YTM32系列Fee模块介绍

文档编号： AN0067

版本： Rev.0.1, 11/2022

YTM32系列Fee模块介绍

内容提要

1. 介绍

本文档简述云途YTM32B1MEx系列SDK Fee模块(DFlash模拟EEPROM功能)实现过程及使用方法，主要内容包括以下两部分内容：

1. Fee模块工作原理；
2. Fee模块配置及使用方法；

2. Fee工作原理

YTM32B1Mx系列MCU无内置EEPROM，故对需要使用EERPOM的应用场景可使用DFlash/PFlash软件模拟实现EEPROM功能。

2.1 Fee相关概念

[Sector]: Flash擦除操作基本单位，为Flash固有特性，YTM32B1MEx系列DFlash sector长度为1kByte。

[Cluster]: Cluster为Fee中使用的虚拟存储器段，Cluster建立Fee与物理存储器之间的虚拟连接。它与DFlash物理Sector对应，一个Cluster应包含至少一个物理DFlash Sector。

[Block]: Block为Fee基本存储单元，是Fee数据写入的最小结构，Block长度可根据应用需求自行定义，目前YCT工具中SDK允许配置的最大Block大小为96Byte。

[Cluster Group]: Cluster Group为Fee中虚拟概念，用于关联Block及Cluster。Cluster Group为Block的容器，一个或多个Block组成一个Cluster Group。Fee代码以Cluster Group为单位分配存储空间，同属一个Cluster Group的所有Block共享一片存储空间。Cluster Group中包含的Block数量需根据实际情况确定，但是一个Cluster Group需分配至少两个Cluster。

上述概念关系如图 1所示。

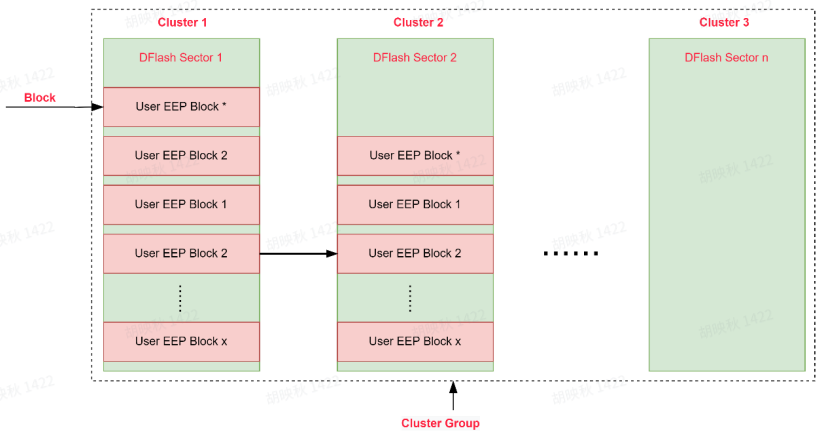

图 1 Fee Block及cluster与物理存储器关系示意图

2.2 Fee记录数据组成

Fee模块记录的数据由两部分构成，分别为:

- **Cluster信息**：该信息描述当前该Cluster Group当前所在Cluster(Sector)的信息，共计32Byte，固定写入在每个Cluster的前32Byte，Cluster擦除后第一次使用该Cluster时被写入。Cluster信息包括Fee Cluster写入ID(4Byte)、Cluster起始地址（相对地址、4Byte）、Cluster长度(4Byte)、Group Hdr校验和（4Byte）以及Cluster有效标志(16Byte)。其中Cluster写入ID每次写入前递增1，因此当前Cluster Group中，ID最大且标志有效的Cluster为当前活跃的Cluster(可执行Fee读、写等操作)。
- **Block信息及数据**：该部分数据为用户定义的Block数据及相关信息，Block数据及信息由3部分组成，包括：①用户EEP Block数据信息(Block Hdr)，共计32Byte，其中Block编号(2Byte)、Block数据长度(2Byte)、Block Data(4Byte)存储地址、Hdr校验和（4Byte）、、Block数据Valid Flag(16Byte)以及预留4byte。②用户Fee Block数据。

