AN2784
Application note

Using the high-density STM32F10xxx FSMC peripheral to drive external memories

Introduction

This application note describes how to use the High-density STM32F10xxx FSMC (flexible static memory controller) peripheral to drive a set of external memories. To that aim, it gives an overview of the STM32F10xxx FSMC controller. Then it provides memory interfacing examples that include the typical FSMC configuration, the timing computation method and the hardware connection.

This application note is based on memories mounted on the STM3210E-EVAL, which is the evaluation board for High-density STM32F10xxx devices. The used memories are a 16-bit asynchronous NOR Flash memory, an 8-bit NAND Flash memory and a 16-bit asynchronous SRAM.

The STM32F10xxx firmware library, the different memory drivers and examples of use for each of the memory types used in this application note, are available for download from the STMicroelectronics website: *www.st.com/mcu*.

Contents

List of tables

List of figures

1 Overview of the STM32F10xxx flexible static memory controller

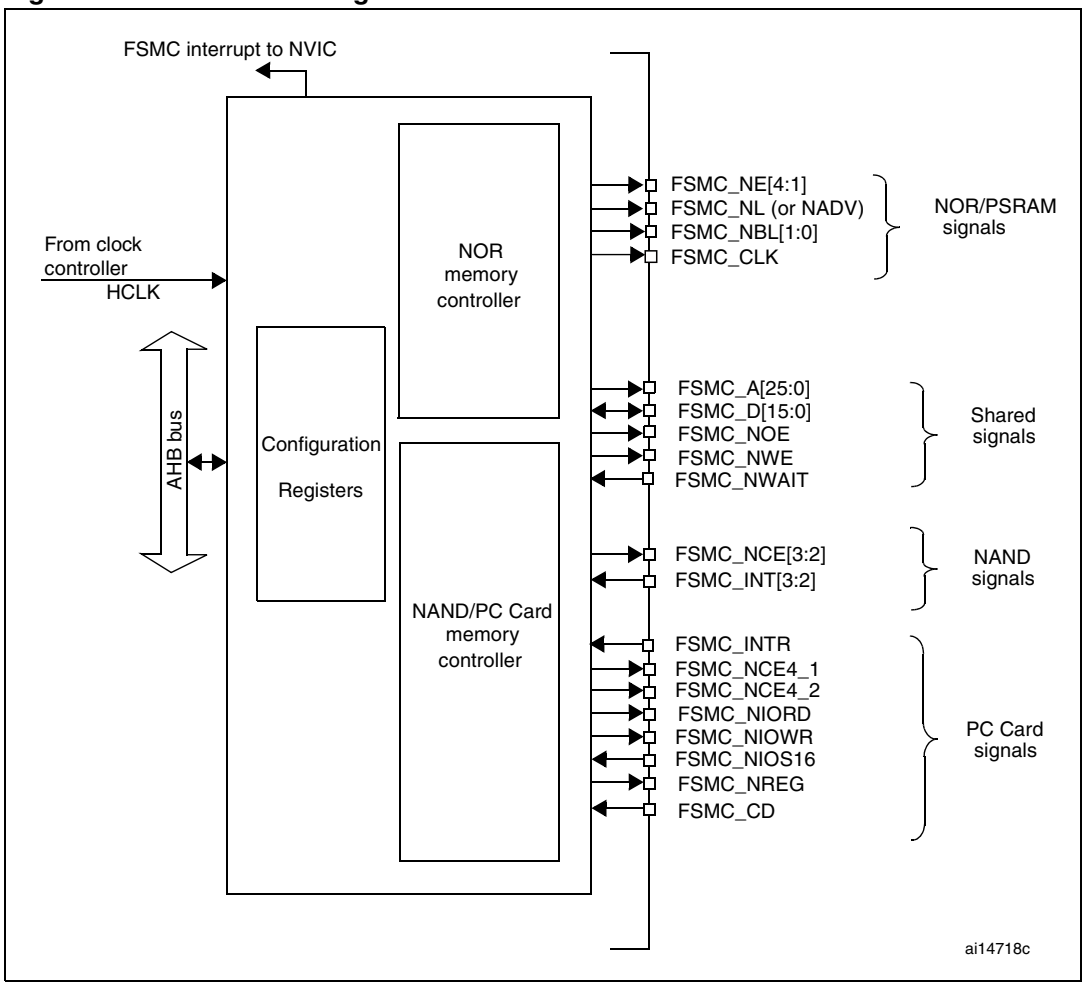
The FSMC has the following main features:

- Interfaces with static memory-mapped devices including:
  - Static random access memory (SRAM)
  - Read-only memory (ROM)
  - NOR Flash memory
  - PSRAM (4 memory banks)
- Two banks of NAND Flash with ECC hardware that checks up to 8 Kbytes of data
- 16-bit PC Card compatible devices
- Supports burst mode access to synchronous devices (NOR Flash and PSRAM)
- 8- or 16-bit wide databus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices, in particular:
  - Programmable wait states (up to 15)
  - Programmable bus turnaround cycles (up to 15)
  - Programmable output enable and write enable delays (up to 15)
  - Independent read and write timings and protocol, so as to support the widest variety of memories and timings
- Write enable and byte lane select outputs for use with PSRAM and SRAM devices
- Translation of 32-bit wide AHB transactions into consecutive 16-bit or 8-bit accesses to external 16-bit or 8-bit devices
- A Write FIFO, 2 words long, each word is 32 bits wide, only stores data and not the address. Therefore, this FIFO only buffers AHB write burst transactions. This makes it possible to write to slow memories and free the AHB quickly for other operations. Only one burst at a time is buffered: if a new AHB burst or single transaction occurs while an operation is in progress, first the FIFO is drained (The FSMC will insert wait states until the current memory access is complete).
- External asynchronous wait control

The FSMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, it is possible to change the settings at any time.

Figure 1 illustrates the FSMC block diagram.

Figure 1. FSMC block diagram

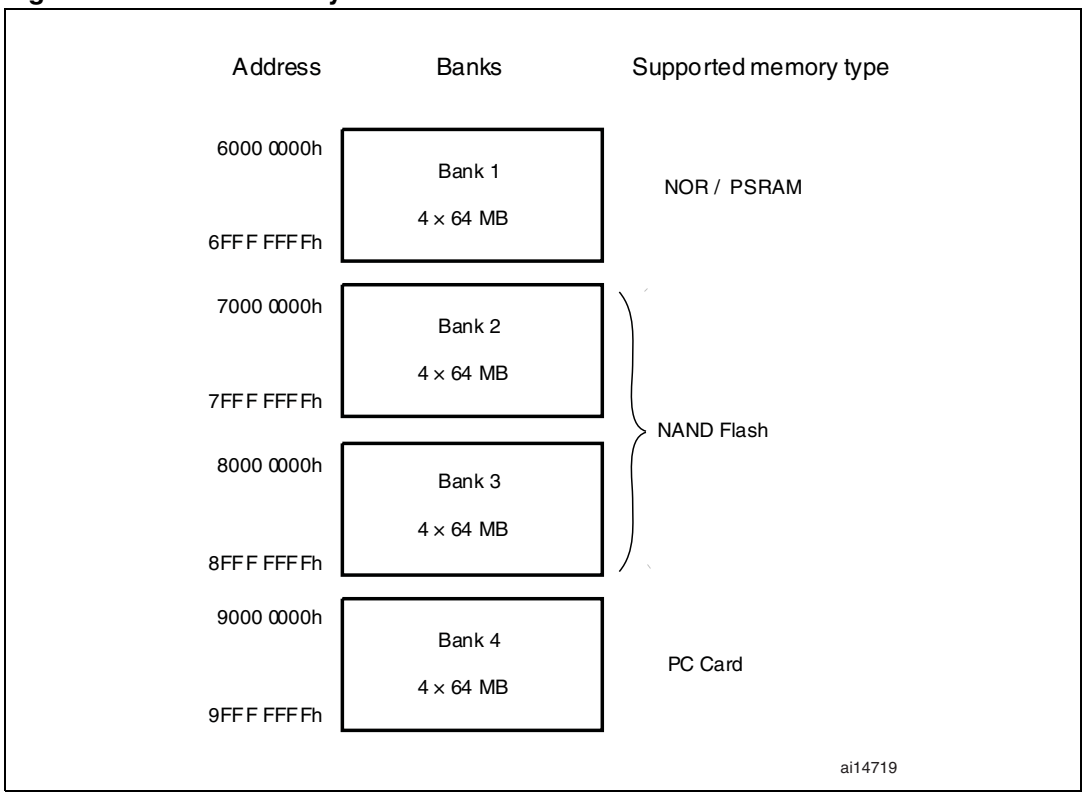


From the FSMC point of view, the external memory is divided into four fixed-size banks of 256 Mbytes each, as shown in *Figure 2*:

- Bank 1 used by the NOR Flash/SRAM controller to address up to 4 memory devices. This bank is split into 4 regions with 4 dedicated Chip Select signals.
- Banks 2 and 3 used by the NAND Flash/PC Card controller to address NAND Flash devices.
- Bank 4 used by the NAND Flash/PC Card controller to address a PC Card device.

