NandFlash 的基本使用

RT-Thread 评估板 RealTouch 裸机例程

版本号：1.0.0

日期：2012/8/14

修订记录

日期	作者	修订历史
2012/8/14	Heyuanjie87	添加例程说明

NANDFLASH

本章节介绍了 STM32 的 NandFlash 控制器和 K9F2G08 的部分信息。并实现了对其的基本操作，通过本章学习你将基本掌握 STM32 NandFlash 控制器以及 K9F2G08 的使用方法。

1. STM32 NANDFLASH 控制器简介

STM32 的 NandFlash 控制器是包含在 FSMC 中的，它可以控制多种设备，在此只针对 NandFlash 控制部分给予介绍。

FSMC 共控制 4 个 Banks，其中 Bank2 和 Bank3 可用于挂载 NandFlash。Bank2 的地址空间为 0x7000 0000 到 0x7800 0000,Bank3 的地址空间为 0x8000 0000 到 0x8800 0000。bank2/bank3 的前半部分为公共区，后半部分为属性区,这两个区的低 256KB 部分又被分为 3 个段:数据段(开始 64KB)、命令段（第二个 64KB）地址段（最后 128KB）。用户可以使用这三个段访问 NandFlash：写命令到命令段任意位置可向 NandFlash 发送命令、。。。、读写 NandFlash 时可访问数据段任意位置。在通过数据段访问 NandFlash 时你可以不必增加数据地址。

FSMC 可以控制 8 位或 16 位的 NandFlash，在这里仅对 8 位 NandFlash 相关部分给予介绍，表 1 是 8 位 NandFlash 要用到的引脚介绍。

表 1. 8 位 NandFlash

FSMC 信号名称	I/O	功能
A[17]	O	地址锁存使能
A[16]	O	命令锁存使能
D[7:0]	I/O	8 位地址/数据复用端口
NCE[x]	O	片选，x=2/,3
NOE(=NRE)	O	输出使能
NWE	O	写使能
NWAIT/INT[3:2]	I	NandFlash 就绪/忙信号 输入给 FMSC

1.1 NANDFLASH 操作

一个典型的读页操作可以分为如下几个步骤完成：

1) 配置 FSMC_PCRx 与 FSMC_PMEMx 寄存器以使能相应的存储 Bank。
2) CPU 写一个命令字节到公共存储区（例如三星的 NandFlash 设备可以写 0x00）。

3) CPU 发送 4 字节的起始地址（小容量设备 3 字节）到公共区或属性区（地址顺序为先[7:0],再[16:9],[24:17],64Mbit x 8bit 设备需要在最后发送[25]）。

4) 等待 NandFlash 就绪（R/NB 变为高电平）。

5) 然后 CPU 就可以一字节一字节的从公共区读出数据(主数据与额外数据）。

1.2 NANDFLASH 控制器寄存器介绍

表 1.2.1 控制寄存器

	19 17	16 13	12 9		6	5 4	3	2	1	
保留	ECCPS	TAR	TCLR	保留	ECCEN	PWID	PTYP	PBKEN	PWAITEN	保留

位 19:17 ECCPS 表示 ECC 页大小，其取值与含义如下所示：

- B000 – 256 字节
- B001 – 512 字节
- B010 – 1024 字节
- B011 – 2048 字节
- B100 – 4096 字节
- B101 – 8192 字节

位 16:13 TAR 表示 ALE 到 RE 之间的延时，用来设置 ALE 低电平到 RE 低电平之间需要经过多少个 HCLK 时钟周期。延时 T = （TAR + SET + 1）/ HCLK(SET is MEMSET or ATTSET according to the addressed space)。

位 12:9 TCLR 表示 CLE 到 RE 之间的延时，来设置 ALE 低电平到 RE 低电平之间需要经过多少个 HCLK 时钟周期。延时 T = （TCLR + SET + 1）/ HCLK。

位 6 ECCEN 表示 ECC 计数使能，0 禁止、1 开启。

位 5:4 PWID 表示数据总线宽度，0 表示 8 位、1 为 16 位其它取值保留。

位 3 PTYP 表示存储器类型，1 为 NandFlash、0 为 PC 卡。

位 2 PBKEN 表示 PC 卡 NandFlash 存储 Bank 使能，0 表示禁止、1 为使能。

位 1 PWAITEN 表示等待使能，0 禁止、1 使能。

表 1.2.2 FIFO 状态与中断寄存器

31 7	6	5	4	3	2	1	0
保留	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS

位 6 FEMPT 表示 FIFO 是否为空，1 为空、非空。

位 5 IFEN 表示中断下降沿检测使能，1 使能、0 禁止。

位 4 ILEN 表示中断高电平检测使能，1 使能、0 禁止。

位 3 IREN 表示中断上升沿检测使能，1 使能、0 禁止。

位 2 IFS 表示中断下降沿状态。

位 1 ILS 表示中断高电平状态。

位 0 IRS 表示中断上升沿状态。

表格 1.2.3 共公存储区时序寄存器

31 24	23 6	15 8	7 0
MEMHIZ	MEMHOLD	MEMWAIT	MEMSET

位 31:24 MEMHIZ 表示公共存储区数据总线高阻时间，用来设置启动一个对公共区写访问后数据总线应保持多少个 HCLK 时钟周期的高阻状态（初始值 255）。