Fee数据分布如图 2所示。

Fee用户Block写入过程如下，Block数据在Cluster(Flash Sector)中的分布如图 2所示Cluster信息被写入在Cluster开始的32Byte中，Block信息与数据分别写入到不同地址中，其中Block信息从低地址端开始写入，写入后Fee空闲Block信息地址递增；Block数据从高地址端开始写入，写入后Fee空闲Data地址递减。

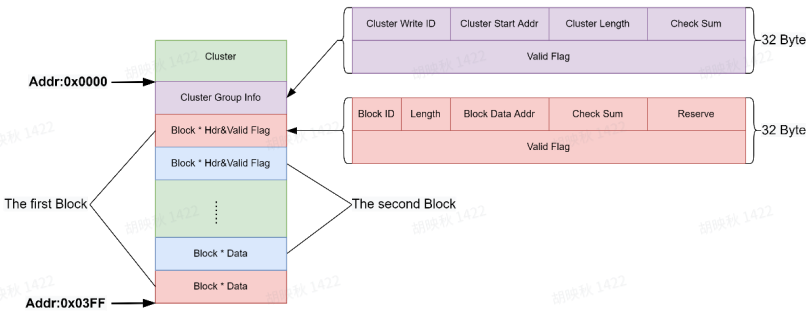

图 2 Fee数据分布示意图

由图2知道，Fee定义的Block数据除用户数据外，需增加32Byte Block 信息，因此为了提高Fee存储器的利用率，在进行Fee Block配置时建议合理分配Block大小，避免出现Block信息大于Block数据的情况。另外，DFlash数据写入最小单位为64bit，因此Block数据长度建议8Byte对齐，避免空间浪费。建议Block大小定义为32Byte-96Byte。

2.3 Fee工作原理

Fee模块根据配置的信息进行存储器分配，分配各Cluster Group使用的虚拟存储器范围、Cluster数量以及Cluster与DFlash物理存储器的映射关系等。在用户请求数据写入时，以Block为单位进行数据写入，将Block数据及信息写入到对应的Cluster中。当前Cluster被写满后，即将当前Cluster Group中所有Block的最新数据搬移至下一个Cluster（Cluster首次写入前会进行擦除），当前Cluster Group的所有Cluster均写满后，擦除最早写入的Cluster并继续写入数据。Fee数据循环写入原理如图 3所示。

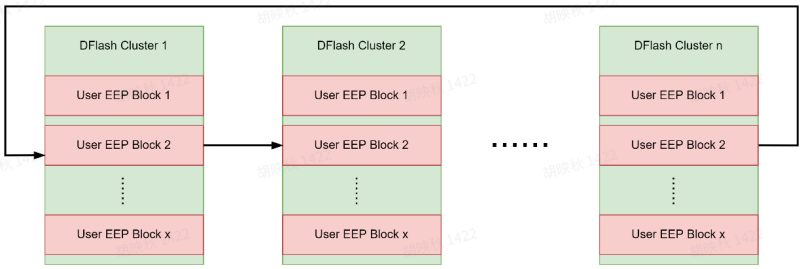

图 3 Fee工作循环写入原理示意图

2.3.1 Fee状态说明

Fee工作过程设计了如下几种状态：

- **MEMIF_UNINIT**：未初始化状态，调用初始化函数前Fee所处的状态。
- **MEMIF_IDLE**：空闲状态，Fee未执行任何操作。此状态下，上层软件可进行Fee操作(读、写、擦除等)。
- **MEMIF_BUSY**：Fee工作状态，此状态下Fee正在执行上层软件申请的操作(读、写等)，若此时上层软件再次申请Fee操作，Fee driver将返回操作失败，新的操作不会被执行。

2.3.2 Fee初始化

Fee初始化过程根据配置初始化Cluster Group及Block信息，获取当前Cluster Group最新数据所在的Cluster位置、最新数据可写入地址、以及所有Block最新数据所在位置等。主要过程需要处理一下操作：

- 未初始化前Fee状态为UNINIT状态。
- 根据图 2中 Cluster Write ID查询Cluster Group最新Cluster位置，并将相关信息(Cluster编号、Write ID等)记录到Fee当前状态信息中，Cluster Write ID为递增式写入，最大值即为当前Cluster Group最新数据所在Cluster。若当前Cluster Group所有Cluster数据均为空，则在第一个Cluster写入Cluster信息。
- 以Block信息长度为单位，读取当前Cluster中的Block信息，初始化当前Cluster Group中所有Block，记录各个Block最新数据所在地址。
- 初始化当前Cluster中未写入数据的地址，为后续Fee写入数据准备。
- 所有Cluster Group初始化完成后将Fee状态置为IDEL状态。

