Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	ARM® Cortex®-M4
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	79
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4s2ca-au

4.1.6 100-ball VFBGA Pinout

Table 4-3. SAM4SD32/SD16/SA16/S16/S8/S4/S2 100-ball VFBGA Pinout

A1	ADVREF	C6	PC9	F1	VDDOUT	H6	PA12/PGMD0
A2	VDDPLL	C7	TMS/SWDIO/PB6	F2	PA18/PGMD6/AD1	H7	PA9/PGMM1
A3	PB9/PGMCK/XIN	C8	PA1/PGMEN1	F3	PA17/PGMD5/AD0	H8	VDDCORE
A4	PB8/XOUT	C9	PA0/PGMEN0	F4	GND	H9	PA6/PGMNOE
A5	JTAGSEL	C10	PC16	F5	GND	H10	PA5/PGMRDY
A6	DDP/PB11	D1	PB1/AD5	F6	PC26	J1	PA20/AD3/PGMD8
A7	DDM/PB10	D2	PC30/AD14	F7	PA4/PGMNCMD	J2	PC12/AD12
A8	PC20	D3	PC31	F8	PA28	J3	PA16/PGMD4
A9	PC19	D4	PC22	F9	TST	J4	PC6
A10	TDO/TRACESWO/PB5	D5	PC5	F10	PC8	J5	PA24/PGMD12
B1	GNDANA	D6	PA29	G1	PC15/AD11	J6	PA25/PGMD13
B2	PC25	D7	PA30	G2	PA19/PGMD7/AD2	J7	PA11/PGMM3
B3	PB14/DAC1	D8	GND	G3	PA21/AD8/PGMD9	J8	VDDCORE
B4	PB13/DAC0	D9	PC14	G4	PA15/PGMD3	J9	VDDCORE
B5	PC23	D10	PC11	G5	PC3	J10	TDI/PB4
B6	PC21	E1	VDDIN	G6	PA10/PGMM2	K1	PA23/PGMD11
B7	TCK/SWCLK/PB7	E2	PB3/AD7	G7	PC1	K2	PC0
B8	PA31	E3	PB2/AD6	G8	PC28	K3	PC7
B9	PC18	E4	GND	G9	NRST	K4	PA13/PGMD1
B10	PC17	E5	GND	G10	PA27/PGMD15	K5	PA26/PGMD14
C1	PB0/AD4	E6	GND	H1	PC13/AD10	K6	PC2
C2	PC29/AD13	E7	VDDIO	H2	PA22/AD9/PGMD10	K7	VDDIO
C3	PC24	E8	PC10	H3	PC27	K8	VDDIO
C4	ERASE/PB12	E9	PA2/PGMEN2	H4	PA14/PGMD2	K9	PA8/XOUT32/PGMM0
C5	VDDCORE	E10	PA3	H5	PC4	K10	PA7/XIN32/ PGMNVALID

9. Real Time Event Management

The events generated by peripherals are designed to be directly routed to peripherals managing/using these events without processor intervention. Peripherals receiving events contain logic by which to select the one required.

9.1 Embedded Characteristics

- Timers, PWM, IO peripherals generate event triggers which are directly routed to event managers such as ADC or DACC, for example, to start measurement/conversion without processor intervention.
- UART, USART, SPI, TWI, SSC, PWM, HSMCI, ADC, DACC, PIO also generate event triggers directly connected to Peripheral DMA Controller (PDC) for data transfer without processor intervention.
- Parallel capture logic is directly embedded in PIO and generates trigger event to PDC to capture data without processor intervention.
- PWM security events (faults) are in combinational form and directly routed from event generators (ADC, ACC, PMC, TIMER) to PWM module.
- PMC security event (clock failure detection) can be programmed to switch the MCK on reliable main RC internal clock without processor intervention.

