应用笔记

Application Note

文档编号：**AN1082**

APM32F4xx_SDRAM 应用笔记

版本：**V1.0**

1 引言

本应用笔记提供如何在 APM32F4xx 系列上配置和应用 DMC 接口从而访问 SDRAM 的指南。

目录

2 SDRAM 简介

SDRAM 全称为 Synchronous Dynamic Random Access Memory，同步动态随机存储器。同步是指数据的传输共用一个时钟线作为基准；动态是指存储阵列需要不断的刷新来保证数据不丢失；随机是指数据不是线性依次存储，而是自由指定地址进行数据读写。

2.1 存储结构

SDRAM 中，每个行列地址组合对应一个存储单元，存储单元的大小由 SDRAM 的规格决定，通常为 16bit。一块 SDRAM 通常有 2 个 Bank 或 4 个 Bank。一块 SDRAM 的容量计算是：（2^行地址位数）x（2^列地址位数）x（存储单元大小）x（Bank 数量）。

2.2 SDRAM 参数简述

2.2.1 CAS Latency

CAS 指的是列地址选通信号，地址寻址先行地址再列地址，所以等地址选通之后，就确定了存储单元，接下来就是数据传输。CAS 等待时间（CAS Latency）指的是 CAS 发出之后到第一笔数据输出的这一段时间。CAS Latency 的数值不能超过芯片的设计规范，否则会导致内存的不稳定。

2.2.2 tRCD

tRCD 即 RAS to CAS Delay，翻译过来就是 RAS 到 CAS 延时，RAS 为行地址选通信号，CAS 为列地址选通信号。这是根据芯片存储阵列电子元件响应时间所定制的延迟。

2.2.3 tRP

在对 SDRAM 进行完读写操作后，如果要对同一个 Bank 的另一行进行寻址，就要将原来有效的行关闭，重新发送行/列地址。关闭现有工作行，准备打开新行的操作就是预充电（Precharge）。在发出预充电命令后，需要经过一段时间才能允许发送 RAS 信号打开新的工作行，这段时间称为 tRP（RAS Precharge Time）。

2.2.4 tRAS

tRAS 即 Active to Precharge Command，表示行激活到预充电命令的间隔。

2.2.5 tWR

tWR 即 Write Recovery Time，可以保证数据的可靠写入。

2.2.6 突发长度

突发(Burst)是指在同一行中相邻的存储单元连续进行数据传输的方式，连续传输所涉及到存储单元的数量就是突发长度。

2.2.7 刷新

动态存储器要不断进行刷新(Refresh)才能保留住数据，因此它是 SDRAM 最重要的操作。刷新操作有固定的周期，依次对所有行进行操作，以保留那些久久没有经历重写的存储体中的数据。

两个刷新的间隔多长合适呢？目前公认的标准是，存储体中电容的数据有效保存期上限是 64ms，也就是刷新所有行的时间是 64ms，这样刷新速度就是：行数量/64ms。

刷新操作分为两种：自动刷新(Auto Refresh)与自刷新(Self Refresh)。两种刷新都不需要外部提供行地址信息，因为这是一个内部的自动操作。

对于自动刷新，SDRAM 内部有一个行刷新计数器自动地依次生成行地址。由于刷新是针对一行中的所有存储体进行，所以无需列寻址，或者说 CAS 在 RAS 之前有效。所以，自动刷新又称为 CBR 式刷新。由于刷新涉及到所有 Bank，因此在刷新过程中，所有 Bank 都停止工作，而每次刷新所占用的时间为 9 个时钟周期，之后就可以进入正常的工作状态，也就是说在这 9 个时钟周期内，所有工作指令只能等待而无法执行。64ms 之后则再次对同一行进行刷新，如此周而复始进行循环刷新。

自刷新则主要用于休眠模式低功耗状态下的数据保存，这方面最著名的应用就是 STR。在发出自动刷新命令时，将 CKE 置于无效状态，就进入了自刷新模式，此时不再依靠系统时钟工作，而是根据内部的时钟进行刷新操作。在自刷新期间除了 CKE 之外所有外部信号都是无效的，只有重新使能 CKE 才能退出自刷新模式并进入正常操作状态