For each bank, the type of memory to be used is user-defined in the Configuration register.

Figure 2. FSMC memory banks



1.1 Interfacing asynchronous static memories (NOR Flash, SRAM)

The interface signals are synchronized by the internal clock HCLK. This clock is not output to the memory

The FSMC always samples the data before de-asserting the chip select signal NE. This guarantees that the memory data-hold timing constraint is met (chip enable high to data transition, usually 0 ns min.).

- If extended mode is enabled (EXTMOD: bit is set in the FSMC_BCRx register), there are up to four extended modes (A, B, C and D) and it is possible to mix modes A, B, C and D in read and write access.
- If Extended mode is disabled, the FSMC operates in Mode1 or Mode2 as follows:
  - Mode 1 is the default mode when SRAM/CRAM memory type is selected (Bits 3:2 MTYP = 0x0 or 0x01 in the FSMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (Bits 3:2 MTYP = 0x10 in the FSMC_BCRx register) "

Table 1. FSMC operating modes

Asynchronous mode	Memory type	
	SRAM/CRAM	NOR
Extended mode disabled	Mode 1	Mode2
Extended mode enabled	Mode A	Mode B, C, D

Table 2. STM32F10xCDE FSMC asynchronous timings

Symbols	Parameter	Value	Unit
HCLK	Internal AHB clock frequency	72	MHz
t_{HCLK}	Internal AHB clock cycle	13.88	ns
$t_{su(Data\_NE)} + t_{v(A\_NE)}$	Data to FSMC_NEx high setup time + FSMC_NEx low to FSMC_A valid	$2t_{HCLK} + 25$	ns

Note: These timings extracted from STM32F103xCDE datasheet are needed for computing the NOR and SRAM asynchronous mode timings.

2 Interfacing with a nonmultiplexed, asynchronous 16-bit NOR Flash memory

2.1 FSMC configuration

To control a NOR Flash memory, the FSMC provides the following possible features:

- Select the bank to be used to map the NOR Flash memory: there are 4 independent banks which can be used to interface with NOR Flash/SRAM/PSRAM memories, each bank has a separate Chip Select pin.
- Enable or disable the address/data multiplexing feature
- Select the memory type to be used: NOR Flash/SRAM/PSRAM
- Define the external memory databus width: 8/16 bits
- Enable or disable the burst access mode for synchronous NOR Flash memories
- Configure the use of the wait signal: enable/disable, polarity setting and timing configuration.
- Enable or disable the extended mode: this mode is used to access the memory with a different timing configuration for read and write operations.

As the NOR Flash/PSRAM controller can support asynchronous and synchronous memories, the user should select only the used parameters depending on the memory characteristics.

The FSMC also provides the possibility of programming several parameters to correctly interface with the external memory. Depending on the memory type, some parameters are not used.

In the case where an external asynchronous nonmultiplexed memory is used, the user has to compute and set the following parameters depending on the information in the memory datasheet:

- ADDSET: address setup time
- DATAST: data setup time
- ACCMOD: access mode

  This parameter gives the FSMC the flexibility to access a wide variety of asynchronous static memories. There are four extended access modes (A, B, C and D) that allow write access while reading the memory with different timings, if the memory supports this kind of feature.

  When the extended mode is enabled, the FSMC_BTR register is used for read operations and the FSMC_BWR register is used for write operations.

In the case where a synchronous memory is used, the user has to compute and set the following parameters:

- CLKDIV: clock divide ratio
- DATLAT: data latency

Note that NOR Flash memory read operations can be synchronous if the memory supports this mode, while write operations usually remain asynchronous.

When programming a synchronous NOR Flash memory, the memory automatically switches between the synchronous and the asynchronous mode, so in this case, all parameters have to be set correctly.

Figure 3 and *Figure 4* show the different timings during a typical NOR Flash memory access.

Figure 3. Asynchronous NOR Flash read access timing

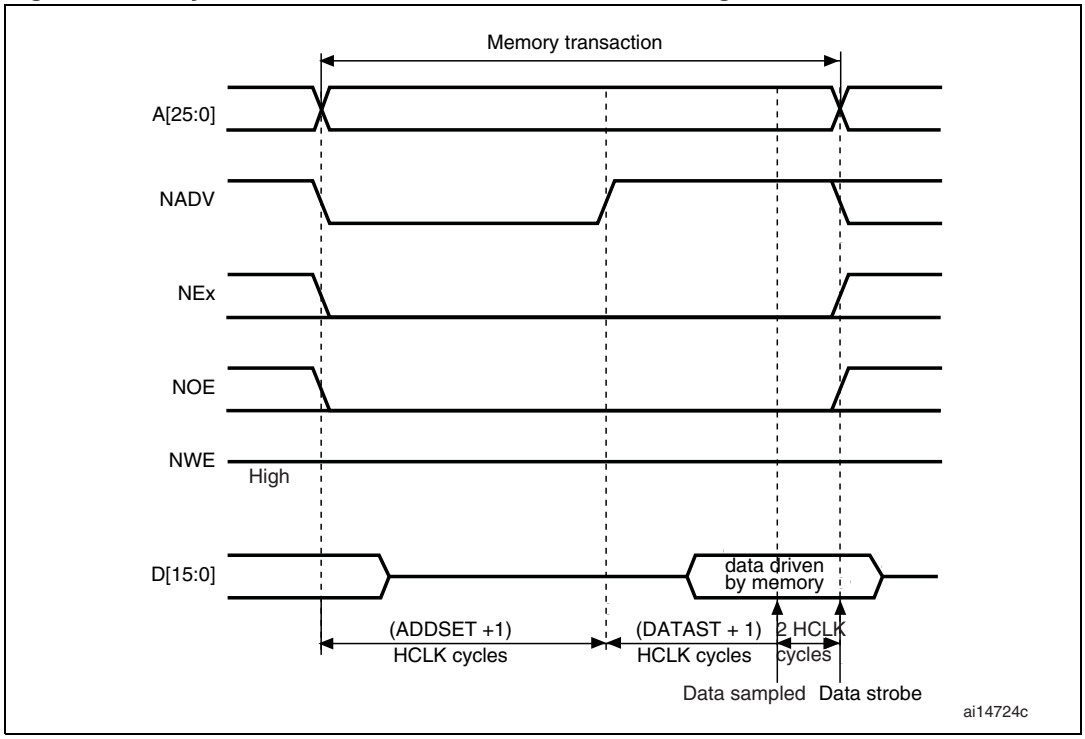
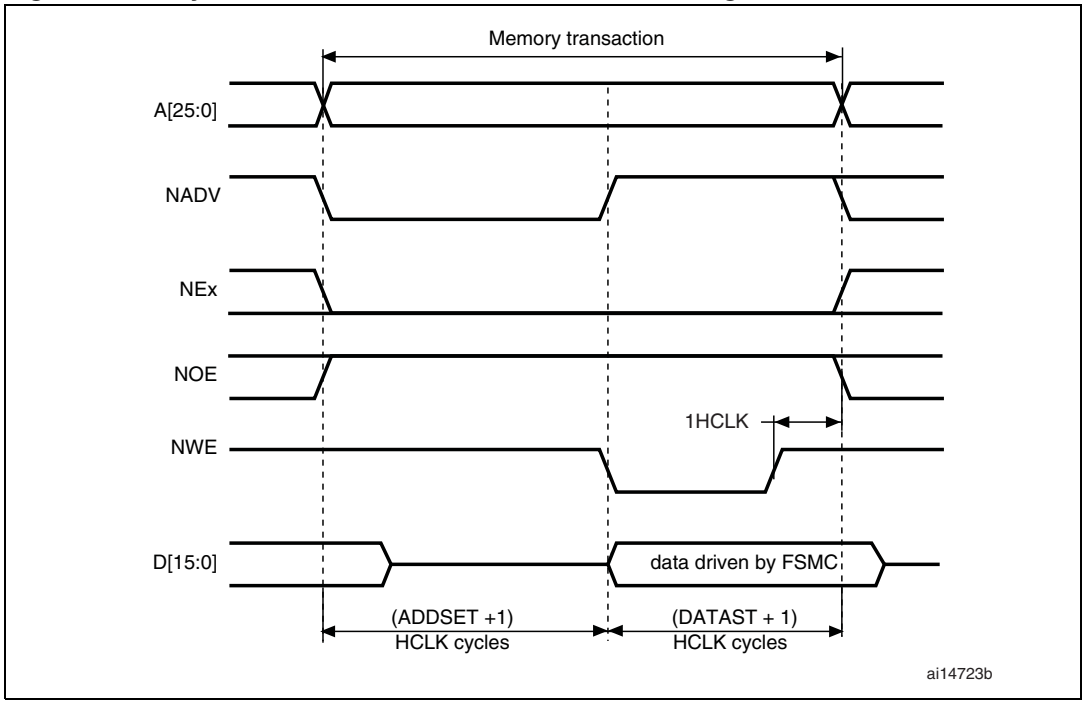


