One-Time Programmable），是一种只能写入一次数据的固态存储器。提供特定标志位烧写、root key 的 sha256 数据烧写、aes key 及 ai key 的烧写、用户自定义区烧写及读取功能。

1.2 使用流程

1.2.1 烧写数据到模拟 OTP 区

场景说明

该功能用于烧写数据到模拟 OTP 区域，用于查看烧写的数据是否与预期的一致，防止烧写错误数据。确认数据烧写无误后，可调用接口烧写数据到真实 OTP 区域。

调用流程

烧写数据到模拟 OTP 区的过程如下：

步骤 1 OTP 模块初始化。调用接口 xmedia_otp_init 完成。

步骤 2 调用 OTP 烧写、读取接口。如 xmedia_otp_user_data_set。

步骤 3 OTP 模块去初始化。调用接口 xmedia_otp_exit 完成。

----结束

注意事项

- 烧写数据到模拟 OTP 区域的流程中，不能调用 xmedia_otp_program_enable!
- 烧写接口只支持调用一次，若烧写的数据为全 0，则可以多次调用；每次加载 CIPHER 模块的 ko 后，模拟 OTP 区域中的数据将清零。
- 烧写数据到模拟 OTP 区后，可通过 echo otp_program_enable=0x0 > /proc/umap/otp;cat /proc/umap/otp 指令查看模拟 OTP 区域的数据烧写情况。
- 若想读取模拟 OTP 用户自定义区中的数据，将步骤 2 中的接口替换成 xmedia_otp_user_data_get 即可。
- 烧写数据到模拟 OTP 区只起调试作用（调用相同接口、传入相同参数，烧写到模拟 OTP 区的数据与烧写到实际 OTP 区的数据是完全一样的，可通过 proc 调试信息查看模拟 OTP 区中的全部内容），模拟 OTP 没有实际 OTP 对应的功能。
- 不支持在多进程或多线程中进行 OTP 的读、写操作。

1.2.2 烧写数据到真实 OTP 区

场景说明

用于烧写数据到真实的 OTP 区域中。

调用流程

烧写数据到真实的 OTP 区域过程如下：

步骤 1　OTP 模块初始化。调用接口 xmedia_otp_init 完成。

步骤 2　使能真实烧写功能。调用 xmedia_otp_program_enable 完成。

步骤 3　调用 OTP 烧写、读取接口。如 xmedia_otp_user_data_set。

步骤 4　OTP 模块去初始化。调用接口 xmedia_otp_exit 完成。

----结束

注意事项

- 使能真实烧写功能后，烧写接口只能调用一次!
- 烧写数据到真实 OTP 区后，可通过 echo otp_program_enable=0x1 > /proc/umap/otp;cat /proc/umap/otp 指令查看真实 OTP 区域的数据烧写情况。需要特别注意的是，执行 echo otp_program_enable=0x1 > /proc/umap/otp 指令后

已使能了 OTP 真实烧写功能，此时可通过 OTP 的 proc 节点，使用 echo 命令更改真实 OTP 用户自定义区域中的数据，请谨慎调用 echo 命令。echo 命令的详细使用说明请参考 proc 调试信息章节。

- 若想读取真实 OTP 用户自定义区域中的数据，将步骤 3 中的接口替换成 xmedia_otp_user_data_get 即可。
- 烧写 OTP 后，重启单板后生效，proc 信息也会在重启单板后更新。
- 不支持在多进程或多线程中进行 OTP 的读、写操作。

2 API 参考

OTP 提供以下 API：

- xmedia_otp_init：初始化 OTP 模块。
- xmedia_otp_exit：去初始化 OTP 模块。
- xmedia_otp_program_enable：使能 otp 烧写功能。
- xmedia_otp_secure_boot_enable：使能安全启动。
- xmedia_otp_ddr_ca_enable：使能 DDR 加扰。
- xmedia_otp_jtag_disable：关闭 jtag 接口。
- xmedia_otp_bootrom_debug_disable：关闭 bootrom 串口打印。
- xmedia_otp_root_key_sha256_set：烧写安全启动使用的 RSA4096 PSS 验签算法公钥的 SHA256 值。
- xmedia_otp_aes256_key_set：烧写 CIPHER 模块使用的硬件 key。
- xmedia_otp_ai_key_set：烧写 AI key。
- xmedia_otp_user_data_set：烧写用户自定义区指定偏移的区域。
- xmedia_otp_user_data_get：读取用户自定义区指定偏移的区域的值。
- xmedia_otp_spi_nand_mode_set：设置 spi nand flash 启动模式为 4 线。
- xmedia_otp_uart_burn_mode_sel：关闭、打开 UART 镜像烧写。
- xmedia_otp_fmc_freq_sel：设置 bootrom 阶段 spi nor、spi nand 的时钟频率。

xmedia_otp_init

【描述】

初始化 OTP 模块。

【语法】

```
xmedia_s32 xmedia_otp_init(xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_OPEN_FAILED	获取文件描述符失败。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

调用本接口后会关闭烧写实际 OTP 区数据的功能。

调用其他烧写、读取接口前必须调用 xmedia_otp_init。

【举例】

无。

xmedia_otp_exit

【描述】

去初始化 OTP 模块。

【语法】

