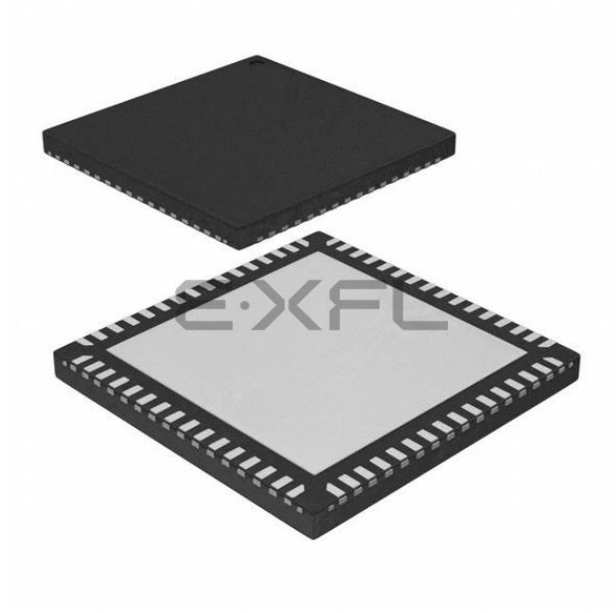


Welcome to E-XFL.COM

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	I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	47
Program Memory Size	256KB (256K 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 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4s4bb-mnr

3. If the returned status bit from step 2 indicates that the Store-Exclusive instruction succeeded then the software has claimed the semaphore. However, if the Store-Exclusive instruction failed, another process might have claimed the semaphore after the software performed the first step.

The Cortex-M4 includes an exclusive access monitor, that tags the fact that the processor has executed a Load-Exclusive instruction. If the processor is part of a multiprocessor system, the system also globally tags the memory locations addressed by exclusive accesses by each processor.

The processor removes its exclusive access tag if:

- It executes a CLREX instruction
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs. This means that the processor can resolve semaphore conflicts between different threads.

In a multiprocessor implementation:

- Executing a CLREX instruction removes only the local exclusive access tag for the processor
- Executing a Store-Exclusive instruction, or an exception, removes the local exclusive access tags, and all global exclusive access tags for the processor.

For more information about the synchronization primitive instructions, see “LDREX and STREX” and “CLREX” .

12.4.2.8 Programming Hints for the Synchronization Primitives

ISO/IEC C cannot directly generate the exclusive access instructions. CMSIS provides intrinsic functions for generation of these instructions:

Table 12-8. CMSIS Functions for Exclusive Access Instructions

Instruction	CMSIS Function
LDREX	uint32_t __LDREXW (uint32_t *addr)
LDREXH	uint16_t __LDREXH (uint16_t *addr)
LDREXB	uint8_t __LDREXB (uint8_t *addr)
STREX	uint32_t __STREXW (uint32_t value, uint32_t *addr)
STREXH	uint32_t __STREXH (uint16_t value, uint16_t *addr)
STREXB	uint32_t __STREXB (uint8_t value, uint8_t *addr)
CLREX	void __CLREX (void)

The actual exclusive access instruction generated depends on the data type of the pointer passed to the intrinsic function. For example, the following C code generates the required LDREXB operation:

```
__ldrex((volatile char *) 0xFF);
```

12.4.3 Exception Model

This section describes the exception model.

12.4.3.1 Exception States

Each exception is in one of the following states:

Inactive

The exception is not active and not pending.

Pending

The exception is waiting to be serviced by the processor.

12.6.4.4 LDR and STR, Unprivileged

Load and Store with unprivileged access.

Syntax

```
op{type}T{cond} Rt, [Rn {, #offset}]      ; immediate offset
```

where:

op is one of:

- LDR Load Register.
- STR Store Register.

type is one of:

- B unsigned byte, zero extend to 32 bits on loads.
- SB signed byte, sign extend to 32 bits (LDR only).
- H unsigned halfword, zero extend to 32 bits on loads.
- SH signed halfword, sign extend to 32 bits (LDR only).
- - omit, for word.