2.3.3 Fee数据写入

上层软件请求Fee数据写入后，Fee模块根据当前Fee是否处于IDLE状态确定能否响应操作。Fee软件根据请求的Block ID识别写入数据所在Cluster，并根据当前Cluster的空闲地址信息写入Block信息及数据。数据写入过程如下：

- 根据写入数据Block编号，准备写入数据地址、长度等信息，将Fee’状态设置为BUSY。
- 查询当前Cluster空闲空间能否满足本次写入，若满足，则依次写入Block Hdr、Block Data、以及Valid标志。写入完成后，将Fee状态设置为IDLE。

- 若Cluster剩余空间不满足本次写入需求，则根据Cluster Group配置擦除下一个Cluster，并依次在该Cluster中写入Cluster 信息、Fee中记录的该Cluster Group中所有Block的最新数据、以及Cluster Valid标志，最后再写入本次申请写入的Block数据。写入完成后，将Fee状态设置为IDLE。
- 数据写入过程中同步跟新本Cluster及Block最新数据地址、空闲数据地址等信息。

2.3.4 Fee数据读取

上层软件请求Fee数据读取后，Fee模块根据当前Fee是否处于IDLE状态确定能否响应操作。Fee软件根据请求的Block ID查询该Block最新数据最在地址，然后根据请求的数据读取偏移地址即长度读取指定数据，Fee数据读取过程如下：

- 根据读取数据Block编号、偏移地址以及长度等信息，准备读取数据地址、长度等信息，将Fee’状态设置为BUSY。
- 根据地址读取DFLASH中数据，并将读取结果拷贝至上层软件指定的位置。
- 读取完成后将Fee状态置为IDLE。

3. Fee软件配置

Fee为通过DFlash模拟实现的EEPROM功能，因此Fee模块配置包括Fee Block、Cluster Group、Cluster以及其使用的DFlash，Fee配置通过结构体以及fee_config.h中的几个宏定义完成。**其中fee_config.h文件因被Fee Driver引用，故该文件不能修改名称，同时该文件中用于定义Fee Block及 Cluster数量的宏定义也不能修改名称。**

3.1 fee_config.h中的配置

在头文件fee_config.h中，提供了Fee Cluster Group数量及用户Block数量限制宏定义，故用户使用时需根据实际情况调整该部分宏定义数值(宏定义名称不允许修改)，如表 1所示。

表 1 Fee配置宏定义列表

序号	宏名称	概述
1	FEE_NUMBER_OF_CLUSTER_GROUPS	配置的Fee Cluster Group数量
2	FEE_CRT_CFG_NR_OF_BLOCKS	配置的Fee Block数量

3.2 fee_config.c中的配置

3.2.1 DFlash空间配置

在fee_config.c中，需要使用结构体Fls_SectorType、Fls_ConfigType配置Fee使用的DFlash空间。

Fls_SectorType用于配置Fee需要使用的DFlash Sector信息，其详情如下所示，对于Fee所用的所有Sector均需要对应的数据进行初始化。

```
typedef struct
{
    /*DFlash Sector虚拟编号*/
    uint32_t sectorId;
    /*Sector虚拟起始地址*/
    Fls_AddressType sectorStartAddress;
    /*Sector长度，单位Byte*/
    Fls_LengthType sectorSize;
    /*DFlash最小写入长度，单位Byte*/
Fls_LengthType pageSize;
    /*Sector物理起始地址*/
    Fls_AddressType sectorHwStartAddress;
    /*DFlash擦、写操作模式，0：同步，1：异步*/
    uint8_t asyncAccess;
} Fls_SectorType
```