- If the instruction is conditional, it must be the last instruction in the IT block

- With the exception of the ADD{*cond*} PC, PC, Rm instruction, *Rn* can be PC only in ADD and SUB, and only with the additional restrictions:
  - The user must not specify the S suffix
  - The second operand must be a constant in the range 0 to 4095.
  - Note: When using the PC for an addition or a subtraction, bits[1:0] of the PC are rounded to 0b00 before performing the calculation, making the base address for the calculation word-aligned.
  - Note: To generate the address of an instruction, the constant based on the value of the PC must be adjusted. ARM recommends to use the ADR instruction instead of ADD or SUB with *Rn* equal to the PC, because the assembler automatically calculates the correct constant for the ADR instruction.

When *Rd* is PC in the ADD{*cond*} PC, PC, Rm instruction:

- Bit[0] of the value written to the PC is ignored
- A branch occurs to the address created by forcing bit[0] of that value to 0.

Condition Flags

If `S` is specified, these instructions update the N, Z, C and V flags according to the result.

Examples

```
ADD     R2, R1, R3          ; Sets the flags on the result
SUBS    R8, R6, #240        ; Subtracts contents of R4 from 1280
RSB     R4, R4, #1280       ; Only executed if C flag set and Z
ADCHI   R11, R0, R3         ; flag clear.
```

Multiword Arithmetic Examples

The example below shows two instructions that add a 64-bit integer contained in R2 and R3 to another 64-bit integer contained in R0 and R1, and place the result in R4 and R5.

64-bit Addition Example

```
ADDS    R4, R0, R2    ; add the least significant words
ADC     R5, R1, R3    ; add the most significant words with carry
```

Multiword values do not have to use consecutive registers. The example below shows instructions that subtract a 96-bit integer contained in R9, R1, and R11 from another contained in R6, R2, and R8. The example stores the result in R6, R9, and R2.

96-bit Subtraction Example

```
SUBS    R6, R6, R9    ; subtract the least significant words
SBCS    R9, R2, R1    ; subtract the middle words with carry
SBC     R2, R8, R11   ; subtract the most significant words with carry
```

12.6.5.2 AND, ORR, EOR, BIC, and ORN

Logical AND, OR, Exclusive OR, Bit Clear, and OR NOT.

Syntax

op{S}{*cond*} {*Rd*,} *Rn*, *Operand2*

where:

op is one of:

AND logical AND.

ORR logical OR, or bit set.

EOR logical Exclusive OR.

BIC logical AND NOT, or bit clear.

ORN logical OR NOT.

12.6.5.14 SASX and SSAX

Signed Add and Subtract with Exchange and Signed Subtract and Add with Exchange.

Syntax

```
op{cond} {Rd}, Rm, Rn
```

where:

op is any of:

SASX Signed Add and Subtract with Exchange.

SSAX Signed Subtract and Add with Exchange.

cond is an optional condition code, see “Conditional Execution” .

Rd is the destination register.

Rn, Rm are registers holding the first and second operands.

Operation

The SASX instruction:

1. Adds the signed top halfword of the first operand with the signed bottom halfword of the second operand.
2. Writes the signed result of the addition to the top halfword of the destination register.
3. Subtracts the signed bottom halfword of the second operand from the top signed highword of the first operand.
4. Writes the signed result of the subtraction to the bottom halfword of the destination register.

The SSAX instruction:

1. Subtracts the signed bottom halfword of the second operand from the top signed highword of the first operand.
2. Writes the signed result of the addition to the bottom halfword of the destination register.
3. Adds the signed top halfword of the first operand with the signed bottom halfword of the second operand.
4. Writes the signed result of the subtraction to the top halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

Examples

```
SASX  R0, R4, R5 ; Adds top halfword of R4 to bottom halfword of R5 and
                 ; writes to top halfword of R0
                 ; Subtracts bottom halfword of R5 from top halfword of R4
                 ; and writes to bottom halfword of R0
SSAX  R7, R3, R2 ; Subtracts top halfword of R2 from bottom halfword of R3
                 ; and writes to bottom halfword of R7
                 ; Adds top halfword of R3 with bottom halfword of R2 and
                 ; writes to top halfword of R7.
```

12.6.5.15 TST and TEQ

Test bits and Test Equivalence.