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

Rt is the register to load or store.

Rn is the register on which the memory address is based.

offset is an offset from *Rn* and can be 0 to 255.

If *offset* is omitted, the address is the value in *Rn*.

Operation

These load and store instructions perform the same function as the memory access instructions with immediate offset, see “LDR and STR, Immediate Offset” . The difference is that these instructions have only unprivileged access even when used in privileged software.

When used in unprivileged software, these instructions behave in exactly the same way as normal memory access instructions with immediate offset.

Restrictions

In these instructions:

- *Rn* must not be PC
- *Rt* must not be SP and must not be PC.

Condition Flags

These instructions do not change the flags.

Examples

```
STRBTEQ  R4, [R7]       ; Conditionally store least significant byte in
                        ; R4 to an address in R7, with unprivileged access
LDRHT    R2, [R2, #8]   ; Load halfword value from an address equal to
                        ; sum of R2 and 8 into R2, with unprivileged access
```

REVSH Reverse byte order in the bottom halfword, and sign extend to 32 bits.

RBIT Reverse the bit order in a 32-bit word.

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

Rd is the destination register.

Rn is the register holding the operand.

Operation

Use these instructions to change endianness of data:

REV converts either:

- 32-bit big-endian data into little-endian data
- 32-bit little-endian data into big-endian data.

REV16 converts either:

- 16-bit big-endian data into little-endian data
- 16-bit little-endian data into big-endian data.

REVSH converts either:

- 16-bit signed big-endian data into 32-bit signed little-endian data
- 16-bit signed little-endian data into 32-bit signed big-endian data.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
REV   R3, R7; Reverse byte order of value in R7 and write it to R3
REV16 R0, R0; Reverse byte order of each 16-bit halfword in R0
REVSH R0, R5; Reverse Signed Halfword
REVHS R3, R7; Reverse with Higher or Same condition
RBIT  R7, R8; Reverse bit order of value in R8 and write the result to R7.
```

- Do not use SP and do not use PC.
- *RdHi* and *RdLo* must be different registers.

Condition Flags

These instructions do not affect the condition code flags.

Examples

