



使用大容量 STM32F10xxx 的 FSMC 驱动外部的存储器

前言

这个应用笔记说明了如何使用大容量的STM32F10xxx的FSMC(灵活的静态存储器控制器)驱动一组外部的存储器。文中首先简要地介绍了STM32F10xxx的FSMC控制器，然后给出了包含典型的FSMC配置的存储器接口实例，以及时序计算和硬件连接方法。

本应用笔记的实例是基于STM3210E-EVAL评估版上的存储器，这是大容量STM32F10xxx的评估版。使用的存储器是一个16位的异步NOR闪存存储器，一个8位的NAND闪存存储器和一个16位的异步SRAM存储器。

文中实例用到的固件库函数和不同存储器的驱动程序，可以在STMicroelectronics的网站上下载：www.st.com/mcu。

译注：

STM3210E-EVAL评估板演示程序说明文档下载地址：

<http://www.st.com/stonline/products/literature/um/14703.pdf>

STM3210E-EVAL评估板演示程序包下载地址：

<http://www.st.com/stonline/products/support/micro/files/um0549.zip>

STM3210E-EVAL评估板线路图下载地址：

<http://www.st.com/stonline/products/support/micro/files/um0488.zip>

本译文的英文版下载地址为：

<http://www.st.com/stonline/products/literature/anp/14779.pdf>

目录

1	STM32F10xxx灵活的静态存储器控制器简介	3
2	与非总线复用模式的异步 16 位NOR闪存接口	5
2.1	FSMC配置	5
2.1.1	与NOR闪存存储器接口的典型应用	6
2.2	时序计算	7
2.3	硬件连接	8
2.4	从外部NOR闪存存储器执行代码	9
3	与非总线复用的 16 位SRAM接口	11
3.1	FSMC配置	11
3.1.1	使用FSMC与SRAM存储器接口的典型应用	12
3.2	时序计算	12
3.3	硬件连接	13
3.4	使用外部SRAM作为数据存储器	14
4	与 8 位的NAND闪存存储器接口	15
4.1	FSMC配置	15
4.1.1	使用FSMC与NAND存储器接口的典型应用	16
4.2	时序计算	17
4.3	硬件连接	18
4.4	错误校验码计算	19
4.4.1	错误校验码(ECC)计算概述	19
4.4.2	错误检测	20
5	100 脚的STM32F10xxx的FSMC配置	22
5.1	FSMC与NAND存储器接口	22
5.2	FSMC与NOR存储器接口	22

1 STM32F10xxx灵活的静态存储器控制器简介

灵活的静态存储器控制器(FSMC)是内置于大容量STM32F10xxx的外部存储器控制器。使用这个控制器，STM3210xxx微控制器可以与许多存储器连接，包括SRAM、NOR闪存和NAND闪存等。

FSMC包含2类控制器：

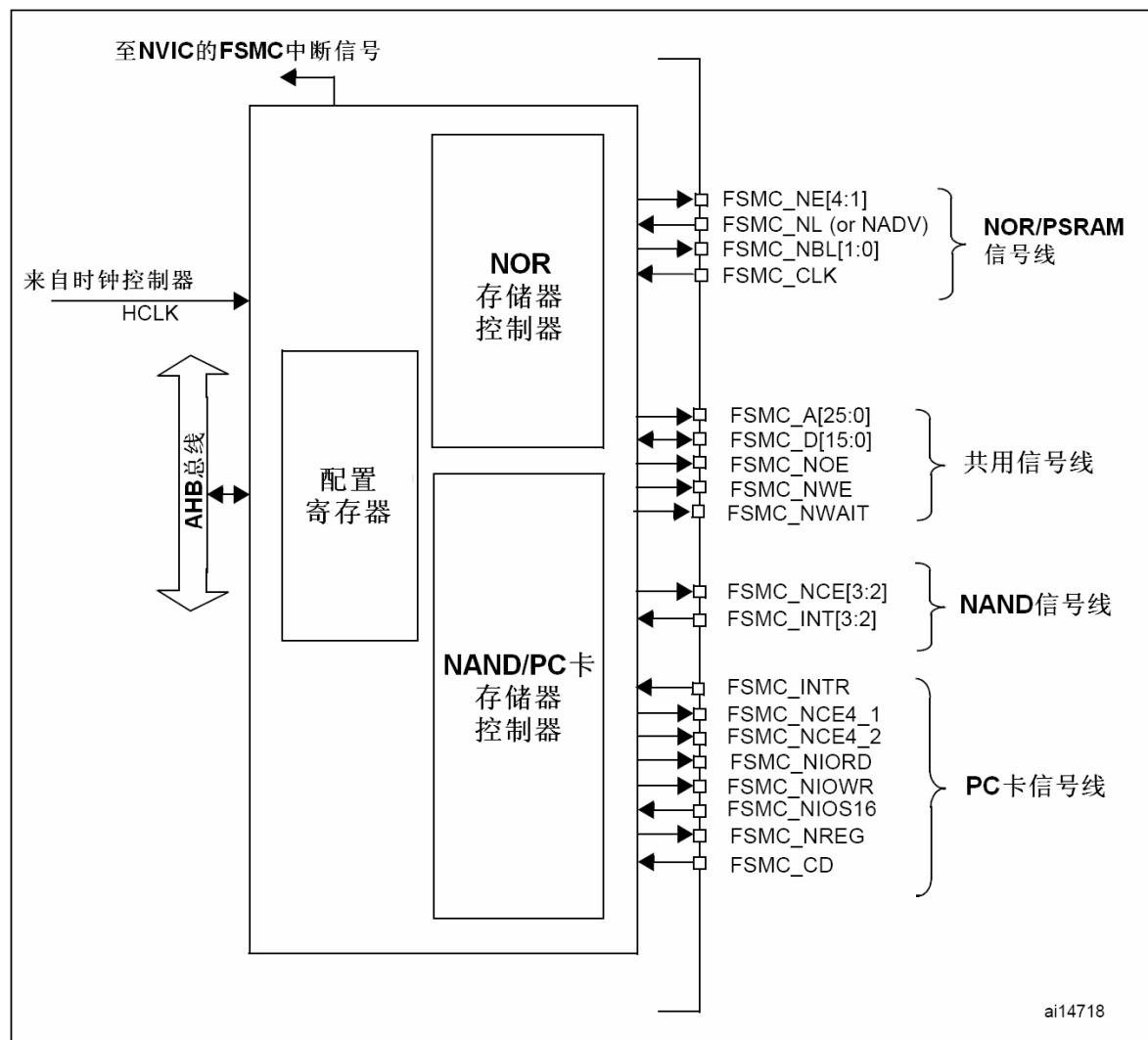
- 一个NOR闪存/SRAM控制器，可以与NOR闪存、SRAM和PSRAM存储器接口。
- 一个NAND闪存/PC卡控制器，可以与NAND闪存、PC卡、CF卡和CF+存储器接口。

控制器产生所有驱动这些存储器的信号时序：

- 16个数据线，用于连接8位或16位存储器
- 26个地址线，最多可连接64M字节的存储器(译注：这里不包括片选线)
- 5个独立的片选信号线
- 一组适合不同类型存储器的控制信号线：
 - 控制读/写操作
 - 与存储器通信，提供就绪/繁忙信号和中断信号
 - 与所用配置的PC卡接口：PC存储卡、PC I/O卡和真正的IDE接口