Syntax

```
TST{cond} Rn, Operand2
TEQ{cond} Rn, Operand2
```

12.9.1.3 Interrupt Control and State Register

Name: SCB_ICSR

Access: Read/Write

31	30	29	28	27	26	25	24
NMIPENDSET	–		PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	–

23	22	21	20	19	18	17	16
–	ISRPENDING	VECTPENDING					

15	14	13	12	11	10	9	8
VECTPENDING				RETTOBASE	–	–	VECTACTIVE

7	6	5	4	3	2	1	0
VECTACTIVE							

The SCB_ICSR provides a set-pending bit for the Non-Maskable Interrupt (NMI) exception, and set-pending and clear-pending bits for the PendSV and SysTick exceptions.

It indicates:

- The exception number of the exception being processed, and whether there are preempted active exceptions,
- The exception number of the highest priority pending exception, and whether any interrupts are pending.

- **NMIPENDSET: NMI Set-pending**

Write:

PendSV set-pending bit.

Write:

0: No effect.

1: Changes NMI exception state to pending.

Read:

0: NMI exception is not pending.

1: NMI exception is pending.

As NMI is the highest-priority exception, the processor normally enters the NMI exception handler as soon as it registers a write of 1 to this bit. Entering the handler clears this bit to 0. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.

- **PENDSVSET: PendSV Set-pending**

Write:

0: No effect.

1: Changes PendSV exception state to pending.

Read:

0: PendSV exception is not pending.

1: PendSV exception is pending.

Writing a 1 to this bit is the only way to set the PendSV exception state to pending.

12.12 Glossary

This glossary describes some of the terms used in technical documents from ARM.

Abort

A mechanism that indicates to a processor that the value associated with a memory access is invalid. An abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction or data memory.

Aligned

A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

Banked register

A register that has multiple physical copies, where the state of the processor determines which copy is used. The Stack Pointer, SP (R13) is a banked register.

Base register

In instruction descriptions, a register specified by a load or store instruction that is used to hold the base value for the instruction's address calculation. Depending on the instruction and its addressing mode, an offset can be added to or subtracted from the base register value to form the address that is sent to memory.

See also "Index register" .

Big-endian (BE)

Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also "Byte-invariant" , "Endianness" , "Little-endian (LE)" .

Big-endian memory

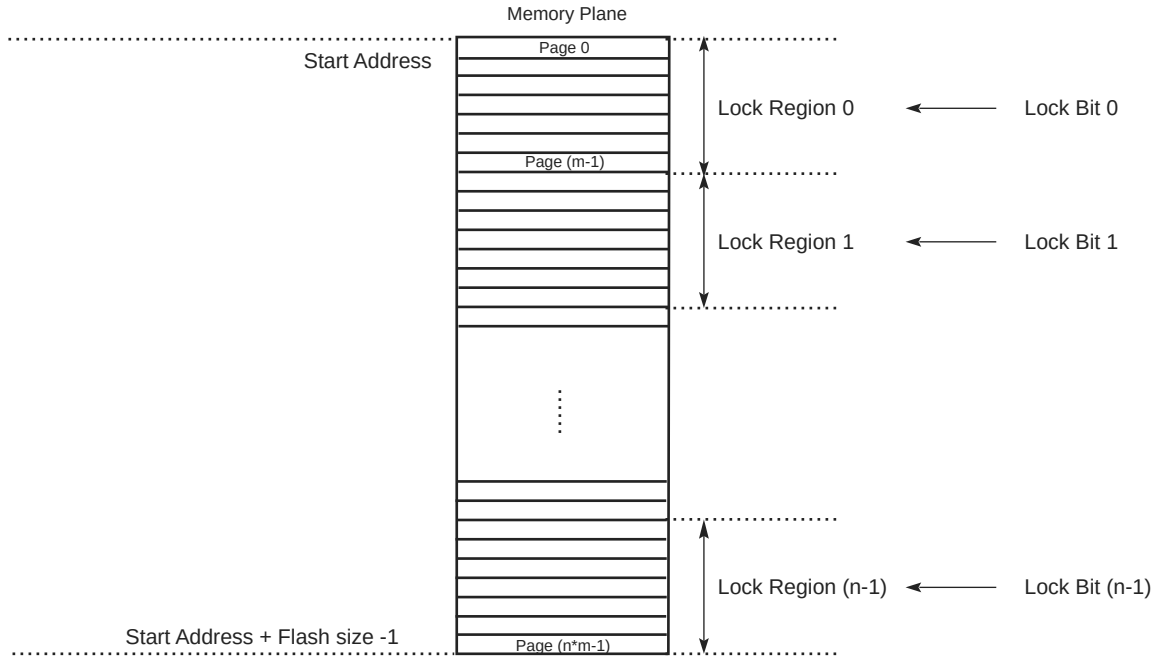
Memory in which:
a byte or halfword at a word-aligned address is the most significant byte or halfword within the word at that address,
a byte at a halfword-aligned address is the most significant byte within the halfword at that address.