Figure 4. Asynchronous NOR Flash write access timing



2.1.1 Typical use of the FSMC to interface with a NOR Flash memory

The STM32F10xxx FSMC has four different banks of 64 Mbytes to support NOR Flash memories/PSRAMs and similar external memories.

The external memories share the address, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select signal, but the FSMC performs only one access at a time to an external device.

Each bank is configured by means of dedicated registers. Configuration includes the different features and the timing parameters.

In this application note, the M29W128FL memory is used as a reference. The M29W128FL memory is a 16-bit, asynchronous, nonmultiplexed NOR Flash memory. Based on these data, the FSMC is configured as follows:

Bank 2 is selected to support the NOR Flash memory device:

- Bank 2 is enabled: BCR2_MBKEN bit set to '1'
- Memory type is NOR: BCR2_MTYP is set to '10' to select the NOR memory type
- Databus width is 16-bit: BCR2_MWID is set to '01' to select the 16-bit width
- It is a nonmultiplexed memory: BCR2_MUXEN is reset.

All remaining parameters must be kept cleared.

2.2 Timing computation

As described above, for asynchronous NOR Flash-like memories, there are different possible access protocols. The first step is therefore to define the kind of protocol that should be used with the specific memory. The choice depends on the different control signals and the behavior of the memory during read and write transactions.

In the case of an asynchronous NOR Flash memory, the Mode 2 protocol will be used. If the used memory can provide an NADV signal, the Extended ModeB protocol should be used.

With the M29W129FL, we will use the Mode2 protocol. We will therefore not use any extended mode and the timings will be the same for read and write operations. In this case, the NOR memory controller needs two timing parameters: ADDSET and DATAST.

These parameters are computed according to the NOR Flash memory characteristics and according to the HCLK clock of the STM32F10xxx.

Based on the NOR Flash memory access timings illustrated in *Figure 3* and *Figure 4*, the following equations are found:

- The write or read access time is the time between the falling edge and the rising edge of the memory Chip Select signal. This time is computed as a function of the FSMC timing parameter:

  $((ADDSET + 1) + (DATAST + 1)) \times t_{HCLK} \geq$ Write access time
- In write operations, the DATAST parameter is measured between the falling edge and the rising edge of the write signal as follows:

  $t_{WP} = DATAST \times t_{HCLK} \geq$ Write Enable signal low to high

To have a correct configuration of the FSMC timings, the timings have to take into account:

- the maximum read/write access time
- the different internal FSMC delays
- the different internal memory delays

Hence, we have:

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

$DATAST \times HCLK = t_{WP}$

For read access, DATAST must verify:

$DATAST \geq (t_{AVQV} + t_{su(Data\_NE)} + t_{v(A\_NE)})/t_{HCLK} - ADDSET - 4$

Table 3 gives the meanings and values of the NOR Flash memory parameters.

Table 3. NOR Flash memory timings

Symbols	Parameter	Value		Unit
		M29W128xx70	S29GL128P90	
t_{WC}	Address valid to next address valid for write operation	70	90	ns
t_{RC}	Address valid to next address valid for read access	70	90	ns
t_{WP}	Write Enable low to Write Enable high	45	35	ns
t_{AVQV}	Address valid to output valid	70	90	ns

Using the memory timings in *Table 3* and the STM32F10xCDE FSMC asynchronous timings in *Table 2*, we can compute the following values:

- For the M29W128 NOR Flash memory, the timings are:
  - Address setup time: 0x0
  - Data setup time: 0x5
- For the S29GL128P NOR Flash memory, the timings are:
  - Address setup time: 0x2
  - Data setup time: 0x5

Note: 1 The S29GL128P NOR Flash memory timings are also valid for the M29W128 NOR Flash memory.

2.3 Hardware connection

Table 4 gives the correspondence between the NOR Flash memory pins and the FSMC pins and shows the GPIO configuration for each FSMC pin.

In case of an 8-bit NOR Flash memory, the data/address bus is 8-bit wide and D8-D15 should not be connected to the FSMC. The unused FSMC pins can be used for general purpose I/O.

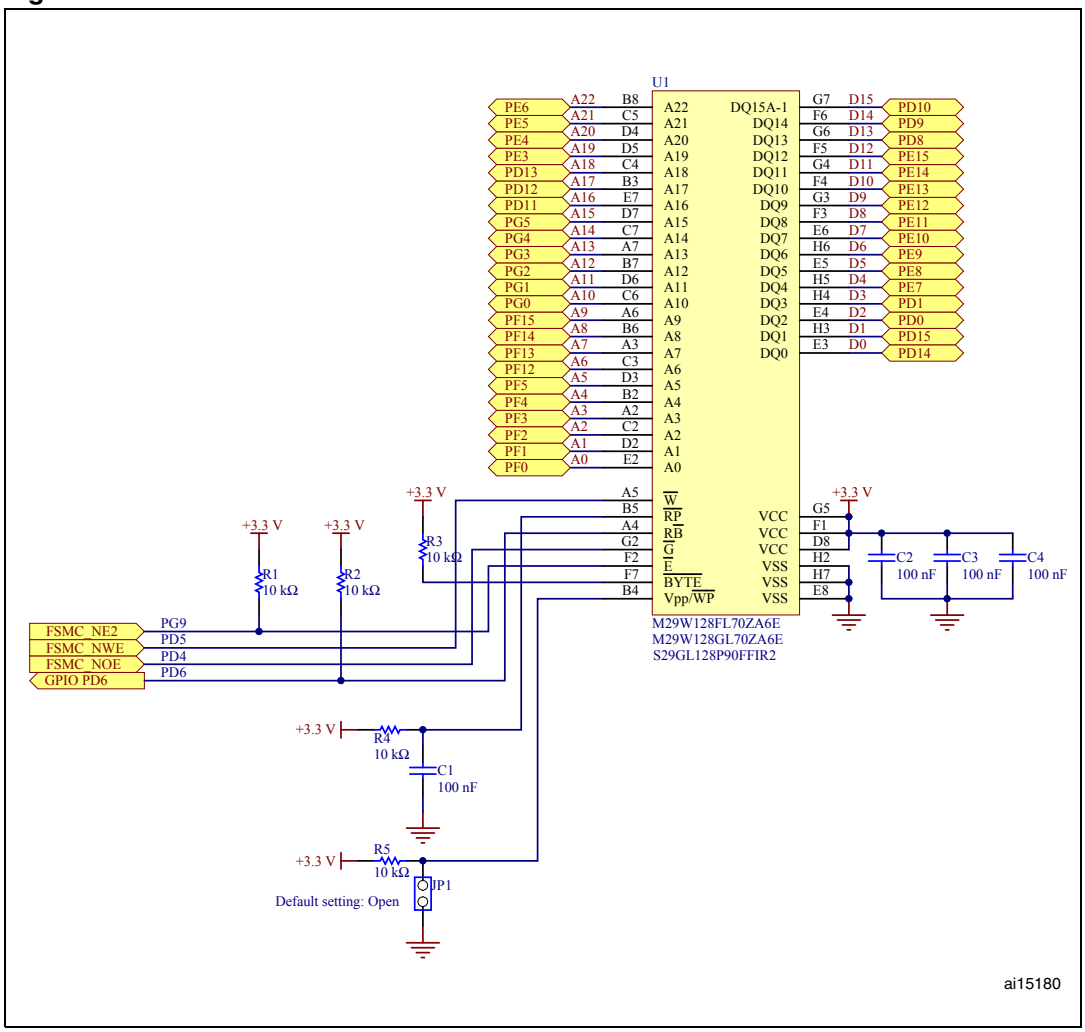
In case of a synchronous memory, the FSMC_CLK pin should be connected to the memory clock pin.