```
xmedia_s32 xmedia_otp_exit(xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_OPEN_FAILED	获取文件描述符失败。

返回值	描述
XMEDIA_ERRCODE_CLOSE_FAILED	关闭文件描述符失败。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

调用本接口后会关闭烧写实际 OTP 区数据的功能。

【举例】

无。

xmedia_otp_program_enable

【描述】

使能 OTP 烧写功能，使能后，其余接口会操作真实的 OTP 区域；未使能时，其余接口操作的是模拟 OTP 区域。

【语法】

```
xmedia_s32 xmedia_otp_program_enable (xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

无。

【举例】

无。

xmedia_otp_secure_boot_enable

【描述】

使能安全启动。

【语法】

```
xmedia_s32 xmedia_otp_secure_boot_enable (xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

需要特别注意，需要先调用 xmedia_otp_root_key_sha256_set 接口并写入正确的值，再调用安全启动使能接口，只调用该接口会使芯片无法正常启动!!!

【举例】

无。

xmedia_otp_ddr_ca_enable

【描述】

使能 DDR 加扰。

【语法】

```
xmedia_s32 xmedia_otp_ddr_ca_enable (xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

无。

【举例】

无。

xmedia_otp_jtag_disable

【描述】

关闭 jtag 接口。

【语法】

```
xmedia_s32 xmedia_otp_jtag_disable (xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

无。

【举例】

无。

xmedia_otp_bootrom_debug_disable

【描述】

关闭 bootrom 串口打印。

【语法】

```
xmedia_s32 xmedia_otp_bootrom_debug_disable (xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。

返回值	描述
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

无。

【举例】

无。

xmedia_otp_root_key_sha256_set

【描述】

烧写安全启动使用的 RSA4096 PSS 验签算法公钥的 SHA256 值。

【语法】

```
xmedia_s32 xmedia_otp_root_key_sha256_set(xmedia_u8 *sha256_data, xmedia_u32 data_len);
```

【参数】

参数名称	描述	输入/输出
sha256_data	SHA256 数据起始地址。	输入
data_len	SHA256 数据长度，单位 Byte。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。
XMEDIA_ERRCODE_NULL_PTR	空指针。

返回值	描述
XMEDIA_ERRCODE_INVALID_PARAM	传入参数不合法。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

- 烧写前请确保输入数据的正确性，若烧写错误数据、不使能安全启动情况下，单板可正常启动；若烧写错误数据并使能安全启动，单板将无法使用!!!
- 假设安全启动使用的 RSA4096 PSS 验签算法公钥的 SHA256 值为：00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff,则需初始化数组为下列形式：

```
xmedia_u8 arr[SHA256_BYTE_LEN] =
{
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
    0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff,
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
    0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff
};
```

再调用 xmedia_otp_root_key_sha256_set(arr, SHA256_BYTE_LEN)进行数据烧写。若烧写到实际 OTP 区，因为验签算法公钥的 SHA256 值对应 OTP 不可回读，所以烧写的数据不可通过 proc 信息查看；若烧写到模拟 OTP 区，可通过 proc 信息查看具体内容，对应区域显示的内容理论上与烧写的数据一致，也为 00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff。

【举例】

无。

xmedia_otp_aes256_key_set

【描述】

烧写 CIPHER 模块使用的硬件 key。

【语法】

```
xmedia_s32 xmedia_otp_aes256_key_set(xmedia_u8 *key, xmedia_u32 key_len,
xmedia_otp_aes_key_index key_index);
```

【参数】

参数名称	描述	输入/输出
key	key 数据起始地址。	输入
key_len	key 数据长度，单位 Byte。	输入
key_index	key 标志。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。
XMEDIA_ERRCODE_NULL_PTR	空指针。
XMEDIA_ERRCODE_INVALID_PARAM	传入参数不合法。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

- AES KEY0 用于安全启动。AES KEY1~AES KEY3 由 cipher 模块使用，用于解密传入的密文 key。
- 假设需要烧写的 AES KEY1 值为：00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff,则需初始化数组为下列形式：