下面是FSMC的框图。

图1 FSMC框图

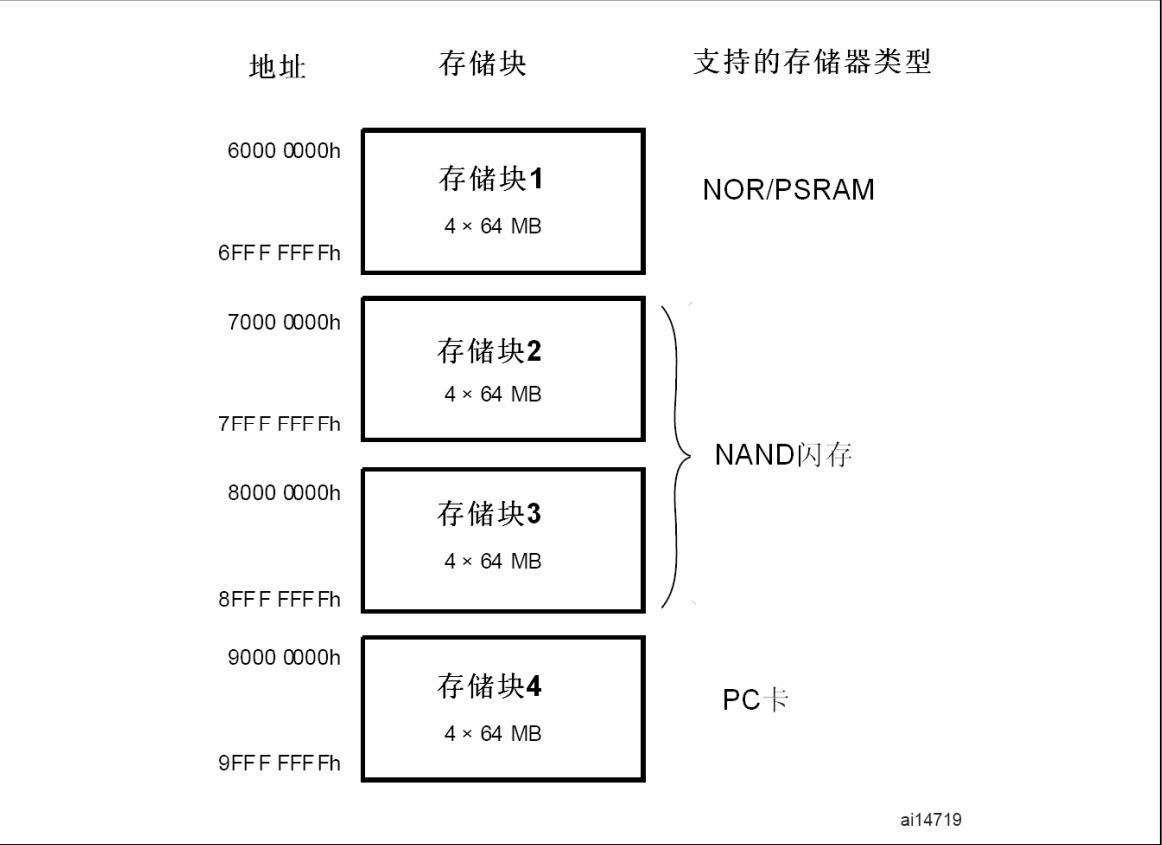


从FSMC的角度看，外部存储器分为4个固定大小为256M字节的存储块，如图2所示。

- NOR闪存/SRAM控制器使用存储块1访问4个存储器设备，这个存储块被划分为4个区域，具有4个专用的片选信号。
- NAND闪存/PC卡控制器使用存储块2和3访问NAND闪存设备。
- NAND闪存/PC卡控制器使用存储块4访问PC卡设备。

对于每个存储块，使用的存储器类型是用户通过配置寄存器定义的。

图2 FSMC存储块



2 与非总线复用模式的异步16位NOR闪存接口

2.1 FSMC配置

控制一个NOR闪存存储器，需要FSMC提供下述功能：

- 选择合适的存储块映射NOR闪存存储器：共有4个独立的存储块可以用于与NOR闪存、SRAM和PSRAM存储器接口，每个存储块都有一个专用的片选管脚。
- 使用或禁止地址/数据总线的复用功能。
- 选择所用的存储器类型：NOR闪存、SRAM或PSRAM。
- 定义外部存储器的数据总线宽度：8或16位。
- 使用或关闭同步NOR闪存存储器的突发访问模式。
- 配置等待信号的使用：开启或关闭，极性设置，时序配置。
- 使用或关闭扩展模式：扩展模式用于访问那些具有不同读写操作时序的存储器。

因为NOR闪存/SRAM控制器可以支持异步和同步存储器，用户只须根据存储器的参数配置使用到的参数。

FSMC提供了一些可编程的参数，可以正确地与外部存储器接口。依存储器类型的不同，有些参数是不需要的。

当使用一个外部异步存储器时，用户必须按照存储器的数据手册给出的时序数据，计算和设置下列参数：

- ADDSET：地址建立时间
- ADDHOLD：地址保持时间
- DATAST：数据建立时间
- ACCMOD：访问模式

这个参数允许FSMC可以灵活地访问多种异步的静态存储器。共有4种扩展模式允许以不同的时序分别读写存储器。

在扩展模式下，FSMC_BTR用于配置读操作，FSMC_BWR用于配置写操作。(译注：如果读时序与写时序相同，只须使用FSMC_BTR即可。)

如果使用了同步的存储器，用户必须计算和设置下述参数：

- CLKDIV：时钟分频系数
- DATLAT：数据延时

如果存储器支持的话，NOR闪存的读操作可以是同步的，而写操作仍然是异步的。

当对一个同步的NOR闪存编程时，存储器会自动地在同步与异步之间切换；因此，必须正确地设置所有的参数。

图3和图4示出了对于一个典型的NOR闪存不同的读写时序。

图3 异步NOR闪存读操作时序

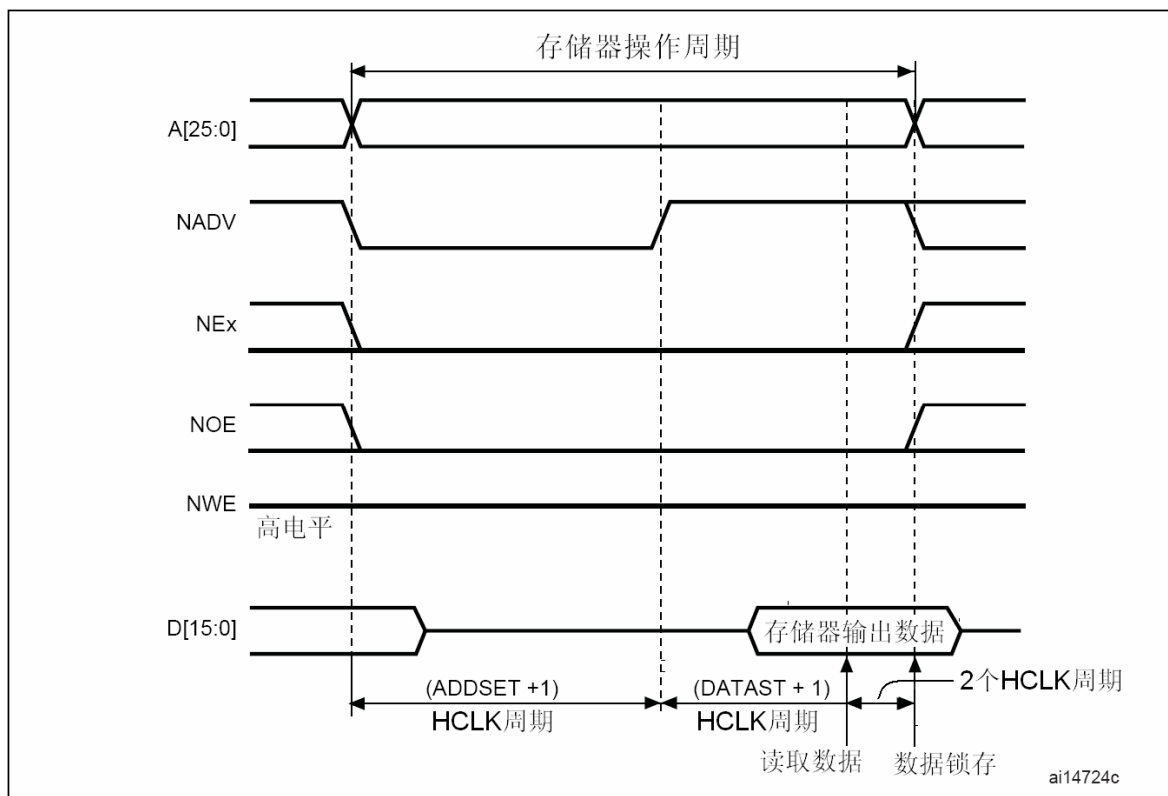
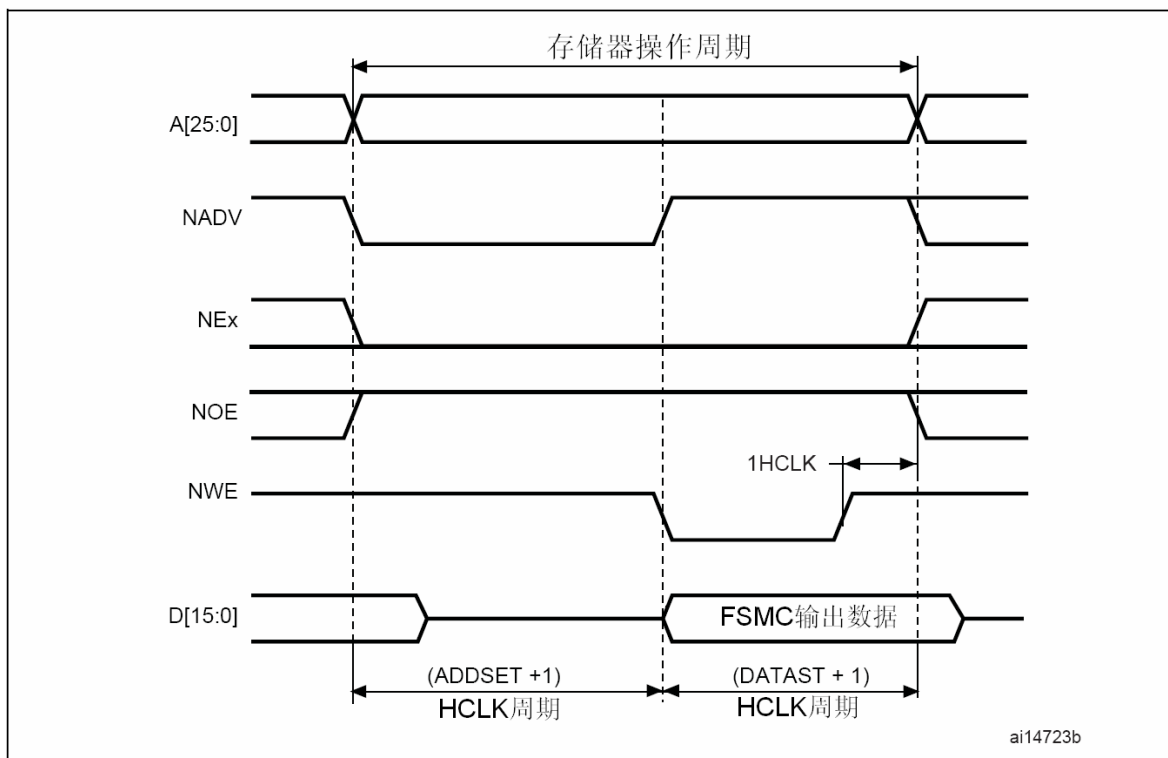