Table 4. M29W128FL signal to FSMC pin correspondence

Memory signals	FSMC signals	Pin / Port assignment	Pin / Port configuration	Signal description
A0-A22	A0-A22	Port F/Port G/Port E/Port D	AF push-pull	Address A0-A22
DQ0-DQ7	D0-D7	Port D/Port E	AF push-pull	Data D0-D7
DQ8-DQ14	D8-D14	Port D/Port E	AF push-pull	Data D8-D14
DQ15A-1	D15	PD10	AF push-pull	Data D15
$\overline{E}$	NE2	PG9	AF push-pull	Chip Enable
$\overline{G}$	NOE	PD4	AF push-pull	Output Enable
$\overline{W}$	NWE	PD5	AF push-pull	Write Enable

Figure 5 shows a typical connection between an STM32F10xxx microcontroller and the M29W128FL NOR Flash memory. It is an abstract from the schematic of the STM3210E-EVAL (STM32F10xxx evaluation board).

Figure 5. 16-bit NOR Flash: M29W128FL/GL connection to STM32F10xxx



Note: The GPIO PD6 pin is used to provide a Ready/Busy output signal for NOR Flash memories (in this case the application needs to poll on the state of this pin to guarantee correct operation).
In the example shown below, this pin is not used with the M29W128FL and M29W128GL NOR Flash memories. It is however required for the S29GL128P NOR Flash memory.

A firmware example is available within the STM32F10xxx firmware library provided in the **NOR** directory at the path:
STM32F10x_StdPeriph_Lib\Project\STM32F10x_StdPeriph_Examples\FSMC.

The main goal of this example is to provide the basics of how to use the FSMC firmware library and the associated NOR Flash memory driver to perform erase/read/write operations on the M29W128FL, M29W128GL or S29GL128P NOR Flash memories mounted on the STM3210E-EVAL board.
For the FSMC timings, the FSMC NOR Flash firmware driver uses the highest timing values, that is those of the S29GL128P NOR Flash memory. In this way, the firmware driver is able to support all the NOR Flash memories available on the STM3210E-EVAL.

2.4 Code execution from an external NOR Flash memory

The High-density STM32F10xxx devices feature up to 512 KB of internal Flash memory, which is enough for most applications. For systems that need greater Flash memory capacity, an attractive alternative is to use external NOR Flash memory.

This section describes how to use a NOR Flash memory to execute user code. For this, two steps are mandatory:

- Load the user code into the external NOR memory:

  This operation requires a special configuration of the development toolchain: in the linker file, you have to specify the NOR Flash memory start address (or any other address) from where the user code is to be programmed.
  A specific loader for NOR Flash memories is also required.

- Execute the user code:

  Once the user code has been loaded into the NOR Flash memory, a specific program should be loaded into the internal Flash memory in order to configure the FSMC, to jump and execute the user code (from the NOR Flash memory).

A typical example of such an application is provided within the STM32F10xxx firmware library at the following path:
STM32F10x_StdPeriph_Lib\Project\STM32F10x_StdPeriph_Examples\FSMC.
Copy the binary of the GPIO IOToggle example into the NOR Flash memory mounted on the STM3210E-EVAL board, then execute it.
For more details on how to use this example with your development toolchain, please refer to the readme files provided within this directory.

3 Interfacing with a nonmultiplexed, asynchronous 16-bit SRAM

3.1 FSMC configuration

SRAMs and NOR Flash memories share the same FSMC banks. The protocol to be used depends on the selected memory type.

To control an SRAM, the FSMC provides the following possible features:

- Enable or disable the address/data multiplexing feature
- Select the memory type to be used: NOR/SRAM/PSRAM
- Define the external memory databus width: 8/16 bits
- Enable or disable the extended mode: this mode is used to access the memory with a different timing configuration for read and write operations.

Like for NOR Flash memories, the user has to compute and set the following parameters as a function of the information in the SRAM datasheet:

ADDSET: address setup time

DATAST: data setup time

Figure 6 and *Figure 7* show the different timings for a typical SRAM access.

Figure 6. SRAM asynchronous read access timing

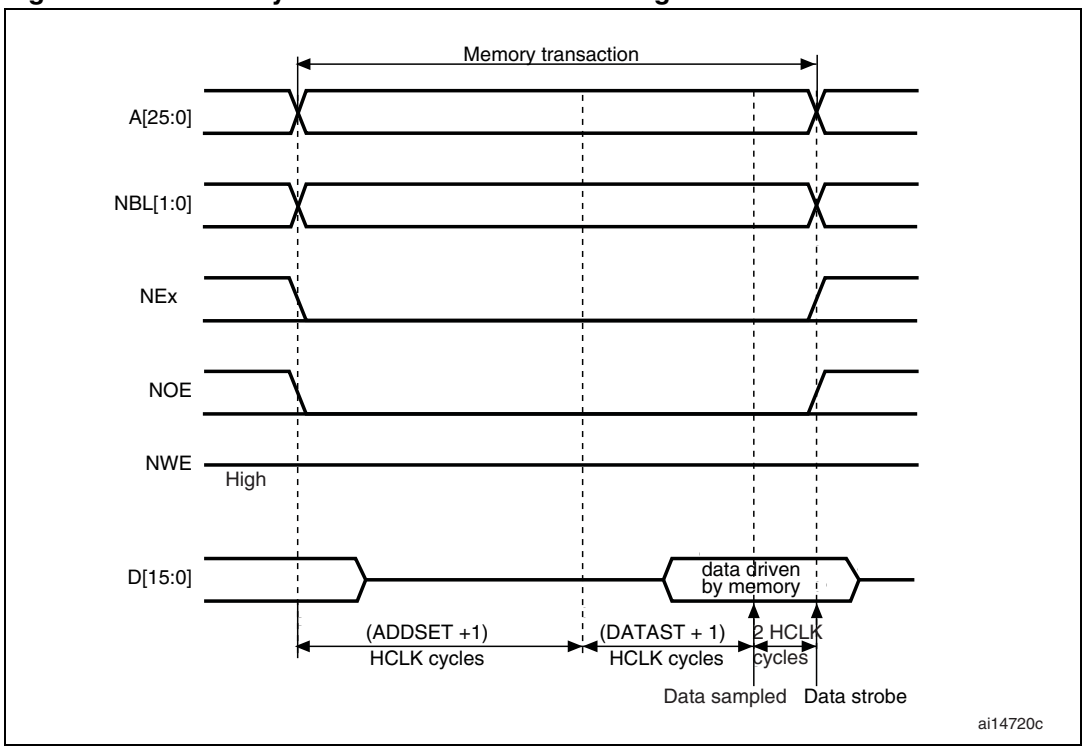
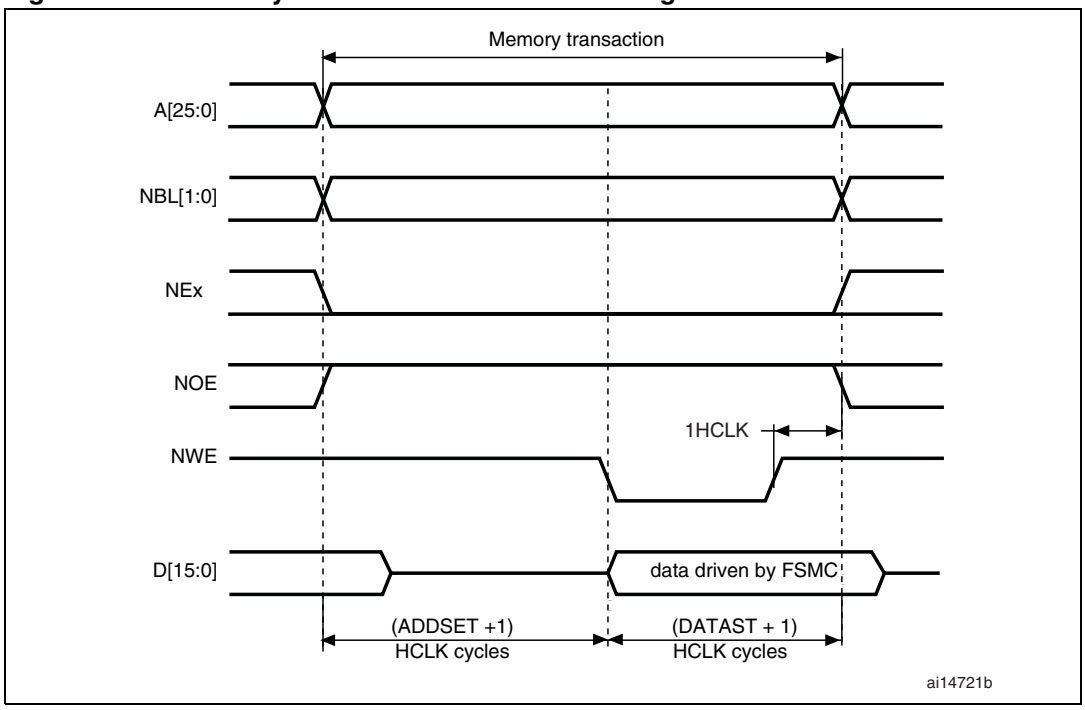