```
u8 arr[AES256_KEY_BYTE_LEN] =
{
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
    0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff,
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
    0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff
};
```

再调用 xmedia_otp_aes256_key_set (arr, AES256_KEY_BYTE_LEN, XMEDIA_OTP_AES256_KEY1)进行数据烧写。若烧写到实际 OTP 区，因为 AES KEY 的值是不可回读的，所以烧写的数据不可通过 proc 信息查看；若烧写到模拟 OTP 区，可通过 proc 信息查看具体内容，对应区域显示的内容理论上与烧写的数据一致，也为 00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff。

【举例】

无。

xmedia_otp_ai_key_set

【描述】

烧写 AI key。

【语法】

```
xmedia_s32 xmedia_otp_ai_key_set(xmedia_u8 *key, xmedia_u32 key_len);
```

【参数】

参数名称	描述	输入/输出
key	key 数据起始地址。	输入
key_len	key 数据长度，单位 Byte。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。
XMEDIA_ERRCODE_NULL_PTR	空指针。
XMEDIA_ERRCODE_INVALID_PARAM	传入参数不合法。

【芯片差异】

芯片类型	是否支持本接口
XM7606V13	支持
XM7206V11A	不支持

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

假设需要烧写的 AI key 值为：112233445566778899aa,则需初始化数组为下列形式：

u8 arr[AI_KEY_BYTE_LEN] =

{

0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xaa

};

再调用 xmedia_otp_ai_key_set (arr, AI_KEY_BYTE_LEN)进行数据烧写。若烧写到实际 OTP 区，因为 AI KEY 的值是不可回读的，所以烧写的数据不可通过 proc 信息查看；若烧写到模拟 OTP 区，可通过 proc 信息查看具体内容，对应区域显示的内容理论上与烧写的数据一致，也为 112233445566778899aa。

【举例】

无。

xmedia_otp_user_data_set

【描述】

烧写用户自定义区指定偏移的区域。

【语法】

```
xmedia_s32 xmedia_otp_user_data_set(xmedia_u32 data, xmedia_u32 offset);
```

【参数】

参数名称	描述	输入/输出
data	要写入 otp 中的值。	输入
offset	用户自定义区中的偏移，以 32bit 为单位。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。
XMEDIA_ERRCODE_INVALID_PARAM	传入参数不合法。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

- XM7606V13 芯片用户自定义区 OTP 特性如下：

  用户自定义区 OTP 读写粒度为 32bit，每次烧写 32bit，写后会自动 lock，lock 后不能再更改。offset 从 0 开始，因用户自定义区共 28Kbits（使能了 OTP 的功能安全，则只有 27Kbits），所以 offset 不能超过 OTP_USER_DATA_LEN_MAX。

  offset 与用户自定区 OTP 对应关系如下：

offset	OTP bit 范围
0	0~31
1	32~63
…	…
895	28640~28671

- XM7206V11A 芯片用户自定义区 OTP 特性如下：

  用户自定义区 OTP 读写粒度为 32bit，每次烧写 32bit，无 lock，可多次烧写，直至所有 bit 烧写为 1。offset 从 0 开始，因用户自定义区共 1Kbits，所以 offset 不能超过 OTP_USER_DATA_LEN_MAX。

  offset 与用户自定区 OTP 对应关系如下：

offset	OTP bit 范围
0	0~31

1	32~63
…	…
31	992~1023

【举例】

无。

xmedia_otp_user_data_get

【描述】

读取用户自定义区指定偏移区域的值。

【语法】

```
xmedia_s32 xmedia_otp_user_data_get(xmedia_u32 *data, xmedia_u32 offset);
```

【参数】

参数名称	描述	输入/输出
data	otp 中指定偏移区域的值。	输出
offset	用户自定义区中的偏移，以 32bit 为单位。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。
XMEDIA_ERRCODE_INVALID_PARAM	传入参数不合法。

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

offset 值的有效范围与 xmedia_otp_user_data_set 中的一致。

【举例】

无。

xmedia_otp_spi_nand_mode_set

【描述】

设置 spi nand flash 启动模式为 4 线。

【语法】

```
xmedia_s32 xmedia_otp_spi_nand_mode_set (xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【芯片差异】

芯片类型	是否支持本接口
XM7606V13	不支持
XM7206V11A	支持

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

默认为 1 线。

【举例】

无。

xmedia_otp_uart_burn_mode_sel

【描述】

关闭、打开 UART 镜像烧写。

【语法】

```
xmedia_s32 xmedia_otp_uart_burn_mode_sel (xmedia_otp_uart_burn_mode burn_mode);
```