图4 异步NOR闪存写操作时序



2.1.1 与NOR闪存存储器接口的典型应用

STM32F10xxx的FSMC有4个各为64M字节的不同存储块，支持NOR闪存、PSRAM存储器和相同的外部存储器。

所有外部存储器共用控制器的地址、数据和控制信号线，每个外部设备由唯一的片选信号区分，而FSMC在任一时刻只访问一个外部设备。

每个存储块都有一组专用的寄存器，配置不同的功能和时序参数。

本文以M29W128FL存储器作为参考。M29W128FL是一个16位、异步、非总线共享的NOR闪存存储器，因此FSMC应按下述方式配置：

选用存储块2驱动这个NOR闪存存储器：

- 使能存储块2：设置BCR2_MBKEN位为'1'
- 存储器类型为NOR：设置BCR2_MTYP为'10'，选择NOR存储器类型
- 数据总线宽度为16位：设置BCR2_MWID为'01'，选择16位宽度
- 这是非总线共享存储器：清除BCR2_MUXEN为'0'

保持其它的所有参数为清除状态。

2.2 时序计算

如上所述，对于异步NOR闪存存储器或类似的存储，有不同的访问协议。首先要确定对特定存储器所需要使用的操作协议，选择的依据是不同的控制信号和存储器在读或写操作中的动作。

对于异步NOR闪存存储器，需要使用模式2协议。如果要使用的存储器有NADV信号，则需要使用扩展的模式B协议。

我们将使用模式2操作M29W128FL，不使用任何扩展模式，即读和写操作的时序是一样的。这时NOR闪存控制器需要3个时序参数：ADDSET、DATAST和ADDHOLD。

需要根据NOR闪存存储器的特性和STM32F10xxx的时钟HCLK来这些计算参数。

基于图3和图4的NOR闪存存储器访问时序，可以得到下述公式：

写或读访问时序是存储器片选信号的下降沿与上升沿之间的时间，这个时间可以由FSMC时序参数的函数计算得到：

$$\text{写/读访问时间} = ((\text{ADDSET} + 1) + (\text{DATAST} + 1)) \times \text{HCLK}$$

在写操作中，DATAST用于衡量写信号的下降沿与上升沿之间的时间参数：

$$\text{写使能信号从低变高的时间} = t_{\text{WP}} = \text{DATAST} \times \text{HCLK}$$

为了得到正确的FSMC时序配置，下列时序应予以考虑：

- 最大的读/写访问时间
- 不同的FSMC内部延迟
- 不同的存储器内部延迟

因此得到：

$$((\text{ADDSET} + 1) + (\text{DATAST} + 1)) \times \text{HCLK} = \max(t_{\text{WC}}, t_{\text{RC}})$$

$$\text{DATAST} \times \text{HCLK} = t_{\text{WP}}$$

DATAST必须满足：

$$\text{DATAST} = (t_{\text{AVQV}} + t_{\text{su(Data_NE)}} + t_{\text{v(A_NE)}})/\text{HCLK} - \text{ADDSET} - 4$$

下表列出了NOR闪存存储器的参数含义和数值：

表1 NOR闪存存储器时序

符号	参数	数值	单位
t_{WC}	地址有效至下一个写操作的地址有效	70	ns
t_{RC}	地址有效至下一个读操作的地址有效	70	ns
t_{WP}	写使能低至写使能高	45	ns
t_{AVQV}	地址有效至输出有效	70	ns

表2 STM32F10xxx参数

符号	参数	数值	单位
HCLK	内部AHB时钟频率	72	MHz
$t_{su(Data_NE)}$	数据至FSMC_NEx高的建立时间	-	ns
$t_{v(A_NE)}$	FSMC_NEx低至FSMC_A有效	-	ns
$t_{su(Data_NE)} + t_{v(A_NE)}$	数据至FSMC_NEx高的建立时间 + FSMC_NEx低至FSMC_A有效	36	ns