Figure 7. SRAM asynchronous write access timing



3.1.1 Typical use of the FSMC to interface with an SRAM

In this application note, the IS61WV51216BLL memory is used as the reference.

The IS61WV51216BLL memory is a nonmultiplexed, asynchronous, 16-bit memory. Bank3 is selected to support the SRAM device. Based on these data, the FSMC is configured as follows:

- Bank3 is enabled: BCR3_MBKEN bit set to '1'
- Memory type is SRAM: BCR3_MTYP is set to '00' to select the SRAM memory type
- Databus width is 16 bits: BCR3_MWID is set to '01' to select the 16-bit width
- The memory is nonmultiplexed: BCR3_MUXEN is reset

All remaining parameters must be kept cleared.

3.2 Timing computation

The SRAM shares the same banks and configuration register as the NOR Flash memory. As a result, the timing computation method is the same as described in detail in the NOR Flash section (*Section 2.2: Timing computation on page 11*).

The FSMC is configured on the basis of the SRAM access timings illustrated in *Figure 6* and *Figure 7*, and taking into account the following:

- the maximum read/write access time
- the different internal FSMC delays
- the different internal memory delays

Hence the following equations:

$((ADDSET + 1) + (DATAST + 1)) \times t_{HCLK} \geq \max (t_{WC})$

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

For read access, DATAST must verify:

$DATAST \geq (t_{AA} + t_{su(Data\_NE)} + t_{v(A\_NE)})/t_{HCLK} - ADDSET - 4$

Table 5 gives the meanings and values of the SRAM parameters.

Table 5. IS61WV51216BL SRAM timings

Symbols	Parameter	Value	Unit
t_{WC}	Write cycle time	12	ns
t_{RC}	Read cycle time	12	ns
t_{PWE1}	Write Enable low pulse width	8	ns
t_{AA}	Address access time	12	ns

Using the above described formulas, the memory timings in *Table 5* and the STM32F10xCDE FSMC asynchronous timings in *Table 2*, we have:

- Address setup time: 0x0
- Data setup time: 0x1

3.3 Hardware connection

Table 6 gives the correspondence between SRAM pins and FSMC pins and shows the GPIO configuration for each FSMC pin.

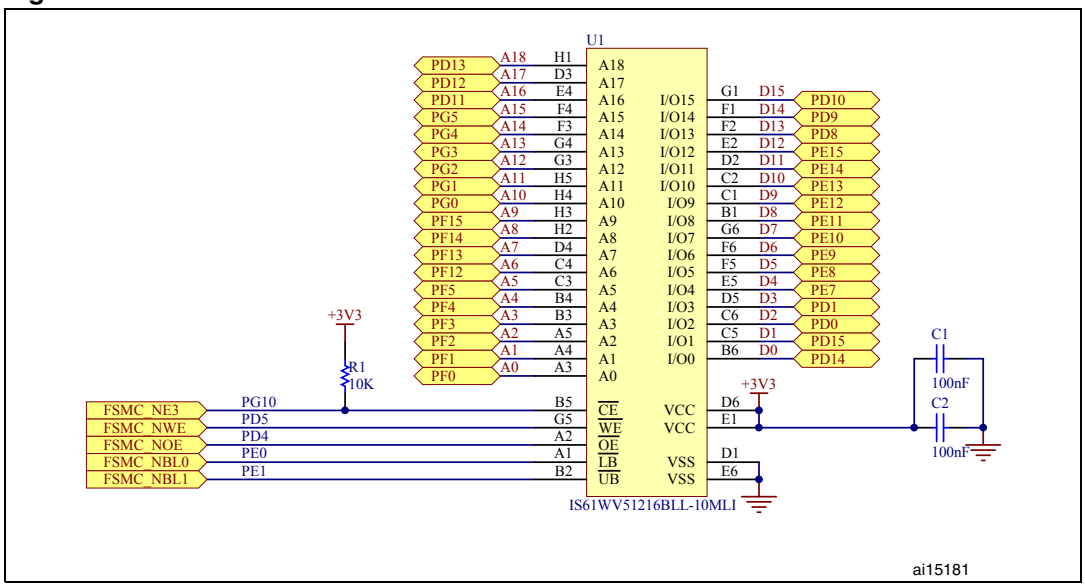
In case of an 8-bit SRAM, the data/address bus is 8 bits wide and D8-D15 should not be connected to the FSMC. The unused FSMC pins can be used as general purpose I/O pins

Table 6. IS61WV51216BLL signal to FSMC pin correspondence

Memory signals	FSMC signals	Pin / Port assignment	Pin / Port configuration	Signal description
A0-A18	A0-A18	Port F/Port G/Port E/Port D	AF push-pull	Address A0-A18
I/O0-I/O15	D0-D15	Port D/Port E	AF push-pull	Data D0-D15
$\overline{CE}$	NE3	PG10	AF push-pull	Chip Enable
$\overline{OE}$	NOE	PD4	AF push-pull	Output Enable
$\overline{WE}$	NWE	PD5	AF push-pull	Write Enable
$\overline{LB}$	NBL0	PE0	AF push-pull	Lower byte control
$\overline{UB}$	NBL1	PE1	AF push-pull	Upper byte control

Figure 8 shows a typical connection between an STM32F10xxx microcontroller and an IS61WV51216BLL SRAM. It is an abstract from the schematic of the STM32F10xxx evaluation board: STM3210E-EVAL.

Figure 8. 16-bit SRAM: IS61WV51216BLL connection to STM32F10xxx



A firmware example is available within the STM32F10xxx firmware library provided in the **FSMC** directory at the path:
STM32F10x_StdPeriph_Lib\Project\STM32F10x_StdPeriph_Examples\FSMC .

The main goal of this example is to provide the basics of how to use the FSMC firmware library and the associate SRAM driver to perform read/write operations on the IS61WV51216BLL SRAM memory mounted on the STM3210E-EVAL board.

3.4 Using the external SRAM as a data memory

The external SRAM mapped on the FSMC can be used as a data memory in several applications where a large R/W data memory is required.

The configuration steps required to use the external SRAM as a data memory depend on the toolchain and hardware being used.

A typical example of such an application is provided within the STM32F10xxx firmware library at the following path:
STM32F10x_StdPeriph_Lib\Project\STM32F10x_StdPeriph_Examples\FSMC.
This example shows how to use the external SRAM mounted on the STM3210E-EVAL board as a program data memory and, the internal SRAM as stack. For more details on how to use this example with your development toolchain, please refer to the readme file provided within this directory.