3 DMC 简介

DMC 是一个动态存储控制器，通过它外接 SDRAM 能够实现 SDRAM 的读写。如下图 1 所示，通过 DMC 可以将 SDRAM 的数据与 AHB 总线进行传输。

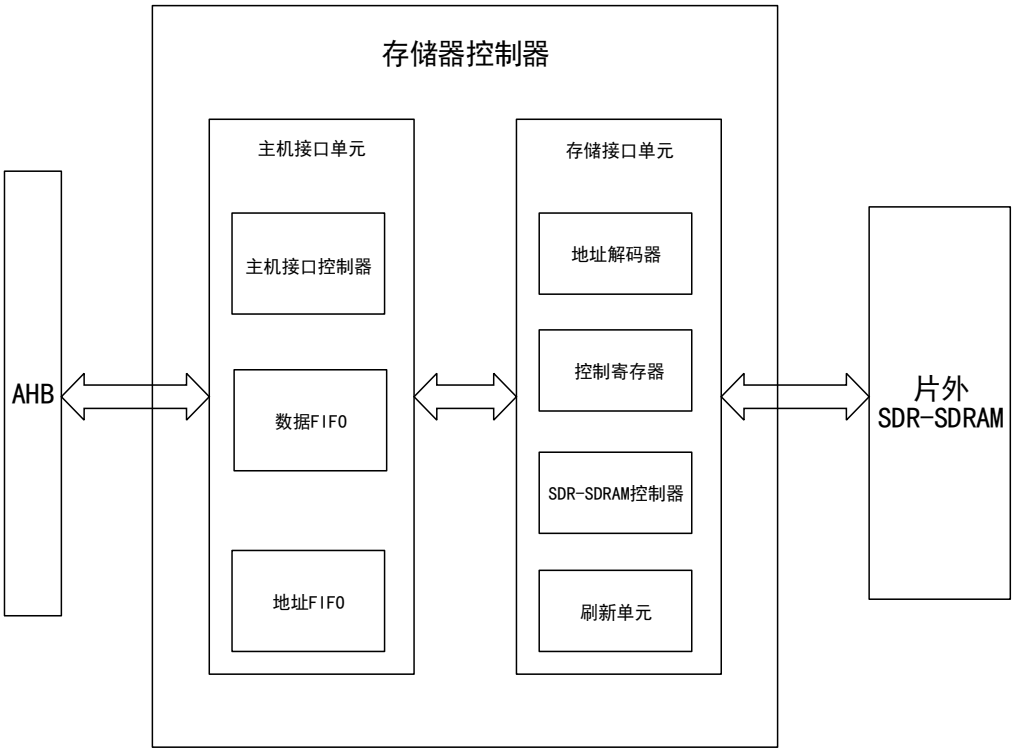

图 1 DMC 结构框图

3.1 DMC 引脚定义

DMC 的引脚定义如下表所示：

表格 1 DMC 引脚定义

信号名称	输入/输出	管脚	功能
A0	输出	PF1	地址
A1	输出	PF2	地址
A2	输出	PF3	地址
A3	输出	PF4	地址
A4	输出	PF6	地址

信号名称	输入/输出	管脚	功能
A5	输出	PF7	地址
A6	输出	PF8	地址
A7	输出	PF9	地址
A8	输出	PF10	地址
A9	输出	PH3	地址
A10	输出	PF0	地址
D0	输入/输出	PG3	双向数据
D1	输入/输出	PG4	双向数据
D2	输入/输出	PG5	双向数据
D3	输入/输出	PG6	双向数据
D4	输入/输出	PG8	双向数据
D5	输入/输出	PH13	双向数据
D6	输入/输出	PH15	双向数据
D7	输入/输出	PI3	双向数据
D8	输入/输出	PH8	双向数据
D9	输入/输出	PH10	双向数据
D10	输入/输出	PD10	双向数据
D11	输入/输出	PD12	双向数据
D12	输入/输出	PD13	双向数据

信号名称	输入/输出	管脚	功能
D13	输入/输出	PD14	双向数据
D14	输入/输出	PD15	双向数据
D15	输入/输出	PG2	双向数据
BA	输出	PI11	Bank 地址
CKE	输出	PH5	时钟使能
CLK/CK	输出	PG1	时钟
LDQM	输入	PG15	16 位数据写入
UNQM	输入	PF11	16 位数据读取
NWE	输出	PI7	写使能
NCAS	输出	PI8	列地址位选通命令
NRAS	输出	PI9	行地址位选通命令
NCS	输出	PI10	片选