```
SMLAL    R4, R5, R3, R8  ; Multiplies R3 and R8, adds R5:R4 and writes to
                         ; R5:R4
SMLALBT  R2, R1, R6, R7  ; Multiplies bottom halfword of R6 with top
                         ; halfword of R7, sign extends to 32-bit, adds
                         ; R1:R2 and writes to R1:R2
SMLALTB  R2, R1, R6, R7  ; Multiplies top halfword of R6 with bottom
                         ; halfword of R7,sign extends to 32-bit, adds R1:R2
                         ; and writes to R1:R2
SMLALD   R6, R8, R5, R1  ; Multiplies top halfwords in R5 and R1 and bottom
                         ; halfwords of R5 and R1, adds R8:R6 and writes to
                         ; R8:R6
SMLALDX  R6, R8, R5, R1  ; Multiplies top halfword in R5 with bottom
                         ; halfword of R1, and bottom halfword of R5 with
                         ; top halfword of R1, adds R8:R6 and writes to
                         ; R8:R6.
```

12.11 Memory Protection Unit (MPU)

The MPU divides the memory map into a number of regions, and defines the location, size, access permissions, and memory attributes of each region. It supports:

- Independent attribute settings for each region
- Overlapping regions
- Export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M4 MPU defines:

- Eight separate memory regions, 0–7
- A background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M4 MPU memory map is unified. This means that instruction accesses and data accesses have the same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault. This causes a fault exception, and might cause the termination of the process in an OS environment.

In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

The configuration of MPU regions is based on memory types (see "Memory Regions, Types and Attributes").

Table 12-35 shows the possible MPU region attributes. These include Share ability and cache behavior attributes that are not relevant to most microcontroller implementations. See "MPU Configuration for a Microcontroller" for guidelines for programming such an implementation.

Table 12-35. Memory Attributes Summary

Memory Type	Shareability	Other Attributes	Description
Strongly-ordered	–	–	All accesses to Strongly-ordered memory occur in program order. All Strongly-ordered regions are assumed to be shared.
Device	Shared	–	Memory-mapped peripherals that several processors share.