See also "Little-endian memory" .

Breakpoint

A breakpoint is a mechanism provided by debuggers to identify an instruction at which program execution is to be halted. Breakpoints are inserted by the programmer to enable inspection of register contents, memory locations, variable values at fixed points in the program execution to test that the program is operating correctly. Breakpoints are removed after the program is successfully tested.

Figure 20-2. Organization of Embedded Flash for Code

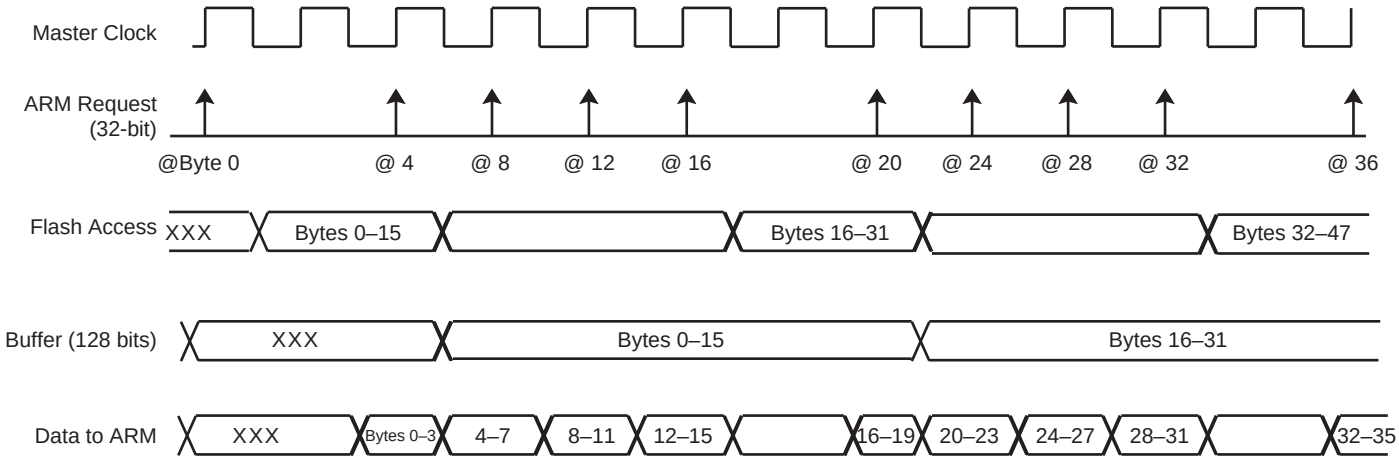


20.4.2.4 Data Read Optimization

The organization of the Flash in 128 bits or 64 bits is associated with two 128-bit or 64-bit prefetch buffers and one 128-bit or 64-bit data read buffer, thus providing maximum system performance. This buffer is added in order to store the requested data plus all the data contained in the 128-bit or 64-bit aligned data. This speeds up sequential data reads if, for example, FWS is equal to 1 (see Figure 20-6). The data read optimization is enabled by default. If the bit EEFC_FMR.SCOD is set to 1, this buffer is disabled and the data read is no longer optimized.