示例中我们使用DFLASH物理空间0x10000至0x4FFF段用于Fee，对各个Sector配置如下，该配置中Fee使用的所有Sector均配置为异步操作模式。

```
Fls_SectorType const Fls_SectorCfg0[FLS_CONFIGURED_SECTOR_NUMBER]=
{
        {
            .sectorId = 0U,
            .sectorStartAddress = 0U,
            .sectorSize = FLS_DATA_FLASH_SECTOR_SIZE,
            .pageSize = FLS_DATA_FLASH_PAGE_SIZE,
            .sectorHwStartAddress = 0x100000U,
            .asyncAccess = true,
        },
        {
            .sectorId = 1U,
            .sectorStartAddress = 0x400U,
            .sectorSize = FLS_DATA_FLASH_SECTOR_SIZE,
            .pageSize = FLS_DATA_FLASH_PAGE_SIZE,
            .sectorHwStartAddress = 0x100400U,
            .asyncAccess = true,
        },
……
       {
            .sectorId = 19U,
            .sectorStartAddress = 0x4C00U,
            .sectorSize = FLS_DATA_FLASH_SECTOR_SIZE,
            .pageSize = FLS_DATA_FLASH_PAGE_SIZE,
            .sectorHwStartAddress = 0x104C00U,
```

```
            .asyncAccess = true,
        }
};
```

Fls_ConfigType用于配置Fee使用的DFLASH总体情况，如下所示：

```
const Fls_ConfigType Fls_PBCfg =
{
    .acEraseFunPtr = NULL_PTR,
    .acWriteFunPtr = NULL_PTR,
    &Fee_JobEndNotification,          /* Fee JobEnd 回调函数，Fee Driver中指定*/
    &Fee_JobErrorNotification,        /* Fee JobEerror 回调函数 Fee Driver中指定
*/
    .eDefaultMode = MEMIF_MODE_SLOW,
    .maxReadFastMode = 256U,         /* DFlash Fast Mode 单次读取数据字节数限制*/
    .maxReadNormalMode = 256U,      /* DFlash Normal Mode 单次读取数据字节数限制*/
    .maxWriteFastMode = 128U,         /* DFlash Fast Mode 单次写入数据字节数限制*/
    .maxWriteNormalMode = 8U,        /* DFlash Normal Mode 单次写入数据字节数限制
*/
    .ConfigSectorNum = FLS_CONFIGURED_SECTOR_NUMBER,  /*Fee使用的Flash Sector数
量*/
    .sectorList = Fls_SectorCfg0        /*Fee DFlash Fls_SectorType配置数据指针
*/
};
```

上述配置中Fee默认使用Flash Normal Mode，单次读取数据字节数限制可根据任务占用率实际情况调整。

注意：在进行FLASH sector配置，若代码所在sector与EEPROM配置的某sector处于同一Block，则该sector仅能配置为同步模式，即.asyncAccess 只能配置为false。

3.2.2 Cluster Group配置

Cluster Group配置亦包含两部分，用于指定各个Cluster Group使用的DFlash虚拟Sector地址、长度、使用的Sector数量以及对Sector预留空间的要求等。

示例中定义了两个Cluster Group，各个Group分配DFlash Sector如下所示：

```
/*Cluster Group 0分配Flash,指定虚拟地址范围0x0000—0x4000的16个sector用于Group 0 存
储*/
static const Fee_ClusterType Fee_FeeClusterGroup_0[16] =
{
    {
        0U, /* Start address */
```

```
        0x400U /* Size */
    },
    {
        0x400U, /* Start address */
        0x400U /* Size */
    },
……
    {
        0x3C00U, /* Start address */
        0x400U /* Size */
    }
};
/*Cluster Group1分配Flash,指定虚拟地址范围0x4000–0x5000的4个sector用于Cluster Group 2 存储*/
static const Fee_ClusterType Fee_FeeClusterGroup_1[4] =
{
    {
        0x4000U, /* Start address */
        0x400U /* Size */
    },
……
    {
        0x4C00U, /* Start address */
        0x400U /* Size */
    }
};
```

示例中定义了两个Cluster Group，Sector数量及预留空间配置如下：

```
/* Cluster Groups 配置 */
const Fee_ClusterGroupType Fee_ClrGrps[FEE_NUMBER_OF_CLUSTER_GROUPS] =
{
    /*FeeClusterGroup_0*/
    {
        Fee_FeeClusterGroup_0, /* Cluster set */
        16U, /* Number of clusters */
        0U /* Size of the reserved area */
    },
    /*FeeClusterGroup_1*/
    {
        Fee_FeeClusterGroup_1, /* Cluster set */
        4U, /* Number of clusters */
        0U /* Size of the reserved area */
    }
};
```