	Non-shared	–	Memory-mapped peripherals that only a single processor uses.
Normal	Shared	Non-cacheable Write-through Cacheable Write-back Cacheable	Normal memory that is shared between several processors.
	Non-shared	Non-cacheable Write-through Cacheable Write-back Cacheable	Normal memory that only a single processor uses.

12.11.1 MPU Access Permission Attributes

This section describes the MPU access permission attributes. The access permission bits (TEX, C, B, S, AP, and XN) of the MPU_RASR control the access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

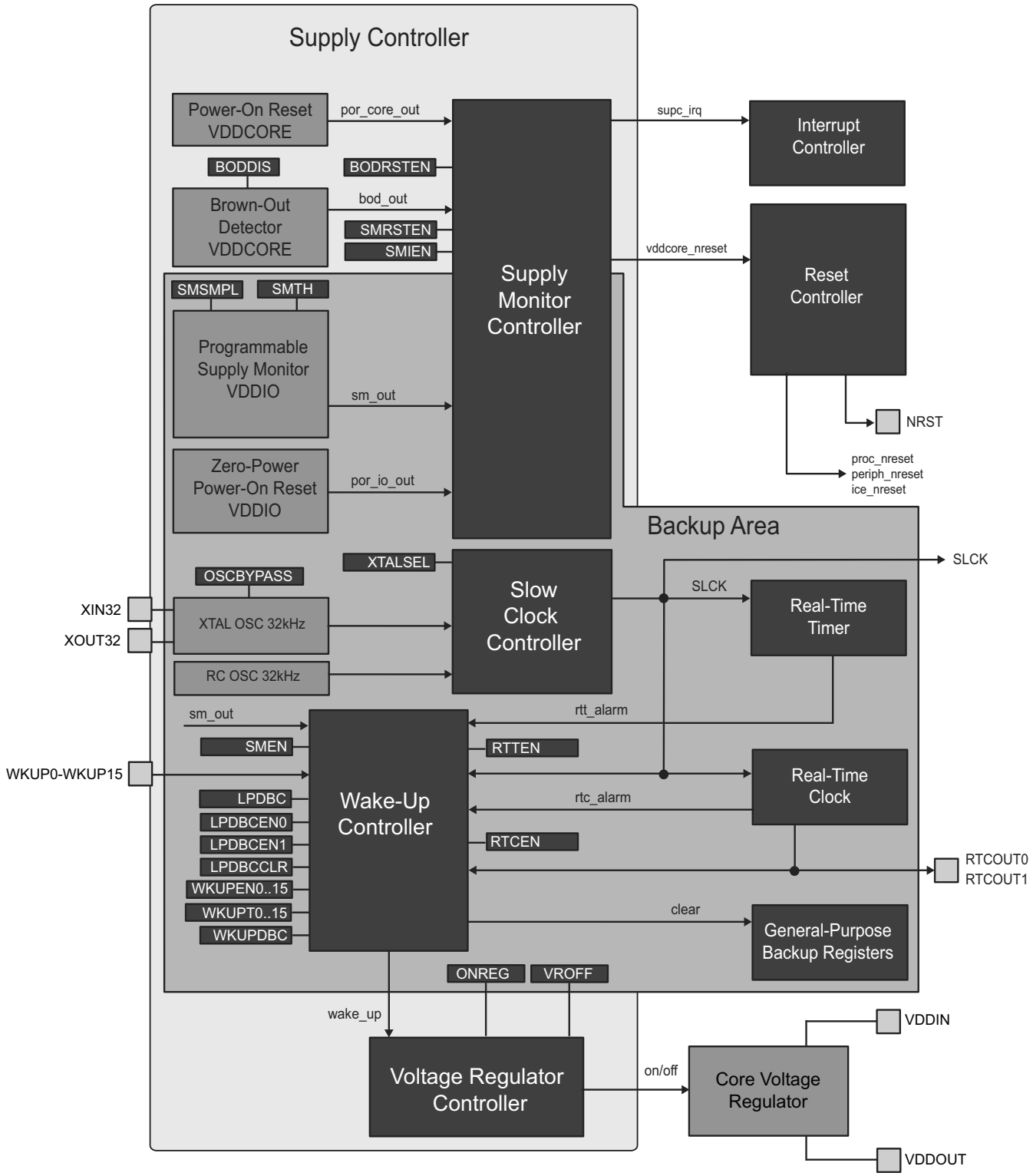
- **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC_SR depends on the value of CALEVSEL

Value	Name	Description
0	WEEK	Week change (every Monday at time 00:00:00)
1	MONTH	Month change (every 01 of each month at time 00:00:00)
2	YEAR	Year change (every January 1 at time 00:00:00)

18.3 Block Diagram

Figure 18-1. Supply Controller Block Diagram



25.8.5 SMC NAND Flash Chip Select Configuration Register

Name: CCFG_SMCNFCS

Address: 0x400E031C

Type: Read/Write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	SMC_NFCS3	SMC_NFCS2	SMC_NFCS1	SMC_NFCS0

- **SMC_NFCS0: SMC NAND Flash Chip Select 0 Assignment**

0: NCS0 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS0)

1: NCS0 is assigned to a NAND Flash (NANDOE and NANWE used for NCS0)

- **SMC_NFCS1: SMC NAND Flash Chip Select 1 Assignment**

0: NCS1 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS1)

1: NCS1 is assigned to a NAND Flash (NANDOE and NANWE used for NCS1)

- **SMC_NFCS2: SMC NAND Flash Chip Select 2 Assignment**

0: NCS2 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS2)

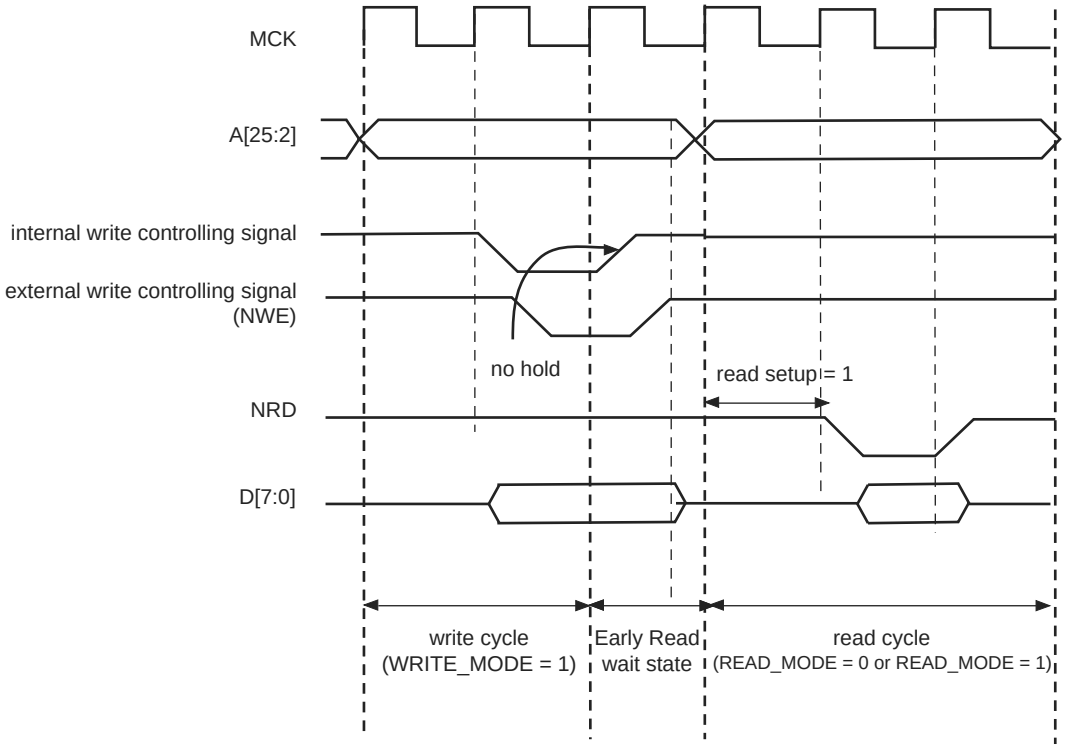
1: NCS2 is assigned to a NAND Flash (NANDOE and NANWE used for NCS2)

- **SMC_NFCS3: SMC NAND Flash Chip Select 3 Assignment**

0: NCS3 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS3)

1: NCS3 is assigned to a NAND Flash (NANDOE and NANWE used for NCS3)

Figure 26-16. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



26.11.3 Reload User Configuration Wait State

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. This “Reload User Configuration Wait State” is used by the SMC to load the new set of parameters to apply to next accesses.