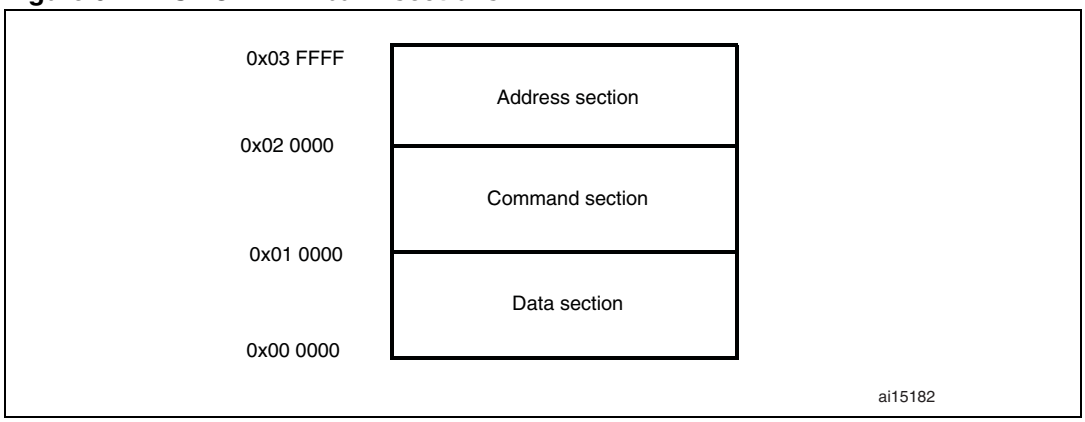
4 Interfacing with an 8-bit NAND Flash memory

NAND Flash memories are accessed in accordance with a specific protocol. To write to or read from the NAND Flash memory, it is necessary to:

1. Send a command to the NAND Flash memory
2. Send the address to write to or read from
3. Read or write the data

The FSMC NAND banks are divides into three sections to allow the user to easily program the NAND Flash memory: data section, address section, and command section.

Figure 9. FSMC NAND bank sections



In fact, the three sections are the representation of the real NAND Flash memory. By writing to any location in the command section, the user writes the command to the NAND Flash memory. By writing to any location in the address section, the user specifies the address of the read or write operation. Depending on the structure of the used NAND Flash memory, four or five operations are required to write the address. By writing to or reading from any location in the data section, the data are written to or read from the address previously sent to the address section.

4.1 FSMC configuration

To control a NAND Flash memory, the FSMC provides the following possible features:

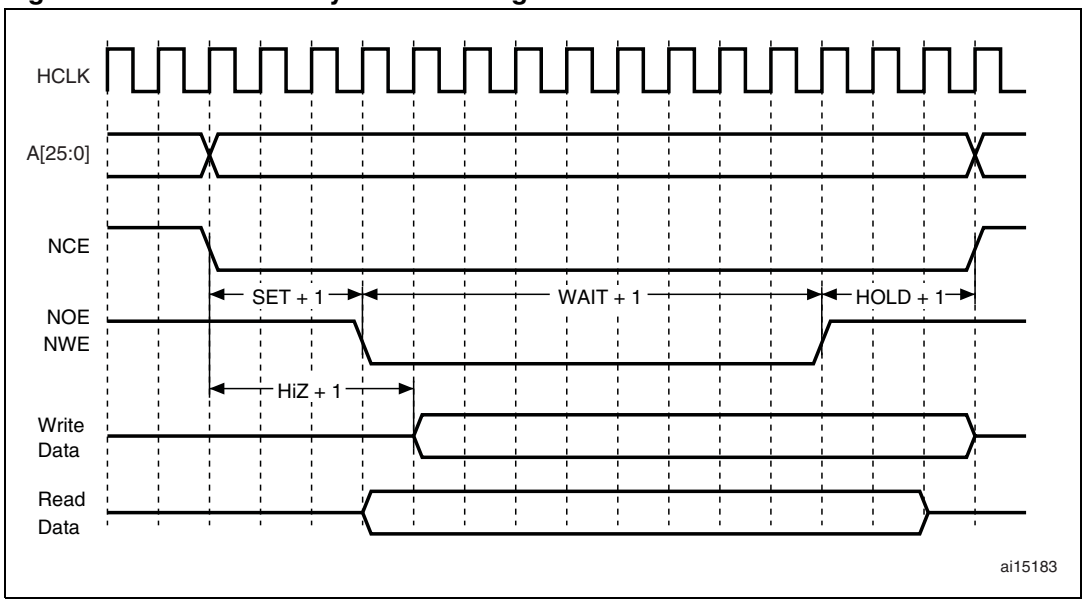
- Enable or disable the use of the memory Ready/$\overline{\text{Busy}}$ signal as the wait input for the FSMC.
- Enable or disable the use of the memory Ready/$\overline{\text{Busy}}$ signal as the interrupt source for the FSMC: the interrupt can be generated with three possible configurations:
  - on the rising edge of the Ready/$\overline{\text{Busy}}$ signal: when the memory has just completed an operation and the new status in ready.
  - on the falling edge of the Ready/$\overline{\text{Busy}}$ signal: when the memory starts the new operation
  - on the high level of the Ready/$\overline{\text{Busy}}$ signal: when the memory is ready.
- Select the NAND Flash databus width: 8- or 16-bit width.
- Enable or disable the ECC computation logic.
- Specify the ECC page size: it can be 256, 512, 1024, 2048, 4096 or 8192 bytes.

The FSMC also provides the possibility of programming the timings for the different NAND Flash sections separately: common section and attribute section. The timings are the following:

- **Setup time**: it is the time (in HCLK) required to set up the address before the command assertion. That is, It is the time from address valid to the start of the read or write operation.
- **Wait time**: It is the time (in HCLK) required to assert the command. That is, it is the time taken by the NOE and NWE signals to become deasserted.
- **Hold time**: it is the time (in HCLK) during which the address is held after the command deassertion. That is, it is the time between the deassertion of the NOE and NWE signals and the end of the operation cycle.
- **Databus HiZ time**: it is only valid for write operations and corresponds to the time (in HCLK) during which the databus is kept in the high-impedance state after the start of a write access. That is, it is the time from address valid to databus driven.

Figure 10 shows the different timings for a typical NAND Flash memory access.

Figure 10. NAND memory access timing



4.1.1 Typical use of the FSMC to interface with a NAND memory

The STM32F10xxx FSMC NAND Flash controller can configure Bank2 or Bank3 to support NAND Flash memories.

The banks are selected using the Chip Select signals, as each bank is associated with a specific Chip Select.

To enable communication with the NAND Flash devices, the FSMC NAND Flash controller has to be initialized to meet the characteristics of the NAND Flash devices: features, timings, data width, etc.

In this application note, the Numonyx NAND512W3A is used as the reference. This memory has the same access protocol as many other NAND Flash memories on today's market.

NAND512W3A characteristics:

- NAND interface: x8 bus width, multiplexed address/ data
- Page size: x8 device: (512 + 16 spare) bytes
- Page Read/Program timings:
  - Random access: 12 µs (3 V)/15 µs (1.8 V) (max)
  - Sequential access: 30 ns (3 V)/50 ns (1.8 V) (min)
  - Page Program time: 200 µs (typ)

Bank2 is selected to support the NAND Flash device. Based on these data, the FSMC is configured as follows:

- Bank2 is enabled: PCR2_PBKEN bit set to '1'
- Memory type is NAND Flash: PCR2_PTYP is set to '1' to select the NAND Flash memory type.
- Databus width is 8 bit: PCR2_PWID is set to '00' to select 8-bit width.
- ECC page size is 512 bytes: PCR2_ECCPS is set to '001' to set the ECC computation page size to 512 bytes.
- ECC hardware calculation on/off as needed: PCR2_ECCEN is set or reset accordingly.
- Wait feature may or not be enabled depending on the user's application: PCR2_PWAITEN is set or reset as needed.

The Ready/$\overline{\text{Busy}}$ memory signal can be connected to the FSMC_NWAIT pin, and in this case the Wait feature must be used to manage the NAND Flash operations.

When using the NAND Flash memory with the wait feature enabled, the controller waits for the NAND Flash to be ready to become active before starting a new access. While waiting, the controller maintains the NCE signal active (low).

Generally, the Ready/$\overline{\text{Busy}}$ signal is an open-drain output. To connect this signal to the STM32F10xxx microcontroller, the corresponding pin must be configured as input pull-up.