位 23:16 MEMHOLD 表示公共存储区保持时间，用来设置通过公共存储区访问 NandFlash 时命令无效置位后应保持多少个 HCLK 时钟周期的地址（初始值 255）。

位 15:8 MEMWAIT 表示公共存储区等待时间，用来定义判定命令（NWE, NOE）的最小周期（（MEMWAIT+1）/ HCLK）。如果在到达设定的时间后 NWAIT 任然为低则命令判定的时间会延长（初始值 255）。

位 7:0 MEMSET 表示公共存储区设置时间，用来定义在判定命令（NEW, NOE）前的多少个 HCLK 设置地址。

2. 本例程所用硬件介绍

本例程主要使用了 K9F2G08，它是一个 8 位 NandFlash 包含 256MB 主存区和 8M B 额外存储区。每页大小为 2KB+64B，每块大小为 128KB+4KB。页编程时间 200μs 块擦出时间 1.5ms。命令、地址、数据复用 I/O 端口。

2.1 引脚描述

表 2.1　K9F2G08 引脚定义

引脚名称	说明
I/O 0~7	用于输入命令、地址和数据的输入输出。当芯片未被选择或输出被禁止时 I/O 引脚处于高阻状态。
CLE	命令锁存使能；当 CLE 为高电平时，命令在 NWE 信号的上升沿通过 I/O 引脚锁存进命令寄存器。
ALE	地址锁存使能；当 ALE 为高电平时，地址在 NWE 的上升沿锁存进地址寄存器。
NCE	芯片使能；
NRE	读使能；
NWE	写使能；在 NWE 上升沿时命令、地址、数据被锁存
NWP	写保护；
R/NB	就绪/忙状态；输出低电平表示忙，高表示就绪，这个引脚是开漏输出且当芯片未选择或输出禁止时不会变为高阻。
Vcc	电源；
Vss	地 ；
N.C	无效引脚；

2.2 地址传输格式

表 2.2　地址传输格式

顺序	I/O0	I/O1	I/O2	I/O3	I/O4	I/O5	I/O6	I/O7
1st	A0	A1	A2	A3	A4	A5	A6	A7
2nd	A8	A9	A10	A11	0	0	0	0
3rd	A12	A13	A14	A15	A16	A17	A18	A19
4th	A20	A21	A22	A23	A24	A25	A26	A27
5th	A28	0	0	0	0	0	0	0

A0-A11 为行地址，是 NandFlash 内数据寄存器的地址，A12-A28 为列地址。

2.3 NANDFLASH 操作命令码

表 2.3 命令

功能	1st	2nd	设备忙时可接受的命令
Read	0x00	0x30	
Read for Copy Back	0x00	0x35	
Read ID	0x90	-	
Reset	0xFF	-	0
Page Program	0x80	0x10	
Two-Plane Page Program	0x80-0x11	0x81-0x10	
Copy-Back Program	0x85	0x10	
Two-Plane Copy-Back Program	0x85-0x11	0x81-0x10	
Block Erase	0x60	0xD0	
Two-Plane Block Erase	0x60-0x60	0xD0	
Random Data Input	0x85	-	
Random Data Output	0x05	0xE0	
Read Status	0x70	-	0
Read Status 2	0xF1	-	0

备注：随机数据读写只能在一个页中执行。

2.4 ID 中包含的信息

表 2.4 芯片 ID

第一字节	厂商 ID
第二字节	设备 ID
第三字节	内部芯片编号、存储单元类型、同时编程的页面数
第四字节	页大小、块大小、额外区大小、
第五字节	Plane Number, Plane Size

表 2.5 第三字节 ID 数据

名称	描述	I/07	I/06	I/0 5 4	I/0 3 2	I/0 1 0
芯片编号	1					0 0
	2					0 1
	4					1 0
	8					1 1
单元类型	2 级单元				0 0	
	4 级单元				0 1	
	8 级单元				1 0	
	16 级单元				1 1	
同时编程页数	1			0 0		
	2			0 1		
	4			1 0		
	8			1 1		
多芯片间交错编程	不支持		0			
			1			
Cache 编程	不支持	0				
		1				