Note: No consecutive data read accesses are mandatory to benefit from this optimization.

Figure 20-6. Data Read Optimization for FWS = 1



20.4.3 Flash Commands

The EEFC offers a set of commands to manage programming the Flash memory, locking and unlocking lock regions, consecutive programming, locking and full Flash erasing, etc.

The commands are listed in the following table.

Table 20-2. Set of Commands

Command	Value	Mnemonic
Get Flash descriptor	0x00	GETD
Write page	0x01	WP
Write page and lock	0x02	WPL
Erase page and write page	0x03	EWP
Erase page and write page then lock	0x04	EWPL
Erase all	0x05	EA
Erase pages	0x07	EPA
Set lock bit	0x08	SLB
Clear lock bit	0x09	CLB
Get lock bit	0x0A	GLB
Set GPNVM bit	0x0B	SGPB
Clear GPNVM bit	0x0C	CGPB

22.5.1 Cache Controller Type Register

Name: CMCC_TYPE

Address: 0x4007C000

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–		CLSIZE			CSIZE		

7	6	5	4	3	2	1	0
LCKDOWN	WAYNUM		RRP	LRUP	RANDP	GCLK	AP

- **AP: Access Port Access Allowed**

0: Access Port Access is disabled.

1: Access Port Access is enabled.

- **GCLK: Dynamic Clock Gating Supported**

0: Cache controller does not support clock gating.

1: Cache controller uses dynamic clock gating.

- **RANDP: Random Selection Policy Supported**

0: Random victim selection is not supported.

1: Random victim selection is supported.

- **LRUP: Least Recently Used Policy Supported**

0: Least Recently Used Policy is not supported.

1: Least Recently Used Policy is supported.

- **RRP: Random Selection Policy Supported**

0: Random Selection Policy is not supported.

1: Random Selection Policy is supported.

- **WAYNUM: Number of Ways**

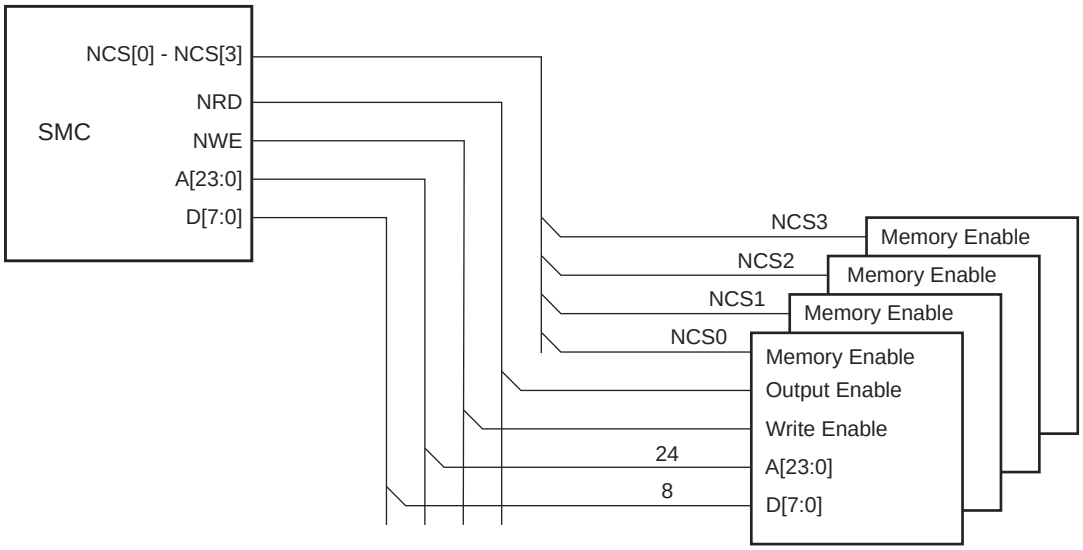
Value	Name	Description
0	DMAPPED	Direct Mapped Cache
1	ARCH2WAY	2-way set associative
2	ARCH4WAY	4-way set associative
3	ARCH8WAY	8-way set associative