使用上述公式、存储器的时序(表1)和STM32F10xxx的参数(表2)，得到：

- 地址建立时间：0x0
- 地址保持时间：0x0
- 数据建立时间：0x6

注：对于S29GL128P NOR闪存存储器，时序为：

- 地址建立时间：0x5
- 地址保持时间：0x0
- 数据建立时间：0x7

2.3 硬件连接

表3给出了NOR闪存存储器与FSMC管脚的对应关系，和每个FSMC管脚的配置。

如果使用8位的NOR闪存存储器，数据总线是8位的，不需要连接FSMC的D8~D15。

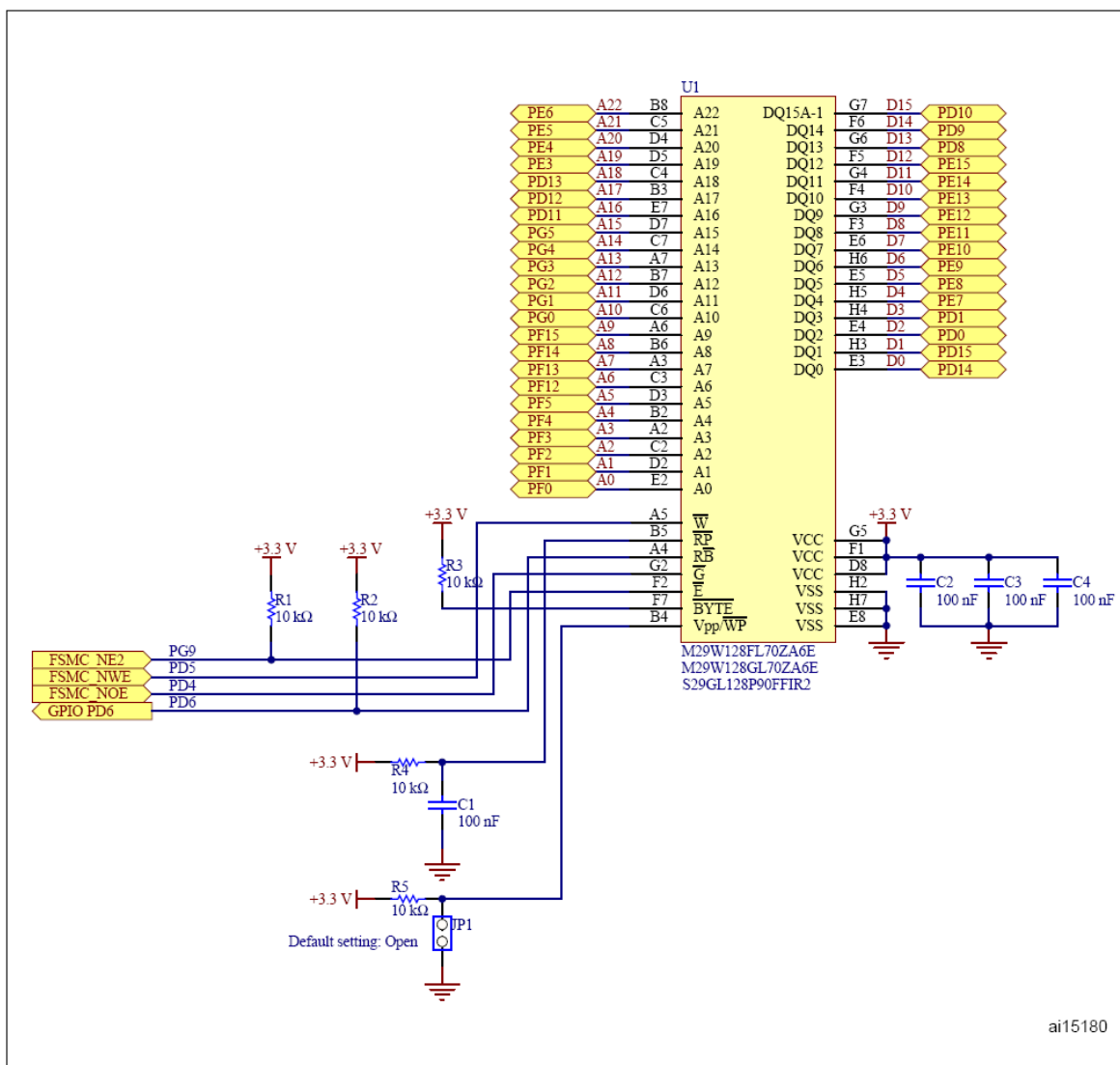
如果使用同步的存储器，FSMC_CLK管脚要接到存储器时钟管脚上。

表3 M29W128FL信号至FSMC管脚的对应

存储器信号	FSMC信号	管脚/端口分配	管脚/端口配置	信号说明
A0~A22	A0~A22	端口F/端口G/端口E/端口D	复用推挽输出	地址线A0至A22
DQ0~DQ7	D0~D7	端口D/端口E	复用推挽输出	数据线D0至D7
DQ8~DQ14	D8~D14	端口D/端口E	复用推挽输出	数据线D8至D14
DQ15A-1	D15	PD10	复用推挽输出	数据线D15
\overline{E}	NE2	PG9	复用推挽输出	芯片使能
\overline{G}	NOE	PD4	复用推挽输出	输出使能
\overline{W}	NEW	PD5	复用推挽输出	写使能

下图是一个典型的STM32F10xxx与M29W128FL NOR闪存存储器连接图，这是STM3210E-EVAL(STM32F10xxx评估板)的部分线路图。

图5 16位NOR闪存：M29W128FL/GL连接至STM32F10xxx



注：PD6管脚是用于为NOR闪存存储器提供就绪/繁忙输出信号(这种情况下，应用程序应该查询这个管脚的状态以确保正确的操作)。

在下面的例子中，M29W128FL和M29W128GL没有用到这个管脚，但是S29GL128P需要使用它。

在STM32F10xxx固件库的NOR目录下——STM32F10xFWLib\FWLib\examples\FSMC\NOR，有一个NOR闪存的例程。

这个例程的目的是示范如何使用FSMC的固件库函数和相应的NOR闪存驱动，在ST的评估板STM3210E-EVAL上对M29W128FL、M29W128GL或S29GL128P执行擦除/读/写操作。

在FSMC的时序方面，FSMC的NOR闪存驱动使用了最大的时序参数，即对应于S29GL128P NOR闪存存储器。这样，这个驱动程序可以支持所有适合STM3210E-EVAL板上的NOR存储器(译注：不同批次的STM3210E-EVAL板可能安装了不同的NOR闪存存储器)。

2.4 从外部NOR闪存存储器执行代码

大容量的STM32F10xxx内置了多达512K字节的闪存存储器，对于多数应用是足够了。需要更多存储器容量的应用，可以使用外加的NOR闪存存储器。

本节说明了如何使用外部NOR闪存存储器运行用户程序。这需要2个重要的步骤：

- 加载用户程序至外部NOR存储器:

这个操作需要对开发工具进行特别的配置: 在链接文件中, 必须指定NOR闪存存储器的开始地址(或任何其它地址), 这是需要放置用户程序的地址。

- 执行用户代码:

一旦用户程序代码加载到NOR闪存存储器, 在内部闪存存储器中需要有一段配置FSMC的程序, 配置好FSMC后可以跳转至(NOR闪存存储器中的)用户程序代码执行。

在STM32F10xxx固件库中有一个外部NOR闪存存储器中运行程序的例程, 例程的路径如下: STM32F10xFWLib\FWLib\examples\FSMC\NOR_CodeExecute。这个例程是把GPIO翻转IO的例子拷贝到STM3210E-EVAL评估板的扩展NOR闪存存储器中, 并执行这个例程。

有关如何在你的开发工具上使用这个例程的详细内容, 请参考上述路径下的readme文件。

3 与非总线复用的16位SRAM接口

3.1 FSMC配置

SRAM存储器和NOR闪存存储器共用相同的FSMC存储块，所用的协议依不同的存储器类型而有所不同。

控制SRAM存储器，FSMC应该具有下述功能：

- 使用或禁止地址/数据总线的复用功能。
- 选择所用的存储器类型：NOR闪存、SRAM或PSRAM。
- 定义外部存储器的数据总线宽度：8或16位。
- 使用或关闭扩展模式：扩展模式用于访问那些具有不同读写操作时序的存储器。

正如配置NOR闪存存储器一样，用户必须按照SRAM存储器的数据手册给出的时序数据，计算和设置下列参数：

- ADDSET：地址建立时间
- ADDHOLD：地址保持时间
- DATAST：数据建立时间

图6和图7示出了对于一个典型的SRAM存储器不同的读写时序

图6 异步SRAM存储器读操作时序

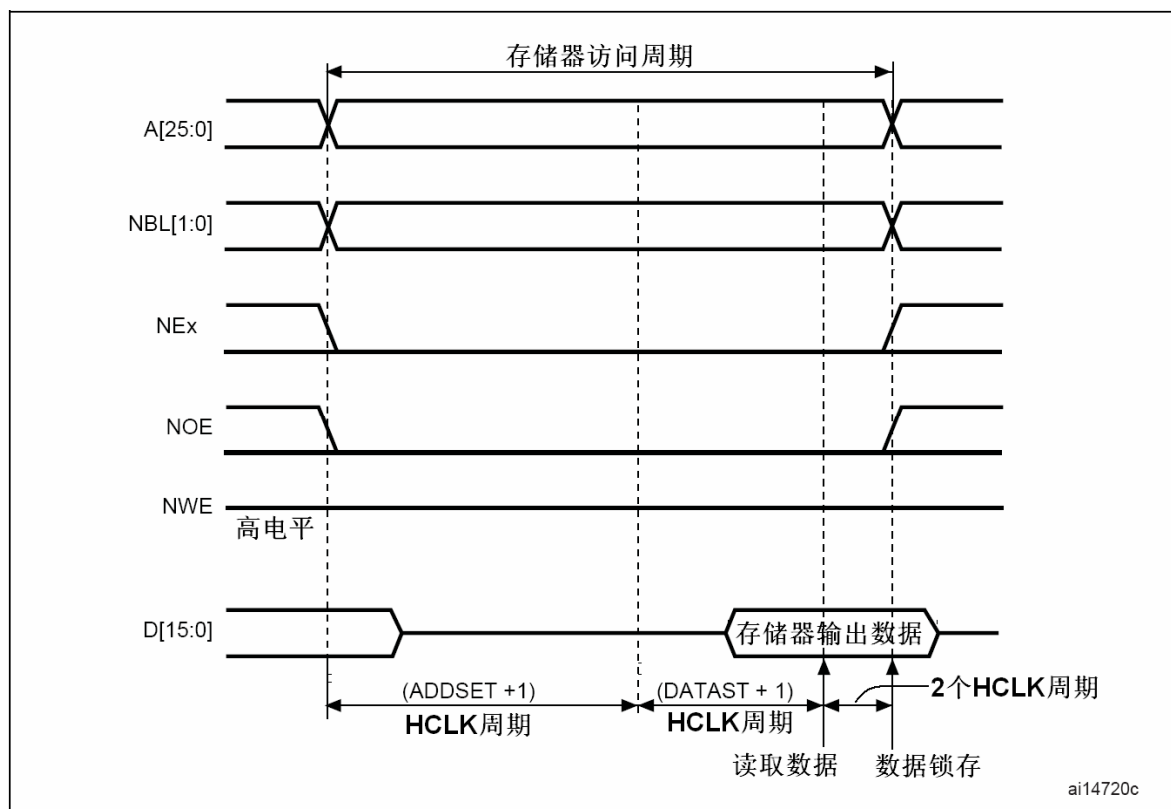
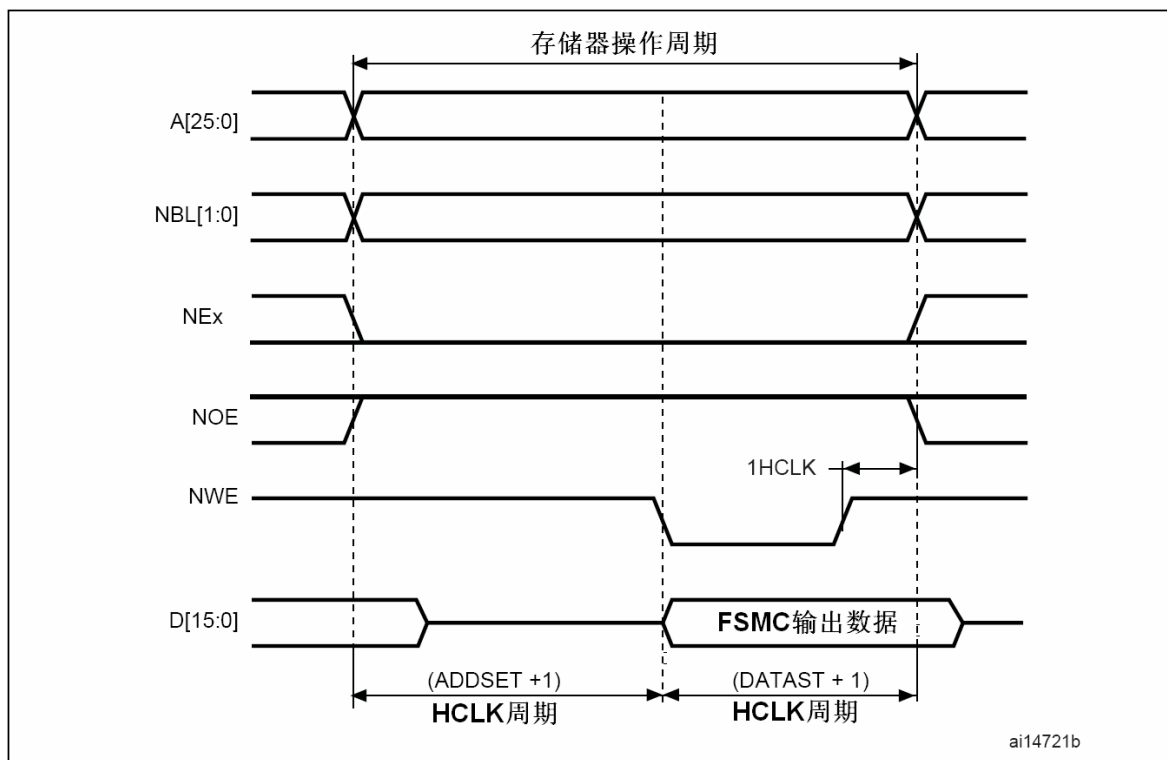