表 2.6　第四字节 ID 数据

名称	描述	7	6	5 4	3	2	1 0
页大小	1KB						0 0
	2KB						0 1
	4KB						1 0
	8KB						1 1
块大小	64KB			0 0			
	128KB			0 1			
	256KB			1 0			
	512KB			1 1			
额外区大小（B/512B）	8					0	
	16					1	
Organization	X8		0				
	X16		1				
串行访问最低限度	50/30ns	0			0		
	25ns	1			1		
	保留	0			0		
	保留	1			1		

表格 2.7　第五字节 ID 数据

名称	描述	7	6	5	4	3	2	1	0
Number of plane	1					0	0		
	2					0	1		
	4					1	0		
	8					1	1		
Plane size	64Mbit		0	0	0				
	128Mbit		0	0	1				
	256Mbit		0	1	0				
	512Mbit		0	1	1				
	1Gbit		1	0	0				
	2Gbit		1	0	1				
	4Gbit		1	1	0				
	8Gbit		1	1	1				
保留		0						0	0

3. 例程

本例程主要实现了对 NandFlash 的基本操作：读 ID、读写数据和擦出。关于 NandFlash 以及相关 GPIO 的初始化过程和函数中用到的一些宏定义请参见示例工程，这里仅对 NandFlash 的几个基本操作给予介绍。

3.1 读设备 ID

代码 3-1 读取设备 ID

```
  /* fsmc_nand.c */

  void FSMC_NAND_ReadID(NAND_IDTypeDef* NAND_ID)
  {
     /* Send Command to the command area */
     *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) = NAND_CMD_READID;
      /* Send Address to the address area */
     *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) = 0x00;

     /* Sequence to read ID from NAND flash */
     NAND_ID->Maker_ID   = *(vu8 *)(NAND_FLASH_START_ADDR |
DATA_AREA);
```

```
     NAND_ID->Device_ID  = *(vu8 *)(NAND_FLASH_START_ADDR |
DATA_AREA);
     NAND_ID->Third_ID   = *(vu8 *)(NAND_FLASH_START_ADDR |
DATA_AREA);
     NAND_ID->Fourth_ID  = *(vu8 *)(NAND_FLASH_START_ADDR |
DATA_AREA);
     NAND_ID->Fifth_ID   = *(vu8 *)(NAND_FLASH_START_ADDR |
DATA_AREA);
  }
```

有些 NandFlash 控制器中有命令寄存器、地址寄存器和数据寄存器，但在 STM32 中是命令区、地址区和数据区的概念。它有很大的一块区域，只要向这块区域的任意位置写命令、地址和数据都能实现相应的发送。RealTouch 使用 Bank3 挂载 NandFlash，因此 NAND_FLASH_START_ADDR 被定义为 0x80000000。

CMD_AREA 为 0x010000,ADDR_AREA 为 0x020000,DATA_AREA 为 0x000000（见 STM32 参考手册 Table 166）。读设备 ID 的时序为先发送读 ID 命令（NAND_CMD_READID）再发送地址 0x00,然后就可以一个字节一个字节地读出 5 字节的 ID 数据。

3.2 块擦除

代码 3-2　块擦除

```
  /* fsmc_nand.c */

  uint32_t FSMC_NAND_EraseBlock(NAND_ADDRESS Address)
  {
      *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) = NAND_CMD_ERASE0;

      *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_1st_CYCLE(ROW_ADDRESS);
      *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_2nd_CYCLE(ROW_ADDRESS);

  #ifdef K9F2G08
      *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_3rd_CYCLE(ROW_ADDRESS);
  #endif

      *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) = NAND_CMD_ERASE1;

      while( GPIO_ReadInputDataBit(GPIOG, GPIO_Pin_6) == 0);

      return (FSMC_NAND_GetStatus());
  }
```

块擦出的时序是先发送第一个擦出命令，再发出 3 个字节的列地址（A12-A17 会被 NandFlash 忽略）然后发出第二个擦出命令。NandFlash 的 R/NB 引脚接在 PG6 上的，因此此处通过 PG6 判断设备是否忙。