The Reload Configuration Wait State is not applied in addition to the Chip Select Wait State. If accesses before and after re-programming the user interface are made to different devices (Chip Selects), then one single Chip Select Wait State is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a Reload Configuration Wait State is inserted, even if the change does not concern the current Chip Select.

26.11.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any SMC_MODE register of the user interface. If the user only modifies timing registers (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, he must validate the modification by writing the SMC_MODE, even if no change was made on the mode parameters.

The user must not change the configuration parameters of an SMC Chip Select (Setup, Pulse, Cycle, Mode) if accesses are performed on this CS during the modification. Any change of the Chip Select parameters, while fetching the code from a memory connected on this CS, may lead to unpredictable behavior. The instructions used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another CS.

Figure 29-8. Change PLLx Programming

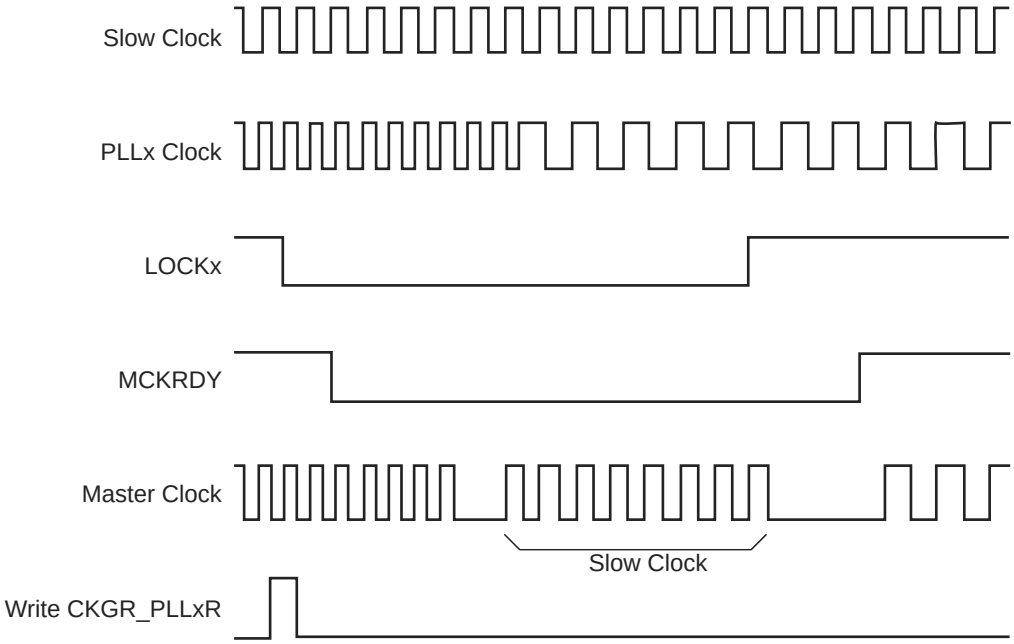
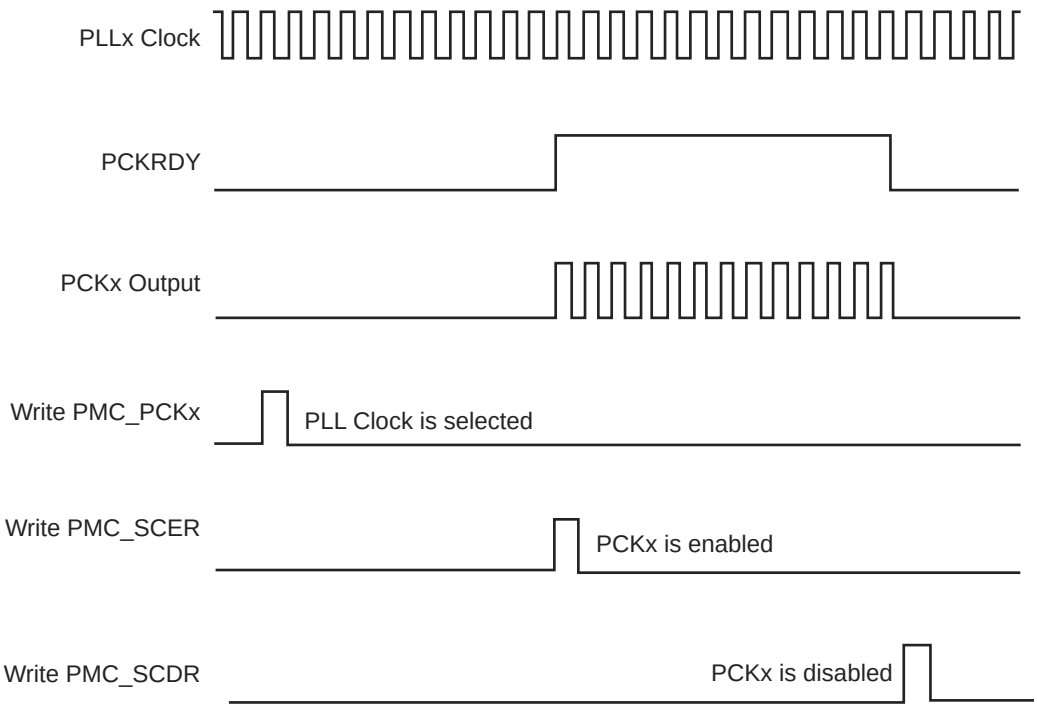


Figure 29-9. Programmable Clock Output Programming



29.17.1 PMC System Clock Enable Register

Name: PMC_SCER

Address: 0x400E0400

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
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	–	–	–	–	–	–	–

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

- **UDP: USB Device Port Clock Enable**

0: No effect.

1: Enables the 48 MHz clock (UDPCK) of the USB Device Port.

- **PCKx: Programmable Clock x Output Enable**

0: No effect.

1: Enables the corresponding Programmable Clock output.

29.17.9 PMC Clock Generator PLLA Register

Name: CKGR_PLLAR

Address: 0x400E0428

Access: Read/Write

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

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
–	–	PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLLA input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

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

- **DIVA:** PLLA Front_End Divider

0: Divider output is stuck at 0 and PLLA is disabled.

1: Divider is bypassed (divide by 1) PLLA is enabled

2–255: Clock is divided by DIVA

- **PLLACOUNT: PLLA Counter**