行列地址线共用引脚，当前 APM32F407 支持一根 bank 地址线，即可以支持 2 个 bank。

3.2 DMC 引脚初始化

DMC 的引脚初始化可以参考如下代码示例，GPIO 需要配置引脚复用：

```
    GPIO_Config_T gpioConfig;

    RCM_EnableAHB1PeriphClock(RCM_AHB1_PERIPH_GPIOD);

    gpioConfig.speed = GPIO_SPEED_50MHz;
    gpioConfig.mode = GPIO_MODE_AF;
    gpioConfig.otype = GPIO_OTYPE_PP;
    gpioConfig.pupd = GPIO_PUPD_NOPULL;
gpioConfig.pin = GPIO_PIN_10;
/** 以 PD10 引脚作为配置参考，其他引脚类似 */
GPIO_Config(GPIOD, &gpioConfig);

/** 需要配置引脚复用为 EMMC */
    GPIO_ConfigPinAF(GPIOD, GPIO_PIN_SOURCE_10, GPIO_AF_EMMC);
```

4 DMC 的初始化及 SDRAM 访问

SDRAM 应用的流程图：

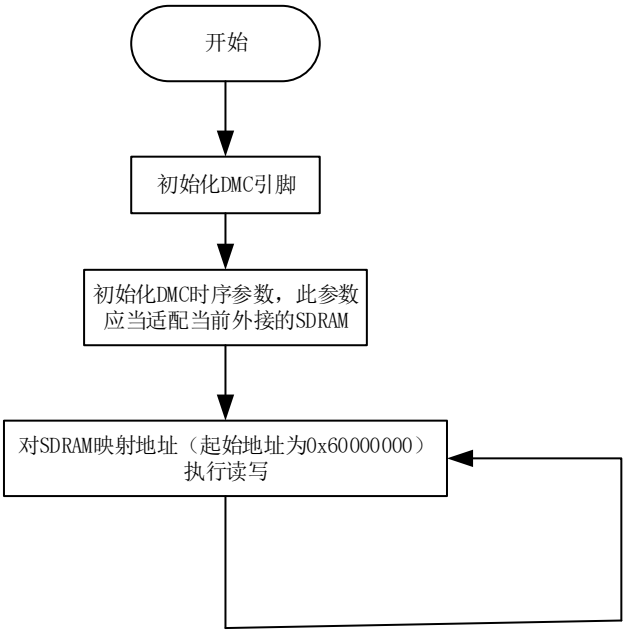

图 2 程序流程图

4.1 初始化 DMC

4.1.1 DMC 初始化结构体

DMC_Config_T 结构体定义在 APM32F4xx_dmc.h 文件中，具体定义如下：

```
/**
 * @brief DMC Config struct
 */
typedef struct
{
    DMC_BANK_WIDTH_T        bankWidth;      //!< Number of bank bits
    DMC_ROW_WIDTH_T         rowWidth;       //!< Number of row address bits
    DMC_COL_WIDTH_T         colWidth;       //!< Number of col address bits
    DMC_CLK_PHASE_T         clkPhase;       //!< Clock phase
    DMC_TimingConfig_T      timing;         //!< Timing
}DMC_Config_T;
```

结构体成员中：

bankWidth 表示 bank 地址宽度，设置为 1 时，能支持 2 个 bank，为 2 时，支持 4 个 bank;

rowWidth 表示行地址宽度

colWidth 表示行地址宽度

clkPhase 表示时钟相位

timing 为 DMC 的时序配置，对应 DMC_TimingConfig_T 结构体成员。

目前 APM32F407 的 DMC 支持行地址 11 位，支持列地址 8 位，支持 1 位 bank，在存储单元为 16bit 数据即 2 个 bytes 的情况下，支持的最大容量为 2^11 * 2^8 * 2 * 2 = 2Mbytes。