【参数】

参数名称	描述	输入/输出
burn_mode	uart 烧写模式，关闭或打开 UART 镜像烧写。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【芯片差异】

芯片类型	是否支持本接口
XM7606V13	不支持
XM7206V11A	支持

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

默认 UART 可以进行镜像的烧写，关闭 UART 烧写后可以再次打开 UART 烧写，再次打开 UART 烧写后不能再更改。

【举例】

无。

xmedia_otp_fmc_freq_sel

【描述】

设置 bootrom 阶段 spi nor、spi nand 的时钟频率。

【语法】

```
xmedia_s32 xmedia_otp_fmc_freq_sel (xmedia_otp_fmc_freq frequency);
```

【参数】

参数名称	描述	输入/输出
frequency	FMC 时钟频率。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_INIT	未调用 xmedia_otp_init 接口进行初始化。
XMEDIA_ERRCODE_NOT_PERMITTED	otp 位已配置过并锁定。

【芯片差异】

芯片类型	是否支持本接口
XM7606V13	不支持
XM7206V11A	支持

【需求】

- 头文件：xmedia_otp.h
- 库文件：libxmedia_cipher.a

【注意】

默认时钟频率为 12MHz，更改顺序只支持：默认频率->50MHz->75MHz->12MHz,更改顺序可以跳过某频率但是不可逆。

【举例】

无。

3 数据类型

相关数据类型、数据结构定义如下：

- SHA256_BYTE_LEN：sha256 算法输出数据长度，单位：Byte。
- AES256_KEY_BYTE_LEN：AES256 算法使用的 key 长度，单位：Byte。
- AI_KEY_BYTE_LEN：AI KEY 长度，单位：Byte。
- OTP_USER_DATA_LEN_MAX：用户自定义区长度，单位：word。
- xmedia_otp_aes_key_index：cipher 模块使用的 key 标记。
- xmedia_otp_uart_burn_mode：uart 镜像烧写模式。
- xmedia_otp_fmc_freq：FMC 时钟频率。

SHA256_BYTE_LEN

【说明】

sha256 算法输出数据长度，单位：Byte。

【定义】

```
#define     SHA256_BYTE_LEN                 32
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

xmedia_otp_root_key_sha256_set

AES256_KEY_BYTE_LEN

【说明】

AES256 算法使用的 key 长度，单位：Byte。

【定义】

```
#define      AES256_KEY_BYTE_LEN          32
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

xmedia_otp_aes256_key_set

AI_KEY_BYTE_LEN

【说明】

AI key 长度，单位：Byte。

【定义】

```
#define      AI_KEY_BYTE_LEN          10
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

xmedia_otp_ai_key_set

OTP_USER_DATA_LEN_MAX

【说明】

用户自定义区长度，单位：word。

【定义】

```
#define      OTP_USER_DATA_LEN_MAX          896
```

【成员】

无。

【注意事项】

- XM7606V13 芯片未使能 OTP 的功能安全时，OTP_USER_DATA_LEN_MAX 值为 896。
- XM7606V13 芯片使能 OTP 的功能安全时，OTP_USER_DATA_LEN_MAX 值为 864。
- XM7206V11A 芯片，OTP_USER_DATA_LEN_MAX 值为 32。

【相关数据类型及接口】

xmedia_otp_user_data_set

xmedia_otp_aes_key_index

【说明】

cipher 模块使用的 key 标记。

【定义】

```
typedef enum {
    XMEDIA_OTP_AES256_KEY0 = 0x0,
    XMEDIA_OTP_AES256_KEY1,
    XMEDIA_OTP_AES256_KEY2,
    XMEDIA_OTP_AES256_KEY3,
    XMEDIA_OTP_AES256_KEY_MAX
} xmedia_otp_aes_key_index;
```

【成员】

成员名称	描述
XMEDIA_OTP_AES256_KEY0	AES 逻辑 key0。只用于安全启动。key1~3 由 cipher 模块使用。
XMEDIA_OTP_AES256_KEY1	AES 逻辑 key1。
XMEDIA_OTP_AES256_KEY2	AES 逻辑 key2。
XMEDIA_OTP_AES256_KEY3	AES 逻辑 key3。

【相关数据类型及接口】

xmedia_otp_aes256_key_set

xmedia_otp_uart_burn_mode

【说明】

uart 镜像烧写模式。

【定义】