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **MULA: PLLA Multiplier**

0: The PLLA is deactivated (PLLA also disabled if DIVA = 0).

7 up to 62 = The PLLA Clock frequency is the PLLA input frequency multiplied by MULA + 1.

Unlisted values are forbidden.

- **ONE: Must Be Set to 1**

Bit 29 must always be set to 1 when programming the CKGR_PLLAR.

33.8.8 SPI Interrupt Mask Register

Name: SPI_IMR

Address: 0x4000801C

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
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **NSSR: NSS Rising Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**
- **UNDES: Underrun Error Interrupt Mask**

38.14.3 HSMCI Data Timeout Register

Name: HSMCI_DTOR

Address: 0x40000008

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	DTOMUL			DTOCYC			

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

- **DTOCYC: Data Timeout Cycle Number**

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

- **DTOMUL: Data Timeout Multiplier**

Value	Name	Description
0	1	DTOCYC
1	16	DTOCYC x 16
2	128	DTOCYC x 128
3	256	DTOCYC x 256
4	1024	DTOCYC x 1024
5	4096	DTOCYC x 4096
6	65536	DTOCYC x 65536
7	1048576	DTOCYC x 1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the HSMCI Status Register (HSMCI_SR) rises.

38.14.4 HSMCI SDCard/SDIO Register

Name: HSMCI_SDCR

Address: 0x4000000C

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SDCBUS		–	–	–	–	SDCSEL	

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

- **SDCSEL: SDCard/SDIO Slot**

Value	Name	Description
0	SLOTA	Slot A is selected.
1	SLOTB	–
2	SLOTC	–
3	SLOTD	–

- **SDCBUS: SDCard/SDIO Bus Width**

Value	Name	Description
0	1	1 bit
1	–	Reserved
2	4	4 bits
3	8	8 bits

38.14.17HSMCI Write Protection Mode Register

Name: HSMCI_WPMR

Address: 0x400000E4

Access: Read/Write

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

- **WPEN: Write Protect Enable**

0: Disables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

See Section 38.13 “Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x4D4349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

39.7.8 PWM Interrupt Status Register 1

Name: PWM_ISR1

Address: 0x4002001C

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**

0: No new counter event has occurred since the last read of the PWM_ISR1.

1: At least one counter event has occurred since the last read of the PWM_ISR1.