图7 异步SRAM存储器写操作时序



3.1.1 使用FSMC与SRAM存储器接口的典型应用

本文中使用IS61WV51216BLL存储器作为例子说明。

IS61WV51216BLL是一个非总线复用、异步的16位存储器。选用存储块3作为SRAM的接口，FSMC配置如下：

- 选用存储块3：BCR3_MBKEN设置为'1'。
- 存储器类型为SRAM：BCR3_MTYP设置为'00'，选择SRAM类型。
- 数据总线为16位：BCR3_MWID设置为'01'，选择16位宽。
- 存储器为非总线复用：清除BCR3_MUXEN为'0'。

保持其它的所有参数为清除状态。

3.2 时序计算

SRAM与NOR闪存存储器共用相同的存储块和配置寄存器，因此时序的计算方法与NOR闪存的计算相同(见2.2节)。

基于图6和图7中SRAM访问时序的描述，FSMC的配置需要考虑下述因素：

- 最大的读/写访问时间
- 不同的FSMC内部延迟
- 不同的存储器内部延迟

因此得到：

$$((ADDSET + 1) + (DATAST + 1)) \times HCLK = \max(t_{WC}, t_{RC})$$

$$DATAST \times HCLK = t_{PWE1}$$

DATAST必须满足：

$$DATAST = (t_{AA} + t_{su(Data_NE)} + t_{v(A_NE)})/HCLK - ADDSET - 4$$

下表列出了SRAM存储器的参数含义和数值：

表4 SRAM存储器时序

符号	参数	数值	单位
t _{WC}	写周期时间	12	ns
t _{RC}	读周期时间	12	ns
t _{PWE1}	写使能低脉冲宽度	8	ns
t _{AA}	地址有效时间	12	ns

使用上述公式、存储器的时序(表4)和STM32F10xxx的参数(表2)，可以得到：

- 地址建立时间：0x0
- 地址保持时间：0x0
- 数据建立时间：0x2

3.3 硬件连接

表5给出了SRAM存储器与FSMC管脚的对应关系，和每个FSMC管脚的配置。

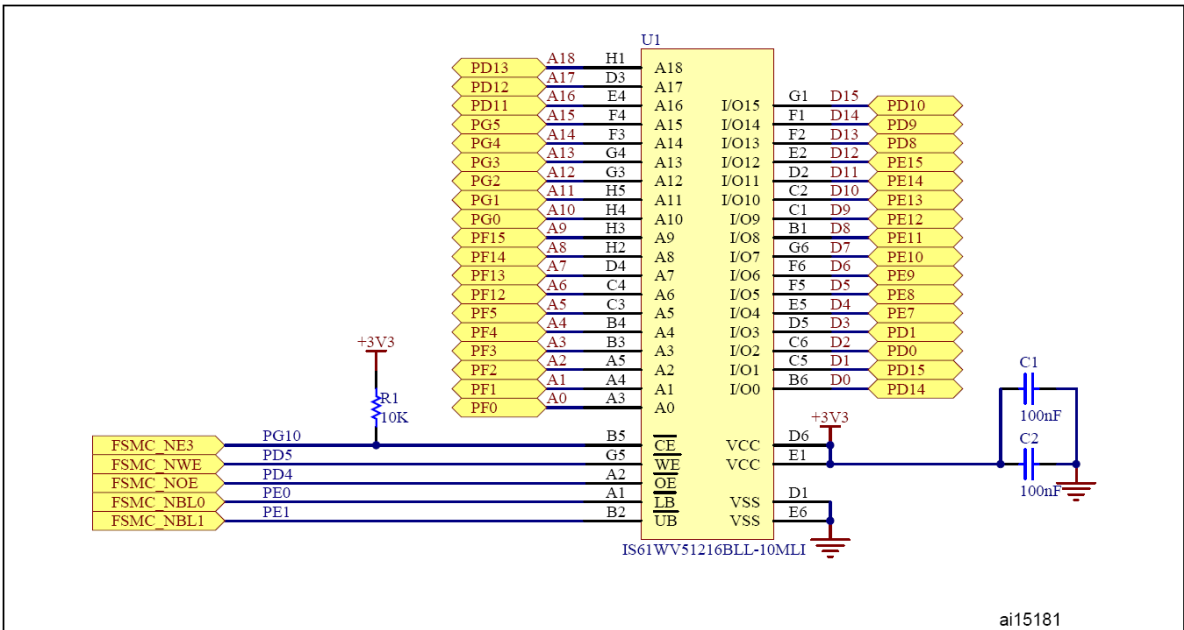
如果使用8位的SRAM存储器，数据总线是8位的，不需要连接FSMC的D8~D15。

表5 IS61WV51216BLL信号至FSMC管脚的对应

存储器信号	FSMC信号	管脚/端口分配	管脚/端口配置	信号说明
A0~A18	A0~A18	端口F/端口G/端口E/端口D	复用推挽输出	地址线A0至A18
I/O0~I/O15	D0~D15	端口D/端口E	复用推挽输出	数据线D0至D15
$\overline{\text{CE}}$	NE3	PG10	复用推挽输出	芯片使能
$\overline{\text{OE}}$	NOE	PD4	复用推挽输出	输出使能
$\overline{\text{WE}}$	NWE	PD5	复用推挽输出	写使能
$\overline{\text{LB}}$	NBL0	PE0	复用推挽输出	低字节控制
$\overline{\text{UB}}$	NBL1	PE1	复用推挽输出	高字节控制

下图是一个典型的STM32F10xxx与IS61WV51216BLL SRAM存储器连接图，这是STM3210E-EVAL(STM32F10xxx评估板)的部分线路图。

图8 16位SRAM：IS61WV51216BLL连接至STM32F10xxx



在STM32F10xxx固件库的SRAM目录下有一个SRAM的例程：

STM32F10xFWLib\FWLib\examples\FSMC\SRAM

这个例程的目的是示范如何使用FSMC的固件库函数和相应的SRAM驱动，在ST的评估板STM3210E-EVAL上对IS61WV51216BLL执行读/写操作。

3.4 使用外部SRAM作为数据存储器

在需要大容量的读写数据存储的应用中，接到FSMC的外部SRAM可以作为数据存储器使用。

使用外部SRAM作为数据存储器使用的配置步骤，与用户使用的开发工具和硬件环境相关。

在STM32F10xxx固件库中有一个使用外部SRAM存储器作为数据存储器的例程，路径如下：

STM32F10xFWLib\FWLib\examples\FSMC\SRAM_DataMemory

这个例程示范了如何使用STM3210E-EVAL板上的外部SRAM存储器作为程序的数据存储器、使用内部SRAM作为堆栈。关于如何在你的开发工具下使用这个例程的细节，请参考上述路径下的readme文件。