26.6 External Memory Mapping

The SMC provides up to 24 address lines, A[23:0]. This allows each chip select line to address up to 16 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 16 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 26-1).

Figure 26-1. Memory Connections for Four External Devices



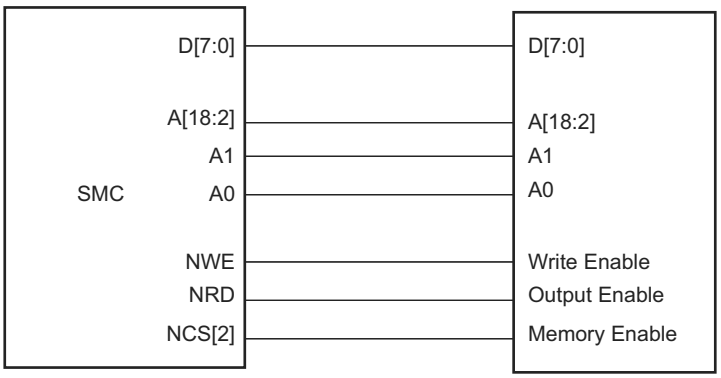
26.7 Connection to External Devices

26.7.1 Data Bus Width

The data bus width is 8 bits.

Figure 26-2 shows how to connect a 512-Kbyte x 8-bit memory on NCS2.

Figure 26-2. Memory Connection for an 8-bit Data Bus



27.6.6 Receive Next Counter Register

Name: PERIPH_RNCR

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RXNCTR							
7	6	5	4	3	2	1	0
RXNCTR							

- **RXNCTR: Receive Next Counter**

RXNCTR contains the next receive buffer size.

When a half-duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

29.17.11PMC Master Clock Register

Name: PMC_MCKR

Address: 0x400E0430

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	PLLBDIV2	PLLADIV2	–	–	–	–

7	6	5	4	3	2	1	0
–	PRES			–	–	CSS	

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **CSS: Master Clock Source Selection**

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLA Clock is selected
3	PLLB_CLK	PLLBClock is selected

- **PRES: Processor Clock Prescaler**

Value	Name	Description
0	CLK_1	Selected clock
1	CLK_2	Selected clock divided by 2
2	CLK_4	Selected clock divided by 4
3	CLK_8	Selected clock divided by 8
4	CLK_16	Selected clock divided by 16
5	CLK_32	Selected clock divided by 32
6	CLK_64	Selected clock divided by 64
7	CLK_3	Selected clock divided by 3

- **PLLADIV2: PLLA Divisor by 2**

PLLADIV2	PLLA Clock Division
0	PLLA clock frequency is divided by 1.
1	PLLA clock frequency is divided by 2.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

Table 32-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SSC	RD	PA18	A
SSC	RF	PA20	A
SSC	RK	PA19	A
SSC	TD	PA17	A
SSC	TF	PA15	A
SSC	TK	PA16	A

32.7.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

32.7.3 Interrupt

The SSC interface has an interrupt line connected to the interrupt controller. Handling interrupts requires programming the interrupt controller before configuring the SSC.

All SSC interrupts can be enabled/disabled configuring the SSC Interrupt Mask Register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC Interrupt Status Register.

Table 32-3. Peripheral IDs

Instance	ID
SSC	22

The command ALL_SEND_CID and the fields and values for the HSMCI_CMDR are described in Table 38-6 and Table 38-7.

Table 38-6. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Response	Abbreviation	Command Description
CMD2	bcr[1]	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: 1. bcr means broadcast command with response.