The Ready/$\overline{\text{Busy}}$ signal can be used as an interrupt source for the FSMC and, in this case, the CPU can perform other tasks during NAND Flash operations.
Three FSMC configurations make it possible to use this signal as an interrupt. For that purpose, the IREN, IFEN or ILEN bits in the SR2 register are used to select the rising edge, the falling edge or the high level of the NAND Flash Ready/$\overline{\text{Busy}}$ signal.

4.2 Timing computation

Besides configuring the different features that are to be used with the NAND Flash memory, the user has to initialize the controller to meet the memory timings.

As described in *Section 4.1*, the FSMC is able to program four different timings for the common space and the attribute space independently: **Setup time, Wait time, Hold time and Databus HiZ time**.

These parameters are computed according to the NAND Flash memory characteristics and the STM32F10xxx HCLK clock.

Based on the NAND Flash memory access timings shown in *Figure 4*, the following equations are found:

- The write or read access time is the time between the falling edge and the rising edge of the NAND Flash memory Chip Select signal. It is computed as a function of the FSMC timing parameter:

  $((SET + 1) + (WAIT + 1) + (HOLD + 1)) \times t_{HCLK} \geq$ Write/Read access
- The Wait time is measured between the falling edge and the rising edge of the Write/Read Enable signal:

  Write/Read Enable signal low to high = $(WAIT + 1) \times t_{HCLK}$
- For write access, the HIZ parameter is the time measured between the falling edge of the Chip Select signal and the data setup on bus:

  $(HIZ+1) \times t_{HCLK} \geq$ Chip Select setup time to Data setup.
- The HOLD timing is given in the NAND datasheet as follows:

  $(HOLD +1) \times t_{HCLK}$ = Write Enable High to Chip Enable / Address/Command Latch High

To make sure of the correct timing configuration of the FSMC, the timings have to take into consideration:

- The maximum read/write access time
- The different internal FSMC delays
- The different internal memory delays

Hence, we have the following equations following the NAND512W3A NAND datasheet:

- $(SET + 1) \times t_{HCLK} \geq \max (t_{CS}, t_{CLS}, t_{ALS}, t_{CLR}, t_{AR}) - t_{WP}$
- $(WAIT + 1) \times t_{HCLK} \geq \max (tWP,tRP)$
- $(HIZ + 1) \times t_{HCLK} \geq \max(t_{CS}, t_{ALS}, t_{CLS}) + (t_{WP} - t_{DS})$
- $(HOLD + 1) \times t_{HCLK} \geq \max (t_{CH}, t_{CLH}, t_{ALH})$
- $((WAIT + 1) + (HOLD + 1) + (SET + 1)) \times t_{HCLK} \geq \max (t_{WC/RC})$

Considering the different timings in the FSMC and the memory, the equations become:

- WAIT must verify:

$(WAIT + 1) \times t_{HCLK} \geq (t_{REA} + t_{su(D\text{-}NOE)})$

$WAIT \geq (t_{REA} + t_{su(D\text{-}NOE)}) / t_{HCLK} - 1$

Note: $t_{su(D\text{-}NOE)}$ is specified in the STM32F10xCDE datasheet.
$t_{su(D\text{-}NOE)}$ = 25 ns

Table 7 gives the meanings and values of the NAND512W3A2C memory parameters.

Table 7. NAND512W3A2C Flash memory timings

Symbol	Parameter	Value	Unit
t_{CEA}	Chip Enable low to output valid	35	ns
t_{WP}	Write Enable low to Write Enable high	15	ns

Table 7. NAND512W3A2C Flash memory timings (continued)

Symbol	Parameter	Value	Unit
t_{RP}	Read Enable low to Read Enable high	15	ns
t_{CS}	Chip Enable low to Write Enable high	20	ns
t_{ALS}	AL setup time	15	ns
t_{CLS}	CL Setup time	15	ns
t_{CH}	$\overline{E}$ Hold time	5	ns
t_{ALH}	AL Hold time	5	ns
t_{CLH}	CL Hold time	5	ns
t_{DS}	Data Valid to Write Enable high	15	ns
t_{WC}	Write Enable low to Write Enable low	30	ns
t_{RC}	Read Enable low to Read Enable low	30	ns
t_{REA}	Read Enable low to Output Valid	18	ns

Using the above described formulas, the memory timings in *Table 7* and the STM32F10xCDE FSMC asynchronous timings in *Table 2*, we have:

- Setup time: 0x0
- Wait time: 0x2
- Hold time: 0x1
- HiZ time: 0x0

4.3 Hardware connection

Table 8 gives the correspondence between the NAND Flash memory pins and the FSMC pins and shows the GPIO configuration for each FSMC pin.

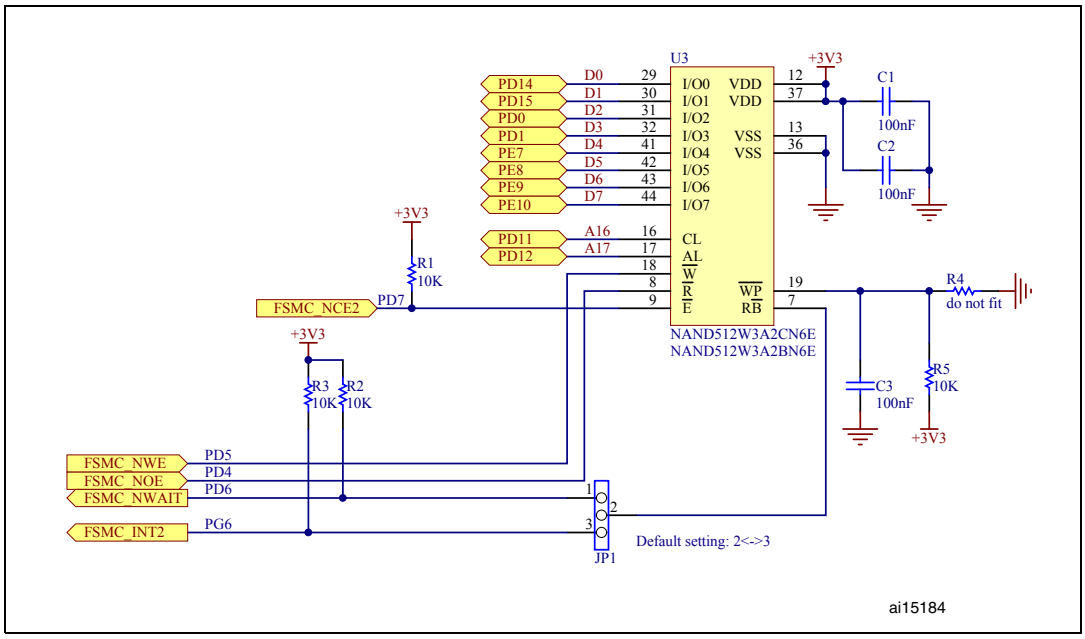
In case of a 16-bit NAND Flash memory, the data/address bus is 16 bits wide and the remaining FSMC data/address bus are used.

Table 8. NAND512W3A signal to FSMC pin correspondence

Memory signals	FSMC signals	Pin / Port assignment	Pin / Port configuration	Signal description
AL	ALE/A17	PD11	AF push-pull	Address Latch Enable
CL	CLE/A16	PD12	AF push-pull	Command Latch Enable
I/O0-I/O7	D0-D7	Port D/Port E	AF push-pull	Data D0-D7
$\overline{E}$	NCE2	PD7	AF push-pull	Chip Enable
$\overline{R}$	NOE	PD4	AF push-pull	Output Enable
$\overline{W}$	NWE	PD5	AF push-pull	Write Enable
$R\overline{B}$	NWAIT/INT2	PD6/PG6	Input pull-up	Ready/$\overline{Busy}$ signal

Figure 11 illustrates a typical connection between the STM32F10xxx microcontroller and the NAND512W3A memory. It is an abstract from the schematic of the STM32F10xxx evaluation board: STM3210E-EVAL.

Figure 11. 8-bit NAND Flash: NAND512W3A2C/NAND512W3A2B connection to STM32F10xxx



A firmware example is available within the STM32F10xxx firmware library provided in the **NAND** directory at the path:
STM32F10x_StdPeriph_Lib\Project\STM32F10x_StdPeriph_Examples\FSMC.

The main goal of this example is to provide the basics of how to use the FSMC firmware library and the associated NAND Flash memory driver to perform erase/read/write operations using the FSMC wait feature on the NAND512W3A2 memory mounted on the STM3210E-EVAL board.