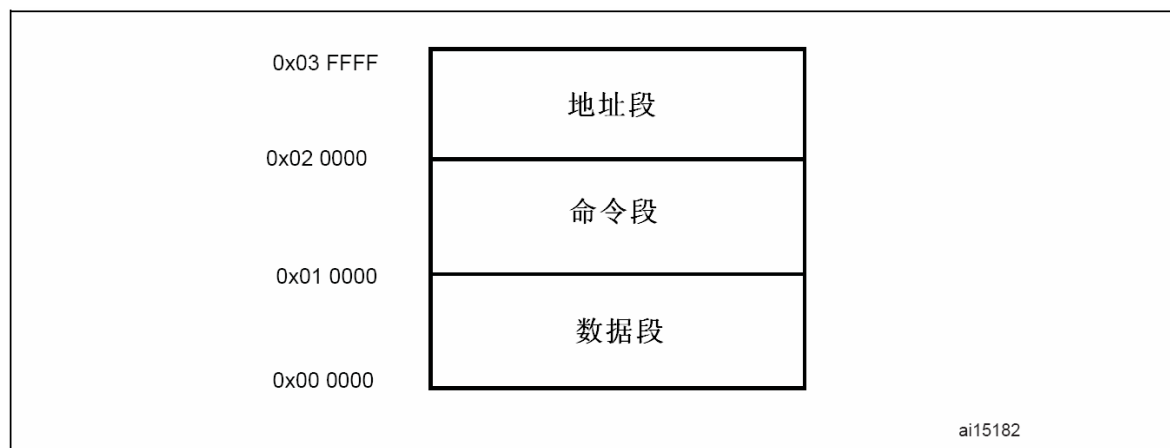
4 与8位的NAND闪存存储器接口

操作NAND闪存存储器，需要使用特别的访问协议，所有的读写操作，需要有下列步骤：

1. 向NAND闪存存储器发送一个命令
2. 发送读或写的地址
3. 读出或写入数据

为了使用户可以方便地操作NAND闪存，FSMC的NAND存储块被划分为3个段：数据段、地址段和命令段。

图9 FSMC的NAND存储块的段划分



实际上，这3个段的划分反映了真实的NAND闪存存储器的结构。写入命令段的任何地址，结果都是向NAND闪存写入命令。写入地址段的任何地址，结果都是向NAND闪存写入读写操作的地址；根据所用NAND闪存的构造，通常需要4~5个写入地址段才能写入一个读写操作的地址。写入或读出数据段的任何地址，结果都是写入或读出NAND的内部单元，该单元的地址是之前在地址段写入的那个地址。

4.1 FSMC配置

为控制NAND闪存存储器，FSMC提供下述功能：

- 开启或关闭存储器就绪/繁忙(Ready/Busy)信号作为FSMC的输入等待。
- 开启或关闭存储器就绪/繁忙(Ready/Busy)信号作为FSMC的中断输入源：中断可以以下述3种方式产生：
 - 在就绪/繁忙信号的上升沿产生中断：存储器刚刚完成一个操作，新的状态已经就绪。
 - 在就绪/繁忙信号的下降沿产生中断：存储器开始一个新的操作。
 - 在就绪/繁忙信号为高电平时产生中断：存储器已经就绪。
- 选择NAND存储器的数据总线宽度：8或16位。
- 开启或关闭ECC计算逻辑。
- 指定ECC计算的页面大小：可以是256、512、1024、2048、4096或8192字节/页。

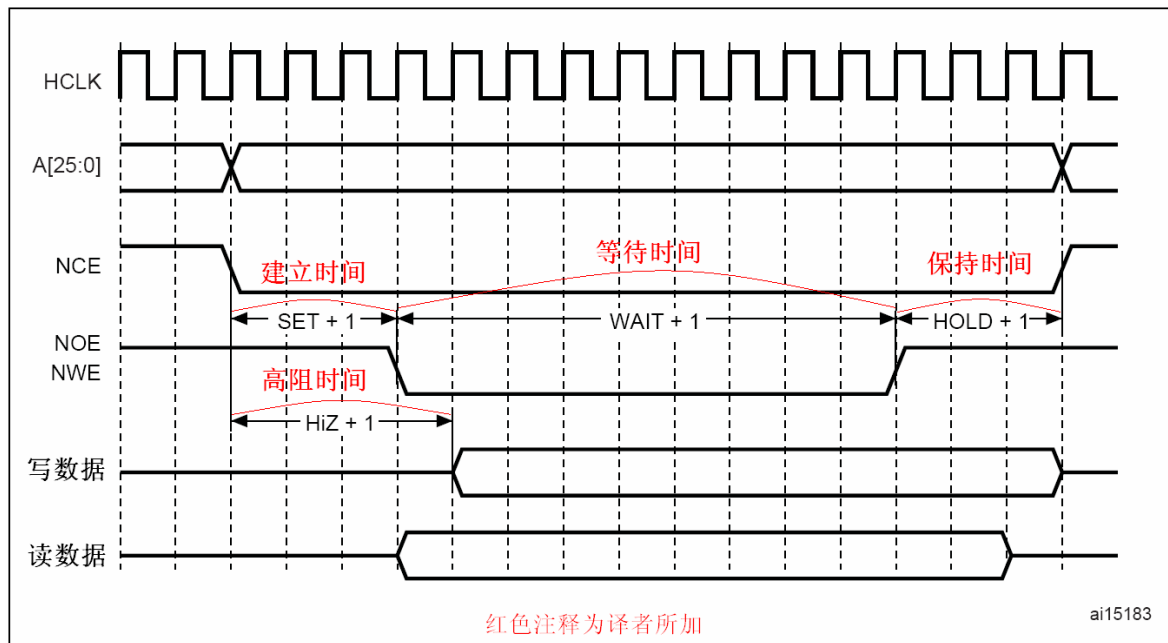
用户可以配置FSMC的时序分别满足NAND闪存的不同段的操作：公共段和属性段。可配置的时序是：

- **建立时间**：这是发送命令字之前地址的建立时间(以HCLK为单位)，即从地址有效至开始读写操作之间的时间。(译注：这里讲的读写操作是指对NAND内控制单元的读写，不一定是
对NAND中存储单元的操作)
- **等待时间**：这是发送命令字所需要的时间(以HCLK为单位)，即从NOE和NWE信号下降至上升之间的时间。
- **保持时间**：这是发送命令字后地址保持的时间(以HCLK为单位)，即从NOE和NWE信号下降至上升至整个操作周期结束的时间。

- **数据总线高阻时间**：这个参数只在写操作时有效，它是在开始写操作后数据总线保持高阻状态的时间(以HCLK为单位)，即从地址有效至FSMC驱动数据总线的时间。

下图显示了一个典型的NAND存储器访问的不同时序。

图10 NAND存储器访问时序



4.1.1 使用FSMC与NAND存储器接口的典型应用

FSMC的NAND闪存控制器通过存储块2和存储块3操作NAND存储器。

每一个存储块都有一个对应的片选信号。

在开始与NAND闪存通信之前，需要根据NAND闪存的特性初始化FSMC的NAND闪存控制器：功能、时序、数据总线宽度等。

本文以恒忆(Numonyx)公司的NAND512W3A存储器作为例子说明，这个产品与现在市场上大部分的NAND存储器具有相同的访问协议。(译注：[Numonyx公司](#)是由原Intel的闪存产品部和原ST的闪存产品部合并后，成立的一间专门生产销售闪存产品的公司。)

NAND512W3A的特性如下：

- **NAND接口**：8位总线宽度，复用的地址/数据线。
- **页大小**：(512 + 16)字节
- **页读/编程时序**：
 - 随机访问：12μs (3V)/15μs(1.8V) (最大)
 - 顺序访问：30ns (3V)/50ns(1.8V) (最小)
 - 页编程时间：200μs(典型值)

本例中，选择存储块2作为NAND闪存的接口。因此FSMC按照下述进行配置：

- **使能存储块2**：设置PCR2_PBKEN为'1'。
- **存储器类型为NAND闪存**：设置PCR2_PTyp为'1'。
- **数据总线宽度是8位**：设置PCR2_PWID为'00'。
- **ECC页大小为512字节**：设置PCR2_ECCPS为'001'。
- **ECC计算电路的开/关**：按需要设置PCR2_ECCEN为'1'或'0'。
- **根据用户应用的需要使用等待功能**：按需要设置PCR2_PWAITEN为'1'或'0'。