```
typedef enum {
    XMEDIA_OTP_UART_BURN_CLOSE = 0x0,
    XMEDIA_OTP_UART_BURN_REOPEN,
    XMEDIA_OTP_UART_BURN_MODE_MAX
} xmedia_otp_uart_burn_mode;
```

【成员】

成员名称	描述
XMEDIA_OTP_UART_BURN_CLOSE	关闭 uart 镜像烧写功能。
XMEDIA_OTP_UART_BURN_REOPEN	关闭 uart 镜像烧写功能后可再次打开 uart 镜像烧写功能。

【相关数据类型及接口】

xmedia_otp_uart_burn_mode_sel

xmedia_otp_fmc_freq

【说明】

FMC 时钟频率。FMC 是 flash memory controller 的缩写。

【定义】

```
typedef enum {
    XMEDIA_OTP_FMC_FREQ_50MHZ = 0x0,
    XMEDIA_OTP_FMC_FREQ_75MHZ,
    XMEDIA_OTP_FMC_FREQ_12MHZ,
    XMEDIA_OTP_FMC_FREQ_MAX
} xmedia_otp_fmc_freq;
```

【成员】

成员名称	描述
XMEDIA_OTP_FMC_FREQ_50MHZ	50MHz。

成员名称	描述
XMEDIA_OTP_FMC_FREQ_75MHZ	75MHz。
XMEDIA_OTP_FMC_FREQ_12MHZ	12MHz。

【相关数据类型及接口】

xmedia_otp_fmc_freq_sel

4 Proc 调试信息

4.1 XM7606V13 芯片 OTP 状态

在未使能真实烧写功能时，proc 信息显示的是模拟 OTP 区域中的数据。在通过 echo otp_program_enable=0x1 > /proc/umap/otp 指令使能真实烧写功能后，proc 信息显示的是真实 OTP 区域中的数据。

4.1.1 模拟 OTP 区域内容

模拟 OTP 区域内容如下：

```
- / # cat proc/umap/otp

-----OTP INFO-----------------------------------------------------------
otp_program_enable: 0x0
The proc info will show simulate otp info.

-----FLAG INFO----------------------------------------------------------
flag                value
secure_boot_en    0
ddr_ca_en           0
jtag_disable      0
boot_info_lv      0

-----KEY INFO-----------------------------------------------------------
key                  value
ai_key               :0000000000000000000
root_key_sha256 :0000000000000000000000000000000000000000000000000000000000000000
aes_key0             :0000000000000000000000000000000000000000000000000000000000000000
0
aes_key1             :0000000000000000000000000000000000000000000000000000000000000000
0
aes_key2             :0000000000000000000000000000000000000000000000000000000000000000
0
aes_key3             :0000000000000000000000000000000000000000000000000000000000000000
0
```

```
-----USER DATA INFO-----------------------------------------------------------
offset                                          value
0000:   12345678 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0008:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0010:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0018:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0020:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0028:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0030:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0038:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0040:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0048:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0050:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0058:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0060:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0068:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0070:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0078:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0080:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0088:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0090:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0098:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00a0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00a8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00b0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00b8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00c0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00c8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00d0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00d8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00e0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00e8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00f0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00f8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0100:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0108:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0110:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0118:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0120:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0128:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0130:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0138:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
0140:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0148:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0150:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0158:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0160:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0168:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0170:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0178:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0180:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0188:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0190:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0198:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01a0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01a8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01b0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01b8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01c0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01c8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01d0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01d8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01e0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01e8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01f0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01f8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0200:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0208:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0210:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0218:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0220:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0228:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0230:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0238:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0240:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0248:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0250:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0258:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0260:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0268:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0270:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0278:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0280:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0288:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0290:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
0298:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02a0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02a8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02b0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02b8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02c0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02c8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02d0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02d8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02e0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02e8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02f0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02f8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0300:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0308:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0310:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0318:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0320:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0328:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0330:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0338:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0340:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0348:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0350:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0358:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0360:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0368:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0370:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0378:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

-----SUPPORT COMMAND-------------------------------------------------------------
If you want to modify actual otp value, you must set otp_program_enable to be 0x1.
You can modify otp_program_enable by 'echo otp_program_enable=value > /proc/umap/otp', value
must be Hex, not support prefix of 0X.
You can modify otp value in user data by 'echo offset=value > /proc/umap/otp', offset and value
must be Hex, not support prefix of 0X.

 For example:
echo otp_program_enable=0x1 > /proc/umap/otp          : enable otp program.
echo otp_program_enable=0x0 > /proc/umap/otp          : enable otp simulation.
echo 0x0=0x12345678 > /proc/umap/otp                   : set the first word in otp to be
0x12345678.
-------------------------------------------------------------------------------
```

4.1.2 真实 OTP 区域内容

真实 OTP 区域内容如下：

```
#/ # cat proc/umap/otp

-----OTP INFO-----------------------------------------------------------------
otp_program_enable: 0x1
The proc info will show actual otp info.