timing 类型为 DMC_TimingConfig_T 结构体，其定义在 APM32F4xx_dmc.h 文件中，具体定义如下：

```
/**
 * @brief Timing config definition
 */
typedef struct
{
    uint32_t latencyCAS : 2;   //!< DMC_CAS_LATENCY_T
    uint32_t tRAS    : 4;      //!< DMC_RAS_MINIMUM_T
    uint32_t tRCD    : 3;      //!< DMC_DELAY_TIME_T
    uint32_t  tRP    : 3;      //!< DMC_PRECHARGE_T
    uint32_t  tWR    : 2;      //!< DMC_NEXT_PRECHARGE_T
    uint32_t  tARP   : 4;      //!< DMC_AUTO_REFRESH_T
    uint32_t  tCMD   : 4;      //!< DMC_ATA_CMD_T
    uint32_t  tXSR   : 9; //!< auto-refresh commands, can be 0x000 to 0x1FF
    uint16_t  tRFP   : 16;//!< Refresh period, can be 0x0000 to 0xFFFF
}DMC_TimingConfig_T;
```

结构体成员中：

latencyCAS 表示 CAS 等待时间；

tRAS 表示行激活和预充电的之间的最小时间；

tRCD 表示 RAS 到 CAS 的延迟时间；

tRP 表示预充电周期；

tWR 表示写最后一个数据到下一次预充电的时间；

tARP 表示两次自动刷新命令间的最小时间间隔；

tCMD 表示 Active to active 命令周期；

tXSR 表示退出自刷新切换至激活命令或自动刷新读命令之间的最小间隔时间；

tRFP 表示连续两次刷新间隔；

上述参数的单位为 DMC 时钟周期，DMC 的时钟由 AHB 时钟经过 DMC 分频而来。

4.1.2 DMC 配置

配置 DMC 可以参考下方代码：

```
uint32_t sdramCapacity;
DMC_Config_T dmcConfig;
DMC_TimingConfig_T timingConfig;

RCM_EnableAHB3PeriphClock(RCM_AHB3_PERIPH_EMMC);

timingConfig.latencyCAS = DMC_CAS_LATENCY_3;
timingConfig.tARP       = DMC_AUTO_REFRESH_10;
timingConfig.tRAS       = DMC_RAS_MINIMUM_2;
timingConfig.tCMD       = DMC_ATA_CMD_1;
timingConfig.tRCD       = DMC_DELAY_TIME_1;
timingConfig.tRP        = DMC_PRECHARGE_1;
timingConfig.tWR        = DMC_NEXT_PRECHARGE_2;
timingConfig.tXSR       = 3;
timingConfig.tRFP       = 0x2F9;
dmcConfig.bankWidth     = DMC_BANK_WIDTH_1;
dmcConfig.clkPhase      = DMC_CLK_PHASE_REVERSE;
dmcConfig.rowWidth      = DMC_ROW_WIDTH_11;
dmcConfig.colWidth      = DMC_COL_WIDTH_8;
dmcConfig.timing        = timingConfig;

DMC_Config(&dmcConfig);  //!< 配置 DMC
DMC_EnableAccelerateModule(); //!<开启 DMC 缓冲器，可以提升 SDRAM 读性能

DMC_Enable(); //!< 使能 DMC
```

在配置时序参数时需要根据 SDRAM 的数据手册，以及当前系统时钟分配到 DMC 外设的时

钟频率来对各个参数进行合理配置。上述代码中的参数配置示例仅供参考。

4.2 SDRAM 访问

SDRAM 的起始地址为 0x60000000，根据不同容量，其结束地址与之对应。例如容量为 2Mbytes 的 SDRAM 的结束地址为 0x60200000。在程序中我们访问 SDRAM 可以通过地址直接访问，无需解锁。

例如对 SDARM 的地址 0x60000000 进行数据读取，读取大小为 1 个字节，则可以使用语句：

```
uint8_t data = *(uint8_t*) (0x6000000);
```

例如对 SDARM 的地址 0x60000000 写入一个字节，为 0x55，则可以使用语句：

```
*(uint8_t*) (0x6000000) = 0x55;
```

5 版本历史

表格 2 文件版本历史

日期	版本	变更历史
2022.05.31	1.0	新建

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“TM”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，除非在极海销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及／或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册相关信息而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失）。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。