3.2.3 Block数据配置

Block的配置信息包括Block ID、数据长度、所属Cluster Group编号、是否支持Block擦除等。

示例中定义了3个Block数据，详情如下：

```
/* Fee Block 配置 */
const Fee_BlockConfigType Fee_BlockConfig[FEE_CRT_CFG_NR_OF_BLOCKS] =
{
    /*Fee Blokk0 配置*/
    {
        1,              /* BLOCK ID 1*/
        32U,            /* Block数据大小 */
        0U,             /* Block所属Cluster Group编号 */
        (uint8_t)true,    /* 支持Block擦除 */
        FEE_PROJECT_RESERVED
    },
    /*Fee Blokk1 配置*/
    {
        2,              /* BLOCK ID 1*/
        50U,            /* Block数据大小 */
        0U,             /* Block所属Cluster Group编号 */
        (uint8_t)true,    /* 支持Block擦除 */
        FEE_PROJECT_RESERVED

    },
    /*Fee Blokk2 配置*/
    {
        3,              /* BLOCK ID 1*/
        4U,             /* Block数据大小 */
        1U,             /* Block所属Cluster Group编号 */
        (uint8_t)true,    /* 支持Block擦除 */
        FEE_PROJECT_RESERVED
    }
};
```

注意：

- **目前YCT工具中SDK允许配置的Block数据最大限制为96Byte，故定义Blcok时数据长度请勿超过该长度。**

- **在进行Block数据配置时，同一个Cluster Group中的所有Block 长度加上Block header长度不能超过该Cluster Group中最小Cluster的长度。**

3.2.4 Fee 模块配置

完成DFlash、Cluster Group、Block等配置后，需将上述配置整合至Fee配置中，如下所示：

```
/*配置Fee*/
const Fee_ModuleUserConfig_t Fee_ModuleConfig =
{
.blockCnt = FEE_CRT_CFG_NR_OF_BLOCKS,          /*Fee需要配置的Block总数量*/
.clusterCnt = FEE_NUMBER_OF_CLUSTER_GROUPS,  /*Fee需要配置的Cluster Group总数量*/
.clusterConfigPtr = Fee_ClrGrps,                    /*Fee Cluster Group配置指针*/
.blockConfigPtr = Fee_BlockConfig,                 /*Fee Block配置指针*/
.flashConfigPtr = &Fls_PBCfg                      /*Fee DFlash配置指针*/
};
```

使用过程中，Fee 配置的几点建议：

1. 根据前文2.2章节可知，Block实际消耗的的存储器长度为Block数据长度+32Byte ，因此Block不宜过小也不宜过大，Block过小浪费存储空间，Block过大会导致读/写时间过长。Block大小建议为8byte整数倍，推荐使用32Byte、64Byte等大小。
2. Cluster Group中的Block数量与大小均会影响Fee性能及寿命，Cluster Group单次所有Block存储消耗的空间（Group中所有Block实际消耗的存储空间总和）应小于Cluster的大小，建议该空间不超过DFlash Sector（1kByte）的1/3。
3. 在规划Cluster Group及Block时，不经常变化的数据与变化频繁的数据分别定义Block及Cluster存放，从而减少Fee翻页操作时，对不经常变化的数据的重复无效拷贝及写入。
4. 为提高Flash使用寿命，尽可能多的分配DFlash用于Fee功能，从而减少DFlash的擦除次数。

4. Fee操作FUNC

4.1 Fee任务调度函数

Fee模块中所有操作均为异步操作，因此在使用Fee时，需周期性调用Fee_MainFunction函数响应上层软件请求的Fee操作，该函数详情如下所示：

Func Name	**void Fee_MainFunction（void）**

param[in]	void
return	void
Description	Service to handle the requested read / write / erase jobs and the internal management operations.

调用Fee_MainFunction函数的任务周期会影响Fee的操作时间，建议不要将该函数分配到周期较大的任务中。

4.2 Fee初始化

ECU上电后，需对调用Fee初始化函数，进行Fee模块初始化，从而完成所有Block最新数据位置扫描，获取所有Block最新数据地址等信息，以便于后续进行读写操作。Fee_Init函数详情如下：

Func Name	**void Fee_Init (const Fee_ConfigType * ConfigPtr)**		
param[in]	const Fee_ConfigType *	ConfigPtr	Fee配置数据结构指针.
return	void		
Description	Service to initialize the FEE module.		