- **FCHIDx: Fault Protection Trigger on Channel x**

0: No new trigger of the fault protection since the last read of the PWM_ISR1.

1: At least one trigger of the fault protection since the last read of the PWM_ISR1.

Note: Reading PWM_ISR1 automatically clears CHIDx and FCHIDx flags.

Value	Name	Description
6	BULK_IN	Bulk IN
3	INT_OUT	Interrupt OUT
7	INT_IN	Interrupt IN

- **DTGLE: Data Toggle (Read-only)**

0: Identifies DATA0 packet

1: Identifies DATA1 packet

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

- **EPEDS: Endpoint Enable Disable**

Read:

0: Endpoint disabled

1: Endpoint enabled

Write:

0: Disables endpoint

1: Enables endpoint

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset, all endpoints are configured as control endpoints (zero).

- **RXBYTECNT[10:0]: Number of Bytes Available in the FIFO (Read-only)**

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_FDRx.

For power-saving options, see Section 43.6.6 ”DACC Timings”.

43.5.2 Interrupt Sources

The DACC interrupt line is connected to one of the internal sources of the interrupt controller. Using the DACC interrupt requires the interrupt controller to be programmed first.

Table 43-2. Peripheral IDs

Instance	ID
DACC	30

43.5.3 Conversion Performances

For performance and electrical characteristics of the DACC, see the product DC Characteristics section of the datasheet.

43.7 Digital-to-Analog Converter Controller (DACC) User Interface

Table 43-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	DACC_CR	Write-only	–
0x04	Mode Register	DACC_MR	Read/Write	0x00000000
0x08–0x0C	Reserved	–	–	–
0x10	Channel Enable Register	DACC_CHER	Write-only	–
0x14	Channel Disable Register	DACC_CHDR	Write-only	–
0x18	Channel Status Register	DACC_CHSR	Read-only	0x00000000
0x1C	Reserved	–	–	–
0x20	Conversion Data Register	DACC_CDR	Write-only	–
0x24	Interrupt Enable Register	DACC_IER	Write-only	–
0x28	Interrupt Disable Register	DACC_IDR	Write-only	–
0x2C	Interrupt Mask Register	DACC_IMR	Read-only	0x00000000
0x30	Interrupt Status Register	DACC_ISR	Read-only	0x00000000
0x34–0x90	Reserved	–	–	–
0x94	Analog Current Register	DACC_ACR	Read/Write	0x00000000
0x98–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	DACC_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	DACC_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–