-----FLAG INFO----------------------------------------------------------------
flag                value          lock
secure_boot_en   0              0
ddr_ca_en          0              0
jtag_disable      0              0
boot_info_lv      0              0

-----KEY INFO-----------------------------------------------------------------
key                  crc            lock
ai_key              0              0
root_key_sha256 0              0
aes_key0           0              0
aes_key1           0              0
aes_key2           0              0
aes_key3           0              0

-----USER DATA INFO-----------------------------------------------------------
offset                                          value
0000:   12345678 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0008:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0010:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0018:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0020:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0028:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0030:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0038:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0040:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0048:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0050:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0058:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0060:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0068:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0070:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
0078:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0080:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0088:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0090:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0098:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00a0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00a8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00b0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00b8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00c0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00c8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00d0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00d8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00e0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00e8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00f0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00f8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0100:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0108:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0110:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0118:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0120:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0128:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0130:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0138:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0140:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0148:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0150:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0158:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0160:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0168:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0170:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0178:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0180:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0188:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0190:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0198:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01a0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01a8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01b0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01b8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01c0:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01c8:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
01d0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01d8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01e0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01e8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01f0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01f8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0200:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0208:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0210:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0218:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0220:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0228:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0230:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0238:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0240:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0248:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0250:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0258:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0260:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0268:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0270:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0278:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0280:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0288:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0290:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0298:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02a0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02a8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02b0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02b8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02c0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02c8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02d0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02d8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02e0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02e8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02f0:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
02f8:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0300:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0308:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0310:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0318:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0320:  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
0328:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0330:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0338:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0340:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0348:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0350:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0358:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0360:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0368:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0370:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0378:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

-----SUPPORT COMMAND---------------------------------------------------------
If you want to modify actual otp value, you must set otp_program_enable to be 0x1.
You can modify otp_program_enable by 'echo otp_program_enable=value > /proc/umap/otp', value
must be Hex, not support prefix of 0X.
You can modify otp value in user data by 'echo offset=value > /proc/umap/otp', offset and value
must be Hex, not support prefix of 0X.

For example:
echo otp_program_enable=0x1 > /proc/umap/otp          : enable otp program.
echo otp_program_enable=0x0 > /proc/umap/otp          : enable otp simulation.
echo 0x0=0x12345678 > /proc/umap/otp                   : set the first word in otp to be
0x12345678.
-------------------------------------------------------------------------------
```

4.1.3 调试信息分析

【调试信息分析】

执行 cat /proc/umap/otp 指令会显示 OTP 区域中的数据烧写情况。根据 otp_program_enable 值的不同，proc 信息也会不同，当 otp_program_enable 为 0x0 时，显示的是模拟 OTP 区的数据信息，当 otp_program_enable 为 0x1 时，显示的是真实 OTP 区的数据信息。

可以通过指令 echo otp_program_enable=x > /proc/umap/otp 修改 otp_program_enable，x 只能为 0x0 或 0x1。

可以通过指令 echo offset=value > /proc/umap/otp 修改用户自定义区 OTP，offset 从 0 开始，小于 OTP_USER_DATA_LEN_MAX，单位为 32bit。offset 和 value 必须为 16 进制数，且只支持 0x 格式，不支持 0X 前缀。value 是对应的值。需要注意的是：用户自定义区 OTP 以 32bit 为 lock 单位，每 32bit 只能写一次，写后不能再更改。当

otp_program_enable 为 0x0 时，通过指令只能烧写模拟的用户自定义区 OTP；当 otp_program_enable 为 0x1 时，通过指令只能烧写真实的用户自定义区 OTP，此时请谨慎使用该指令。

模拟 OTP 区在每次 CIPHER 模块 ko 加载后，数据都会清零。

【参数说明】

以下是模拟 OTP 区域中的数据信息：

参数		描述
OTP INFO	otp_program_enable	真实烧写使能标志。为 0x0 时，未使能；为 0x1 时，使能
FLAG INFO	flag	OTP 中的标志位。secure_boot_en 是安全启动标志，ddr_ca_en 是 ddr 加扰标志，jtag_disable 是关闭 jtag 标志，boot_info_lv 是关闭 bootrom 串口打印标志
	value	标志位的值，为 1 代表使能了对应标志的功能
KEY INFO	key	OTP 中的 key，包括 root key 的 sha256 值
	value	对应 OTP 的值，16 进制显示
USER DATA INFO	offset	用户自定义区域中的偏移，从 0 开始，16 进制显示，单位为 word
	value	用户自定义区域中的内容，16 进制显示。如 proc 信息中 offset 为 0x0 处的值 12345678，对应值为 0x12345678

以下是真实 OTP 区域中的数据信息：

参数		描述
OTP INFO	otp_program_enable	真实烧写使能标志。为 0x0 时，未使能；为 0x1 时，使能
FLAG INFO	flag	OTP 中的标志位。secure_boot_en 是安全启动标志，ddr_ca_en 是 ddr 加扰标志，jtag_disable 是关闭 jtag 标志，boot_info_lv 是关闭 bootrom 串口打印标志

参数		描述
	value	标志位的值。为 1 代表使能了对应标志的功能，默认为 0。烧写后，需重启单板，value 值才能更新
	lock	lock 状态，lock 为 1，未 lock 为 0。调用烧写接口后会自动 lock
KEY INFO	key	OTP 中的 key，包括 root key 的 sha256 值。因为不可回读，所以不显示实际值
	crc	key 的 crc 校验结果。烧写 otp 后有效，校验成功为 1，校验失败为 0
	lock	lock 状态，lock 为 1，未 lock 为 0。调用烧写接口后会自动 lock
USER DATA INFO	offset	用户自定义 OTP 区域中的偏移，从 0 开始，16 进制显示，单位为 word
	value	用户自定义 OTP 区域中的内容，16 进制显示。如 proc 信息中 offset 为 0 处的值 12345678，对应值为 0x12345678

4.2 XM7206V11A 芯片 OTP 状态

在未使能真实烧写功能时，proc 信息显示的是模拟 OTP 区域中的数据。在通过 echo otp_program_enable=0x1 > /proc/umap/otp 指令使能真实烧写功能后，proc 信息显示的是真实 OTP 区域中的数据。

4.2.1 模拟 OTP 区域内容

模拟 OTP 区域内容如下：