3.3 读取一页的数据

代码 3-3　读取一页的数据

```
  /* fsmc_nand.c */

  uint32_t FSMC_NAND_ReadPage(uint8_t *pBuffer, NAND_ADDRESS
Address, uint32_t NumPageToRead)
  {
      uint32_t index = 0x00, numpageread = 0x00, addressstatus =
NAND_VALID_ADDRESS;
      uint32_t status = NAND_READY, size = 0x00;
```

```
     while((NumPageToRead != 0x0) && (addressstatus ==
NAND_VALID_ADDRESS))
     {
        /* Page Read command and page address */
        *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) =
NAND_CMD_READ_1;

        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) = 0x00;
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) = 0X00;
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_1st_CYCLE(ROW_ADDRESS);
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_2nd_CYCLE(ROW_ADDRESS);
  #ifdef K9F2G08
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_3rd_CYCLE(ROW_ADDRESS);
  #endif

        *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) =
NAND_CMD_READ_TRUE;

        while( GPIO_ReadInputDataBit(GPIOG, GPIO_Pin_6) == 0 );

        /* Calculate the size */
        size = NAND_PAGE_SIZE + (NAND_PAGE_SIZE * numpageread);

        /* Get Data into Buffer */
        for(; index < size; index++)
        {
           pBuffer[index]= *(vu8 *)(NAND_FLASH_START_ADDR |
DATA_AREA);
        }

        numpageread++;

        NumPageToRead--;

        /* Calculate page address */
        addressstatus = FSMC_NAND_AddressIncrement(&Address);
     }

     status = FSMC_NAND_GetStatus();

     return (status | addressstatus);
```

```
  }
```

读页的处理过程跟块擦出比较类似，这里两个行地址为 0 表示从一页的开始处读取数据。

3.4 写数据

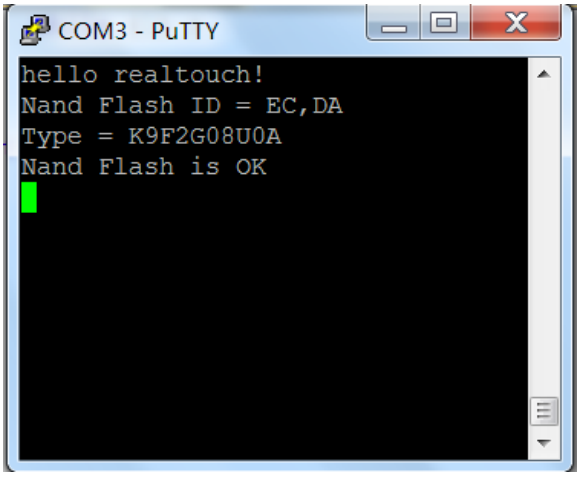
代码 3-4　写数据

```
  /* fsmc_nand.c */

  uint32_t FSMC_NAND_WritePage(uint8_t *pBuffer, NAND_ADDRESS
Address, uint32_t NumPageToWrite)
  {
     uint32_t index = 0x00, numpagewritten = 0x00, addressstatus
= NAND_VALID_ADDRESS;
     uint32_t status = NAND_READY, size = 0x00;

     while((NumPageToWrite != 0x00) && (addressstatus ==
NAND_VALID_ADDRESS) && (status == NAND_READY))
     {
        /* Page write command and address */
        *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) =
NAND_CMD_PAGEPROGRAM;

        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) = 0x00;
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) = 0X00;
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_1st_CYCLE(ROW_ADDRESS);
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_2nd_CYCLE(ROW_ADDRESS);
  #ifdef K9F2G08
        *(vu8 *)(NAND_FLASH_START_ADDR | ADDR_AREA) =
ADDR_3rd_CYCLE(ROW_ADDRESS);
  #endif

        /* Calculate the size */
        size = NAND_PAGE_SIZE + (NAND_PAGE_SIZE * numpagewritten);

        /* Write data */
        for(; index < size; index++)
        {
```

```
            *(vu8 *)(NAND_FLASH_START_ADDR | DATA_AREA) =
pBuffer[index];
        }

        *(vu8 *)(NAND_FLASH_START_ADDR | CMD_AREA) =
NAND_CMD_PAGEPROGRAM_TRUE;

        while( GPIO_ReadInputDataBit(GPIOG, GPIO_Pin_6) == 0 );

        /* Check status for successful operation */
        status = FSMC_NAND_GetStatus();

        if(status == NAND_READY)
        {
           numpagewritten++;

           NumPageToWrite--;

           /* Calculate Next page Address */
           addressstatus = FSMC_NAND_AddressIncrement(&Address);
        }
     }

     return (status | addressstatus);
  }
```

4. 下载运行

执行 fsmc_nand.c 中的 FSMC_NAND_Test 就可对 NandFlash 进行基本的读写测试了。在这个函数中首先读取了设备 ID 以判断型号，并擦出 NandFlash 中的某块，然后写入一页数据再读取出来进行对比。如果无误则通过串口输出“Nand Flash is OK”否则输出“Nand Flash is error”。下图是运行结果：


COM3 - PuTTY
hello realtouch!
Nand Flash ID = EC,DA
Type = K9F2G08U0A
Nand Flash is OK