Table 38-7. Fields and Values for HSMCI_CMDR

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The HSMCI_ARGR contains the argument field of the command.

To send a command, the user must perform the following steps:

- Fill the argument register (HSMCI_ARGR) with the command argument.
- Set the command register (HSMCI_CMDR) (see Table 38-7).

The command is sent immediately after writing the command register.

While the card maintains a busy indication (at the end of a STOP_TRANSMISSION command CMD12, for example), a new command shall not be sent. The NOTBUSY flag in the Status Register (HSMCI_SR) is asserted when the card releases the busy indication.

If the command requires a response, it can be read in the HSMCI Response Register (HSMCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The HSMCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the HSMCI Interrupt Enable Register (HSMCI_IER) allows using an interrupt method.

38.14.11HSMCI Transmit Data Register

Name: HSMCI_TDR

Address: 0x40000034

Access: Write-only

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

- **DATA: Data to Write**

39.7.7 PWM Interrupt Mask Register 1

Name: PWM_IMR1

Address: 0x40020018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Mask**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Mask**

- **WAKEUP: Disable USB Bus Interrupt**

0: No effect

1: Disables USB Bus Wakeup Interrupt

- **TRANSFER: Hold Time**

  The TRANSFER field should be set to 2 to guarantee the optimal hold time.

- **USEQ: Use Sequence Enable**

Value	Name	Description
0	NUM_ORDER	Normal Mode: The controller converts channels in a simple numeric order depending only on the channel index.
1	REG_ORDER	User Sequence Mode: The sequence respects what is defined in ADC_SEQR1 and ADC_SEQR2 registers and can be used to convert the same channel several times.

42.7.5 ADC Channel Enable Register

Name: ADC_CHER

Address: 0x40038010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

- **CHx: Channel x Enable**

0: No effect.

1: Enables the corresponding channel.

Note: If USEQ = 1 in the ADC_MR, CHx corresponds to the xth channel of the sequence described in ADC_SEQR1 and ADC_SEQR2.

Table 44-44 is a computation example for the above formula, where V_{ADVREF} = 3V.

Table 44-44. Input Voltage Values in Single-ended Mode, OFFx = 0

Ci	Gain = 1	Gain = 2	Gain = 4
0	0	0	0
2047	1.5	0.75	0.375
4095	3	1.5	0.75

44.8.4.3 Example of LSB Computation

The LSB is relative to the analog scale V_{ADVREF}.

The term LSB expresses the quantization step in volts, also used for one ADC code variation.

- Single-ended (SE) (ex: V_{ADVREF} = 3.0V)
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5V / 4096) = 366 µV
  - Gain = 4, LSB = (750 mV / 4096) = 183 µV
- Differential (DIFF) (ex: V_{ADVREF} = 3.0V)
  - Gain = 0.5, LSB = (6.0V / 4096) = 1465 µV
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5V / 4096) = 366 µV

44.8.5 ADC Electrical Characteristics

The gain error depends on the gain value and the OFFx bit. The data are given with and without autocorrection at T_A 27°C. The data include the ADC performances as the PGA and ADC core cannot be separated. The temperature and voltage dependency are given as separate parameters.

Table 44-45. Voltage and Temperature Dependencies

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
α_G	Gain Temperature dependency	-40°C to 105°C	–	–	5	ppm/°C
α_{GV}	Gain Supply dependency	V_{DDIN}	–	–	0.025	%/V
α_O	Offset Temperature dependency	-40°C to 105°C	–	–	5	ppm/°C
α_{OV}	Offset Supply dependency	V_{DDIN}	–	–	0.025	%/V

44.8.5.1 Gain and Offset Errors

For:

- a given gain error: E_G (%)
- a given ideal code (Ci)
- a given offset error: E_O (LSB)

the actual code (Ca) is calculated using the following formula:

$$Ca = \left(1 + \frac{E_G}{100}\right) \times (Ci - 2047) + 2047 + E_O$$