```
- / # cat proc/umap/otp
-----OTP INFO----------------------------------------------------------------
otp_program_enable: 0x0
The proc info will show simulate otp info.

-----FLAG INFO----------------------------------------------------------------
```

```
flag                    value
secure_boot_en    0
ddr_ca_en           0
jtag_disable      0
boot_info_lv      0
nand_boot_mode    0
uart_burn_mode    0
fmc_freq            0

-----KEY INFO-------------------------------------------------------------
key                     value
root_key_sha256 :0000000000000000000000000000000000000000000000000000000000000000
aes_key0            :0000000000000000000000000000000000000000000000000000000000000000
0
aes_key1            :0000000000000000000000000000000000000000000000000000000000000000
0
aes_key2            :0000000000000000000000000000000000000000000000000000000000000000
0
aes_key3            :0000000000000000000000000000000000000000000000000000000000000000
0

-----USER DATA INFO-------------------------------------------------------
offset                                          value
0000:   12345678 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0008:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0010:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0018:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

-----SUPPORT COMMAND------------------------------------------------------------
If you want to modify actual otp value, you must set otp_program_enable to be 0x1.
You can modify otp_program_enable by 'echo otp_program_enable=value > /proc/umap/otp', value
must be Hex.
You can modify otp value in user data by 'echo offset=value > /proc/umap/otp', offset and value
must be Hex.

For example:
echo otp_program_enable=0x1 > /proc/umap/otp          : enable otp program.
echo otp_program_enable=0x0 > /proc/umap/otp          : enable otp simulation.
echo 0x0=0x12345678 > /proc/umap/otp                   : set the first word in otp to be
0x12345678.
------------------------------------------------------------------------------
```

4.2.2 真实 OTP 区域内容

真实 OTP 区域内容如下：

```
#/ # cat proc/umap/otp
-----OTP INFO-----------------------------------------------------------------
otp_program_enable: 0x1
The proc info will show actual otp info.

-----FLAG INFO----------------------------------------------------------------
flag                    value          lock
secure_boot_en          0              0
ddr_ca_en               0              0
jtag_disable            0              0
boot_info_lv            0              0
nand_boot_mode          0              0
uart_burn_mode          0              0
fmc_freq                1              0

-----KEY INFO-----------------------------------------------------------------
key                  crc            lock
root_key_sha256 0                   0
aes_key0             0              0
aes_key1             0              0
aes_key2             0              0
aes_key3             0              0

-----USER DATA INFO-----------------------------------------------------------
offset                                          value
0000:   12345678 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0008:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0010:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0018:   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

-----SUPPORT COMMAND----------------------------------------------------------
If you want to modify actual otp value, you must set otp_program_enable to be 0x1.
You can modify otp_program_enable by 'echo otp_program_enable=value > /proc/umap/otp', value
must be Hex.
You can modify otp value in user data by 'echo offset=value > /proc/umap/otp', offset and value
must be Hex.