如果用户把NAND的就绪/繁忙信号连接至FSMC_NWAIT管脚，就需要使用等待功能管理NAND闪存的操作。

当使用NAND闪存的等待功能时，控制器将在开始一个新的访问之前，等待NAND闪存就绪，在等待期间，控制器会保持NCE信号一直有效(低电平)。

通常，就绪/繁忙信号是一个开路输出信号，将它连到STM32F10xxx微控制器时，对应的管脚必须配置为上拉输入模式。

FSMC还可以把就绪/繁忙信号作为一个中断源使用，这样CPU可以在NAND闪存操作的等待周期内执行其他的任务。把这个信号作为中断源使用时有3种用法，通过SR2寄存器中IREN、IFEN或ILEN位设置，可以选择就绪/繁忙信号的上升沿、下降沿或高电平触发中断。

4.2 时序计算

除了配置与NAND闪存相关的不同功能外，用户还需要初始化控制器以满足存储器的时序。

正如4.1节的说明，可以分别设置FSMC的公共空间和属性空间上的时序：建立时间、等待时间、保持时间和数据总线高阻时间。

这些参数需要根据NAND存储器的特性和STM32F10xxx的HCLK时钟计算。

根据图10显示的NAND存储器访问时序，可以得到下述公式：

读或写访问时间是NAND存储器的片选信号，从下降沿至上升沿之间的时间，这是FSMC时序参数的函数：

$$\text{读/写访问时间} = ((\text{SET} + 1) + (\text{WAIT} + 1) + (\text{HOLD} + 1)) \times \text{HCLK}$$

等待时间是读/写使能信号，从下降沿至上升沿之间的时间：

$$\text{读/写使能信号低至高时间} = (\text{WAIT} + 1) \times \text{HCLK}$$

对于读操作，数据总线高阻时间(HiZ)是由片选建立时间和数据建立时间衡量：

$$\text{片选建立时间} - \text{数据建立时间} = \text{HiZ} \times \text{HCLK}$$

保持时间参数可以在第一个公式中获得。实际上，NAND存储器的数据手册给出了写操作中片选低至写使能高的时序，保持时间可以由此计算得出：

$$\text{片选低至写使能高时间} = ((\text{SET} + 1) + (\text{WAIT} + 1)) \times \text{HCLK}$$

为了保证正确地配置FSMC的时序，下述因素应加以考虑：

- 最大读/写访问时间
- FSMC内部各部分的延迟
- 存储器内部各部分的延迟

因此，我们得到下述公式：

$$\begin{aligned} (\text{WAIT} + 1) \times \text{HCLK} &= \max(t_{\text{WP}}, t_{\text{RP}}) \\ ((\text{SET} + 1) + (\text{WAIT} + 1)) \times \text{HCLK} &= \max(t_{\text{CS}}, t_{\text{ALS}}, t_{\text{CLS}}) \\ \text{HOLD} &= \max(t_{\text{CH}}, t_{\text{ALH}}, t_{\text{CLH}}) / \text{HCLK} \end{aligned}$$

还需要满足下述公式的验证：

$$\begin{aligned} ((\text{SET} + 1) + (\text{WAIT} + 1) + (\text{HOLD} + 1)) \times \text{HCLK} &= \max(t_{\text{RC}}, t_{\text{WC}}) \\ \text{HiZ} &= (\max(t_{\text{CS}}, t_{\text{ALS}}, t_{\text{CLS}}) - t_{\text{DS}}) / \text{HCLK} - 1 \end{aligned}$$

考虑FSMC和存储器内部各部分的延迟，这些公式变为如下形式：

- WAIT需要满足：

$$\begin{aligned} (\text{WAIT} + 1 + \text{SET} + 1) &= ((t_{\text{CEA}} + t_{\text{su(Data_NE)}} + t_{\text{v(A_NE)}}) / \text{HCLK}) \\ \text{WAIT} &= ((t_{\text{CEA}} + t_{\text{su(Data_NE)}} + t_{\text{v(A_NE)}}) / \text{HCLK}) - \text{SET} - 2 \end{aligned}$$
- SET需要满足

$$\begin{aligned} (\text{SET} + 1) &= \max((t_{\text{CS}}, t_{\text{ALS}}, t_{\text{CLS}}) - \max(t_{\text{WP}}, t_{\text{RP}})) / \text{HCLK} - 1 \\ \text{SET} &= (\max(t_{\text{CS}}, t_{\text{ALS}}, t_{\text{CLS}}) - \max(t_{\text{WP}}, t_{\text{RP}})) / \text{HCLK} - 1 \end{aligned}$$

下表列出了NAND存储器各项参数的意义和时序。

表6 NAND闪存存储器时序

符号	参数	数值	单位
t_{CEA}	片选低至输出有效	35	ns
t_{WP}	写使能低至写使能高	15	ns
t_{RP}	写使能低至写使能高	15	ns
t_{CS}	片选低至写使能高	20	ns
t_{ALS}	AL建立时间	15	ns
t_{CLS}	CL建立时间	15	ns
t_{CH}	\bar{E} 保持时间	5	ns
t_{ALH}	AL保持时间	5	ns
t_{CLH}	CL保持时间	5	ns

使用上述公式、存储器时序(表6)和STM32F10xxx参数(表2)，我们得到：

- 地址建立时间：0x1
- 地址保持时间：0x3
- 数据建立时间：0x2
- 数据总线高阻时间：0x2

4.3 硬件连接

表7给出了NAND存储器与FSMC管脚的对应关系，和每个FSMC管脚的配置。

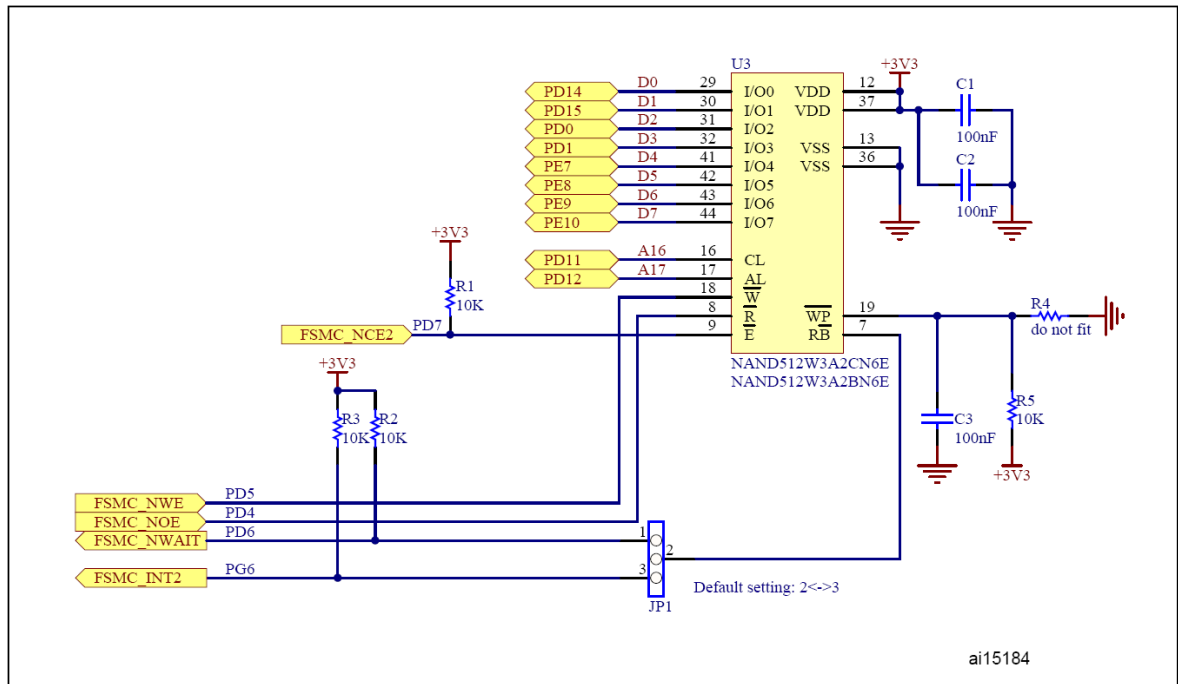
如果使用16位的NAND存储器，数据/地址总线是16位的，需要连接其它的FSMC数据/地址信号线。

表7 NAND512W3A信号至FSMC管脚的对应

存储器信号	FSMC信号	管脚/端口分配	管脚/端口配置	信号说明
AL	ALE/A17	PD11	复用推挽输出	地址锁存使能
CL	CLE/A16	PD12	复用推挽输出	命令锁存使能
I/O0~I/O7	D0~D7	端口D / 端口E	复用推挽输出	数据总线D0~D7
\bar{E}	NCE2	PD7	复用推挽输出	片选使能
\bar{R}	NOR	PD4	复用推挽输出	输出使能
\bar{W}	NWE	PD5	复用推挽输出	写使能
R/\bar{B}	NWAIT/INT2	PD6/PG6	输入上拉	就绪/繁忙信号