4.4 Error correction code computation

4.4.1 Error correction code (ECC) computation overview

The FSMC NAND Flash controller includes two pieces of error correction code computation hardware, one for each NAND Flash memory block.

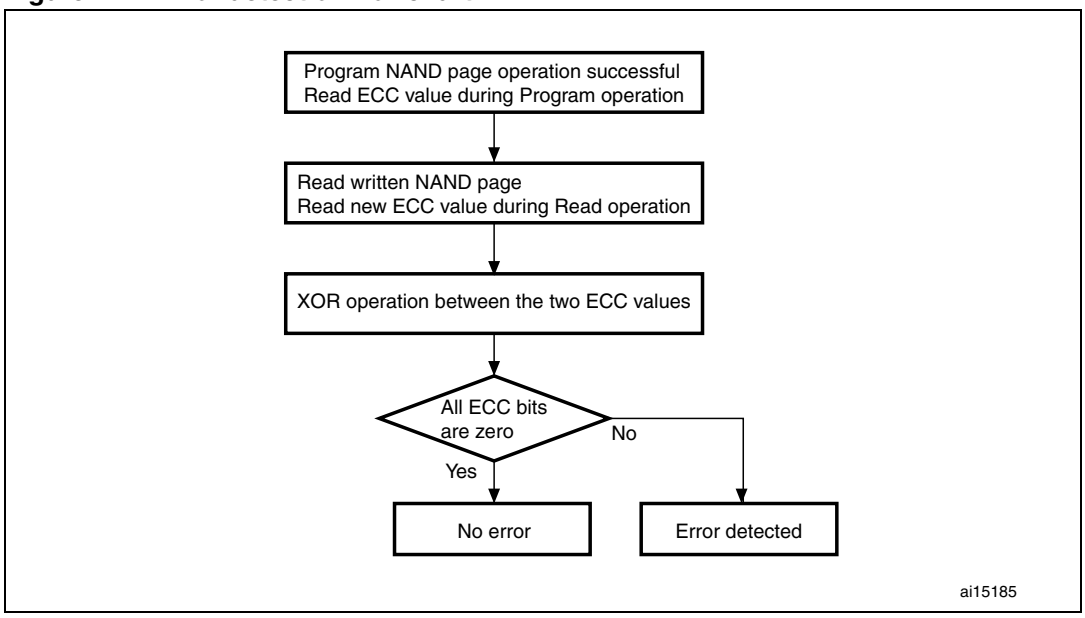
The ECC can be performed for page sizes of 256, 512, 1024, 2048, 4096 or 8192 bytes, depending on the ECC page size configured by the user. Depending on the configured page size, the ECC code will be 22, 24, 26, 28, 30 or 32 bits.

To even improve the error coverage, the user can read/write the NAND Flash page with a reduced ECC page size. This is possible when starting and stopping the ECC computation after the desired number of bytes to check. In this case, the ECC code is only calculated for the bytes written and read.

The error correction code algorithm implemented in the FSMC can perform 1-bit and 2-bit error detection per page read from or written to the NAND Flash memory. It is based on the Hamming algorithm and consists in calculating the row and column parity.

4.4.2 Error detection

Figure 12. Error detection flowchart



When an error occurs during the write operation, this error is either correctable or uncorrectable depending on the ECC XOR operation:

- Case of a correctable error

  The ECC XOR operation contains 11-bit data at 1. And each pair parity is 0x10 or 0x01.
- Case of an ECC error

  The ECC XOR operation contains only one bit at 1.
- Case of an uncorrectable error

  The ECC XOR operation is random data. In this case the page data cannot be corrected.

Based on the flowchart shown in *Figure 12*, the correction software is easy to implement.

The first step consists in detecting whether an error occurred during the write operation. If that was the case, the second step consists in determining if the error is correctable or not. If it is correctable, then the third step consists in correcting the error.

The error correction is based on the second ECC generated after the read operation. The error location can be identified from this code. Usually, the following data are extracted from the ECC:

P1024, P512, P256, P128, P64, P32, P16, P8, P4, P2, P1, where Px are the line and column parity.

In case of an 8-bit memory, P4, P2, P1 define the error bit position. And P1024, P512, P256, P128, P64, P32, P16, P8 define the error byte position.

5 STM32F10xxx FSMC configuration in 100-pin packages

The FSMC is present in devices delivered in both 144-pin and 100-pin packages. For devices in 100-pin packages, however, only some FSMC banks can be used because not all pins are available.

5.1 Interfacing the FSMC with a NAND Flash memory

In devices that come in 100-pin packages, only Bank2 can be used to interface the FSMC with an 8-/16-bit NAND Flash memory. This is because the NCE3 pin is not available in these packages.

Likewise, no interrupt can be used because the two interrupt pins INT2 and INT3 are not available in 100-pin packages.

Table 9 shows how to connect an 8-/16-bit NAND Flash memory to the FSMC peripheral of devices delivered in 100-pin packages.

Table 9. NAND Flash memory connection to the FSMC

8-/16-bit NAND memory pins	FSMC pins	100-pin package
$\overline{E}$	NCE2/NCE3	NCE2
$\overline{R}$	NOE	NOE
$\overline{W}$	NWE	NWE
AL	A17	A17
CL	A16	A16
R/$\overline{B}$	NWAIT/INT2/INT3	NWAIT
I/O0-I/O7	D0-D7	D0-D7
I/O8-I/O15	D8-D15	D8-D15

5.2 Interfacing the FSMC with a NOR Flash memory

In devices that come in 100-pin packages, only Bank 1 (NOR/PSRAM 1) can be used to interface with a NOR Flash memory. This is because the NE2, NE3 and NE4 pins are not available in these packages.

Likewise, the A0-A15 pins are not available, so the NOR Flash controller should be used in multiplexed mode to use the databus for both address and data.

In 8-bit muxed mode, the FSMC address signals A[15:8] are avaialble on the AD[15:8] pins.

Table 10 shows how to connect a NOR Flash memory to the FSMC peripheral of devices delivered in 100-pin packages.

Table 10. NOR Flash memory connection to the FSMC

8-/16-bit NOR/PSRAM pins	FSMC pins	100-pin package
A[15:0]	A[15:0]	DA[15:0]
A[23:16]	A[23:16]	A[23:16]
DQ[15:0]	D[15:0]	D[15:0]
$\overline{W}$	NWE	NWE
$\overline{E}$	NE1/NE2/NE3/NE4	NE1
$\overline{G}$	NOE	NOE
ADV	NADV	NADV
R/B	NWAIT	NWAIT
UB, LB	NBL[1:0]	NBL[1:0]

6 Revision history

Table 11. Document revision history

Date	Revision	Changes
22-Jul-2008	1	Initial release.
19-Sep-2008	2	Cellular RAM, OneNAND and COSMORAM references removed. NWAIT signal removed from *Figure 3: Asynchronous NOR Flash read access timing*, *Figure 4: Asynchronous NOR Flash write access timing*, *Figure 5: 16-bit NOR Flash: M29W128FL/GL connection to STM32F10xxx*, *Figure 6: SRAM asynchronous read access timing* and *Figure 7: SRAM asynchronous write access timing*. Note modified in *Section 2.3: Hardware connection*. *Table 10: NOR Flash memory connection to the FSMC* modified. Note added to *Section 2.2: Timing computation*. Small text changes.
07-Apr-2009	3	*Figure 1: FSMC block diagram* corrected. $t_{su(Data\_NE)}$ and $t_{v(A\_NE)}$ rows removed and $t_{su(Data\_NE)} + t_{v(A\_NE)}$ value modified in *Table 2: STM32F10xCDE FSMC asynchronous timings*. *Note: 1 on page 12* modified. Data setup time modified in *Section 2.2: Timing computation* and *Section 3.2: Timing computation*. Small text change in *Section 4.1.1: Typical use of the FSMC to interface with a NAND memory*. Small text changes.
07-Jun-2010	4	Updated *Section 1: Overview of the STM32F10xxx flexible static memory controller* Added *Section 1.1: Interfacing asynchronous static memories (NOR Flash, SRAM)* Modified *Section 2.1: FSMC configuration* Updated *Section 2.2: Timing computation* Updated *Section 2.3: Hardware connection* Updated *Section 3.2: Timing computation* Updated *Section 4.2: Timing computation* Updated *Section 5.2: Interfacing the FSMC with a NOR Flash memory*

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com