For example:
echo otp_program_enable=0x1 > /proc/umap/otp          : enable otp program.
```

```
echo otp_program_enable=0x0 > /proc/umap/otp          : enable otp simulation.
echo 0x0=0x12345678 > /proc/umap/otp                  : set the first word in otp to be
0x12345678.
--------------------------------------------------------------------------------
```

4.2.3 调试信息分析

【调试信息分析】

执行 cat /proc/umap/otp 指令会显示 OTP 区域中的数据烧写情况。根据 otp_program_enable 值的不同，proc 信息也会不同，当 otp_program_enable 为 0x0 时，显示的是模拟 OTP 区的数据信息，当 otp_program_enable 为 0x1 时，显示的是真实 OTP 区的数据信息。

可以通过指令 echo otp_program_enable=x > /proc/umap/otp 修改 otp_program_enable，x 只能为 0x0 或 0x1。

可以通过指令 echo offset=value > /proc/umap/otp 修改用户自定义区 OTP，offset 从 0 开始，小于 OTP_USER_DATA_LEN_MAX，单位为 32bit。offset 和 value 必须为 16 进制数，且只支持 0x 格式，不支持 0X 前缀。value 是对应的值。需要注意的是：用户自定义区 OTP 以 32bit 为 lock 单位，每 32bit 只能写一次，写后不能再更改。当 otp_program_enable 为 0x0 时，通过指令只能烧写模拟的用户自定义区 OTP；当 otp_program_enable 为 0x1 时，通过指令只能烧写真实的用户自定义区 OTP，此时请谨慎使用该指令。

模拟 OTP 区在每次 CIPHER 模块 ko 加载后，数据都会清零。

【参数说明】

以下是模拟 OTP 区域中的数据信息：

参数		描述
OTP INFO	otp_program_enable	真实烧写使能标志。为 0x0 时，未使能；为 0x1 时，使能
FLAG INFO	flag	OTP 中的标志位。secure_boot_en 是安全启动标志，ddr_ca_en 是 ddr 加扰标志，jtag_disable 是关闭 jtag 标志，boot_info_lv 是关闭 bootrom 串口打印标志, nand_boot_mode 是 spi nand 启动模式标志，uart_burn_mode 是 uart 烧写模式标准，fmc_freq 是 FMC 频率标志

参数		描述
	value	标志位的值，为 1 代表使能了对应标志的功能，因为 uart_burn_mode、fmc_freq 有多个选项，所以值也有多个。
KEY INFO	key	OTP 中的 key，包括 root key 的 sha256 值
	value	对应 OTP 的值，16 进制显示
USER DATA INFO	offset	用户自定义区域中的偏移，从 0 开始，16 进制显示，单位为 word
	value	用户自定义区域中的内容，16 进制显示。如 proc 信息中 offset 为 0x0 处的值 12345678，对应值为 0x12345678

以下是真实 OTP 区域中的数据信息：

参数		描述
OTP INFO	otp_program_enable	真实烧写使能标志。为 0x0 时，未使能；为 0x1 时，使能
FLAG INFO	flag	OTP 中的标志位。secure_boot_en 是安全启动标志，ddr_ca_en 是 ddr 加扰标志，jtag_disable 是关闭 jtag 标志，boot_info_lv 是关闭 bootrom 串口打印标志, nand_boot_mode 是 spi nand 启动模式标志，uart_burn_mode 是 uart 烧写模式标准，fmc_freq 是 FMC 频率标志
	value	标志位的值。为 1 代表使能了对应标志的功能，默认为 0。因为 uart_burn_mode、fmc_freq 有多个选项，所以值也有多个。烧写后，需重启单板，value 值才能更新
	lock	lock 状态，lock 为 1，未 lock 为 0。调用烧写接口后会自动 lock。因为 uart_burn_mode、fmc_freq 有多个选项，只有当其烧写了最后一个选项后，才会被 lock

参数		描述
KEY INFO	key	OTP 中的 key，包括 root key 的 sha256 值。因为不可回读，所以不显示实际值
	crc	key 的 crc 校验结果。烧写 otp 后有效，校验成功为 1，校验失败为 0
	lock	lock 状态，lock 为 1，未 lock 为 0。调用烧写接口后会自动 lock
USER DATA INFO	offset	用户自定义 OTP 区域中的偏移，从 0 开始，16 进制显示，单位为 word
	value	用户自定义 OTP 区域中的内容，16 进制显示。如 proc 信息中 offset 为 0 处的值 12345678，对应值为 0x12345678