下图是STM32F10xxx微控制器与NAND512W3A存储器的一个典型的连接图，这是STM3210E-EVAL(STM32F10xxx评估板)的部分线路图。

图11 8位NAND闪存: NAND512W3A2C/NAND512W3A2B与STM32F10xxx的连接



在STM32F10xxx固件库的NAND目录下有一个例程：

STM32F10xFWLib\FWLib\examples\FSMC\NAND

这个例程的目的是示范如何使用FSMC的固件库函数和相应的NAND驱动，在ST的评估板STM3210E-EVAL上使用等待功能对NAND512W3A2执行擦除/读/写操作。

4.4 错误校验码计算

4.4.1 错误校验码(ECC)计算概述

FSMC的NAND闪存控制器中有2个错误校验码计算的硬件电路，分别用于2个NAND存储块。

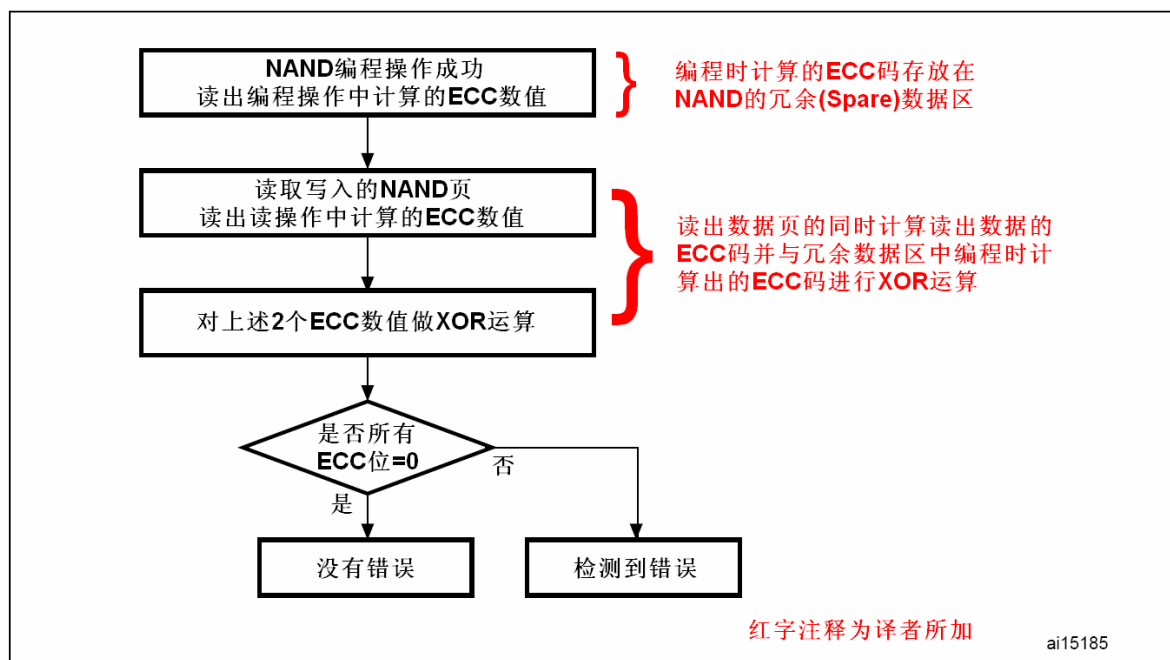
根据用户设置的页面大小，ECC电路可以计算每页256、512、1024、2048、4096或8192数据字节的错误校验码。依配置的页面大小，ECC码长度为22、24、26、28、30或32位。

为了更加提供错误检测的覆盖率，用户可以在读/写NAND闪存页时减小ECC计算时的页面大小，这可以通过在需要数目的数据字节长度处开始和停止ECC计算实现，ECC的计算只在写入或读出数据时进行。

FSMC中实现的错误校验码算法，可以实现在读出或写入一页**NAND**闪存数据中检测**1**位和**2**位错误。这个算法是基于海明算法，包括计算行和列的奇偶检验。

4.4.2 错误检测

图12 错误检测



在写操作时如果发生了错误，根据XOR运算的结果，有可纠正的错误和不可纠正的错误(译注：以下是以每页256字节为例说明)：

- 可纠正的错误
ECC码的XOR运算结果包含11位的数据'1'，同时每对的奇偶检验值是'10'或'0x01'。
- ECC码错误
ECC码的XOR运算结果只包含一个'1'。
- 不可纠正的错误
ECC码的XOR运算结果是一个随机数，此时不能纠正错误的数据。

根据图12的流程图，校验算法很容易实现。

第一步是检测写操作是否有错误；如果写操作有错误，则第二步是判断是否为可纠正的错误；如果是可纠正的错误，则第三步是纠正错误。

纠正错误是基于读操作时算出的ECC码，错误的位置可以从这个ECC码中识别出来。下述数据可以从ECC码中抽取出来：

P_{1024} 、 P_{512} 、 P_{256} 、 P_{128} 、 P_{64} 、 P_{32} 、 P_{16} 、 P_8 、 P_4 、 P_2 、 P_1 ，其中 P_x 是行和列的奇偶检验值。

对于8位的存储器， P_4 、 P_2 、 P_1 定义了错误位的位置，而 P_{1024} 、 P_{512} 、 P_{256} 、 P_{128} 、 P_{64} 、 P_{32} 、 P_{16} 、 P_8 定义了错误字节的位置。

译注：上述XOR的运算结果中，每2位(每对)代表一个奇偶检验值，从低位向高位依次定义为 P_1 、 P_2 、 P_4 、 P_8 、 P_{16} 、 P_{32} 、 P_{64} 、 P_{128} 、 P_{256} 、 P_{512} 、 P_{1024} ；如果ECC计算时，每页数据超过256字节，则还有 P_{2048} 、 P_{4096} 等。 P_x 与XOR运算结果的对应关系如下图所示：

奇偶检验值	P_{16}		P_8		P_4		P_2		P_1	
XOR的运算结果	位9	位8	位7	位6	位5	位4	位3	位2	位1	位0

- 如果每一个 P_x 的值都是'00'，则表示没有错误。
- 如果每一个 P_x 的值都是'10'或'01'，则表示有可纠正的错误。需要执行纠正。
- 如果所有 P_x 值中除了一个为'10'或'01'，其余的都是'00'，则表示ECC有错误但数据正确。
- 其它情况，则表示数据有错误，而且是不可纠正的错误。

进行错误纠正时， P_x 按下述规则取值：

P_x 对应的2位为'10'，则 $P_x=1$ ，否则 $P_x=0$

$P_{1024}P_{512}P_{256}P_{128}P_{64}P_{32}P_{16}P_8$ 构成的8位数指示了错误字节的位置。

$P_4P_2P_1$ 构成的3位数指示了错误字节中错误位的位置。

5 100脚的STM32F10xxx的FSMC配置

FSMC出现在144脚和100脚封装的产品中，然而，对于100脚封装的产品，因为管脚数目的限制，只可以使用部分的FSMC存储块。

5.1 FSMC与NAND存储器接口

在100脚封装的产品中没有NCE3管脚，只有FSMC的存储块2可以用于与8或16位的NAND存储器接口。

同样，在100脚封装的产品中没有2个中断管脚(INT2和INT3)，也不能使用中断功能。

下表显示了如何使用100脚封装的FSMC连接一个8或16位的NAND存储器。

表8 FSMC连接至NAND存储器

8或16位的NAND存储器管脚	FSMC管脚	100脚封装的管脚
\bar{E}	NCE2/NCE3	NCE2
\bar{R}	NOE	NOE
\bar{W}	NWE	NWE
AL	A17	A17
CL	A16	A16
R/ \bar{B}	NWAIT/INT2/INT3	NWAIT
I/O0~I/O7	D0~D7	D0~D7
I/O8~I/O15	D8~D15	D8~D15

5.2 FSMC与NOR存储器接口

在100脚封装的产品中，因为没有NE2、NE3和NE4管脚，所以只有FSMC的存储块1可以用于与NOR存储器接口。

同样，在100脚封装的产品中没有管脚A0~A15，用户必须使用NOR闪存控制器的总线复用模式，通过数据总线输出地址和数据。

下表显示了如何使用100脚封装的FSMC连接一个NOR存储器。

表9 FSMC连接至NOR存储器

8或16位的NOR存储器管脚	FSMC管脚	100脚封装的管脚
A0~A15	A0~A15	DA0~DA15
A16~A23	A16~A23	A16~A23
\bar{W}	NWE	NWE
\bar{E}	NE1/NE2/NE3/NE4	NE1
\bar{G}	NOE	NOE
DQ0~DQ14	D0~D14	D0~D14
DQ15A-1	D15	D15