由于Fee为异步操作模式，故调用Fee_Init函数后，仅仅产生Fee Init请求，尚未完成实际初始化操作，此时Fee仍为UNINIT状态。为不影响后续Fee操作，建议在调用初始化函数后，即调用Fee任务调度函数，并等待Fee初始化完成，如下所示：

```
/*Fee 初始化*/
Fee_Init(&Fee_ModuleConfig);
/*执行初始化，并等待初始化完成*/
do
{
    Fee_MainFunction();
    FeeTstRstStatus= Fee_GetStatus();
}while(MEMIF_IDLE!=FeeTstRstStatus);
```

4.3 Fee状态及结果读取

Fee模块当前状态通过调用Fee_GetStatus函数读取，该函数详情如下所示：

Func Name	**MemIf_StatusType Fee_GetStatus (void)**

param[in]	void		
return	MemIf_StatusType	MEMIF_UNINIT	Fee not initial
		MEMIF_IDLE	Fee is idle
		MEMIF_BUSY	Fee is opterated now，wait idle for next operation
		MEMIF_BUSY_INTERNAL	
Description	Service to return the status.		

Fee最近一次操作的结果可通过调用Fee_GetJobResult函数读取，该函数详情如下所示：

Func Name	**MemIf_JobResultType Fee_GetJobResult (void)**		
param[in]	void		
return	MemIf_JobResultType	MEMIF_JOB_OK	Job done OK
		MEMIF_JOB_FAILED	Job done failed
		MEMIF_JOB_PENDING	Job not complement
		MEMIF_JOB_CANCELED	Job canceled
		MEMIF_BLOCK_INCONSISTENT	Requested Fee block is inconsistent
		MEMIF_BLOCK_INVALID	Requested Fee block is invalid
Description	Service to query the result of the last accepted job issued by the upper layer software.		

在执行任何Fee操作前，建议调用Fee_GetStatus确认Fee当前是否处于IDLE状态，执行操作完成后，调用Fee_GetJobResult确认操作是否成功，以便于进行异常处理。

4.4 Fee写操作

Fee写Block数据操作通过调用Fee_Write函数执行，该函数详情如下所示：

Func Name	**Std_ReturnType Fee_Write (uint16 BlockNumber, const uint8 * DataBufferPtr)**

param[in]	uint16	BlockNumber	Number of logical block wants to write.
	const uint8 *	DataBufferPtr	Pointer to write data buffer
return	Std_ReturnType	0:Write request has been accessed; 1:Write request has not been accessed;	
Description	Service to initiate a write job.		

4.5 Fee读操作

Fee读Block数据操作通过调用Fee_Read函数执行，该函数详情如下所示：

Func Name	**Std_ReturnType Fee_Read (uint16 BlockNumber, uint16 BlockOffset, uint8 * DataBufferPtr, uint16 Length)**		
param[in]	uint16	BlockNumber	Number of logical block wants to read.
	uint16	BlockOffset	Read address offset inside the block.
	uint8 *	DataBufferPtr	Pointer to dest data buffer
	uint16	Length	Number of bytes to read
return	Std_ReturnType	0:Read request has been accessed; 1:Read request has not been accessed;	
Description	Service to initiate a read job.		

5. 代码应用示例

关于Fee应用的完整代码，请参考ME0 SDK中的fee。

6. 总结

目前Fee模块依赖于DFlash实现，因此当前Fee模块仅支持YTM32B1MEx系列MCU使用。

参考文档

1. YTM32B1ME0x RM, YTMicro, Rev 1.1, 2022/10/19.
2. AUTOSAR_SRS_EEPROMDriver, AUTOSAR Confidential, Rev4.4, 2018/10/31

3. AUTOSAR_SWS_EEPROMDriver, AUTOSAR Confidential, Rev4.4, 2018/10/31

历史版本

版本	说明	作者	日期
Rev 0.1	初始版本，创建SDK Fee应用笔记。	Colin Xiao	2022.11.10