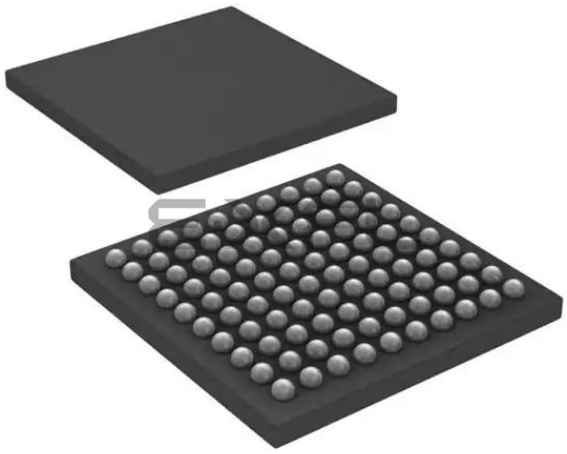


Welcome to E-XFL.COM

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	2MB (2M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	160K 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 ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-VFBGA
Supplier Device Package	100-VFBGA (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4sd32cb-cfn

- If subtracting a negative value from a positive value generates a negative value.

The Compare operations are identical to subtracting, for CMP, or adding, for CMN, except that the result is discarded. See the instruction descriptions for more information.

Note: Most instructions update the status flags only if the S suffix is specified. See the instruction descriptions for more information.

Condition Code Suffixes

The instructions that can be conditional have an optional condition code, shown in syntax descriptions as {cond}. Conditional execution requires a preceding IT instruction. An instruction with a condition code is only executed if the condition code flags in the APSR meet the specified condition. Table 12-16 shows the condition codes to use.

A conditional execution can be used with the IT instruction to reduce the number of branch instructions in code.

Table 12-16 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

Table 12-16. Condition Code Suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned ≥
CC or LO	C = 0	Lower, unsigned <
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned >
LS	C = 0 or Z = 1	Lower or same, unsigned ≤
GE	N = V	Greater than or equal, signed ≥
LT	N != V	Less than, signed <
GT	Z = 0 and N = V	Greater than, signed >
LE	Z = 1 and N != V	Less than or equal, signed ≤
AL	Can have any value	Always. This is the default when no suffix is specified.

Absolute Value

The example below shows the use of a conditional instruction to find the absolute value of a number. R0 = ABS(R1).

```
MOVS    R0, R1          ; R0 = R1, setting flags
IT      MI              ; IT instruction for the negative condition
RSBMI   R0, R1, #0      ; If negative, R0 = -R1
```

Compare and Update Value

The example below shows the use of conditional instructions to update the value of R4 if the signed values R0 is greater than R1 and R2 is greater than R3.

```
CMP     R0, R1   ; Compare R0 and R1, setting flags
ITT     GT       ; IT instruction for the two GT conditions
CMPGT   R2, R3   ; If 'greater than', compare R2 and R3, setting flags
MOVGT   R4, R5   ; If still 'greater than', do R4 = R5
```

Examples

```
SMUAD    R0, R4, R5  ; Multiplies bottom halfword of R4 with the bottom
                     ; halfword of R5, adds multiplication of top halfword
                     ; of R4 with top halfword of R5, writes to R0
SMUADX   R3, R7, R4  ; Multiplies bottom halfword of R7 with top halfword
                     ; of R4, adds multiplication of top halfword of R7
                     ; with bottom halfword of R4, writes to R3
SMUSD    R3, R6, R2  ; Multiplies bottom halfword of R4 with bottom halfword
                     ; of R6, subtracts multiplication of top halfword of R6
                     ; with top halfword of R3, writes to R3
SMUSDX   R4, R5, R3  ; Multiplies bottom halfword of R5 with top halfword of
                     ; R3, subtracts multiplication of top halfword of R5
                     ; with bottom halfword of R3, writes to R4.
```

12.6.6.10 SMUL and SMULW

Signed Multiply (halfwords) and Signed Multiply (word by halfword)

Syntax

op{*XY*}{*cond*} *Rd*,*Rn*, *Rm*
op{*Y*}{*cond*} *Rd*. *Rn*, *Rm*

For *SMULXY* only:

op is one of:

SMUL{*XY*} Signed Multiply (halfwords).

X and *Y* specify which halfword of the source registers *Rn* and *Rm* is used as the first and second multiply operand.
If *X* is B, then the bottom halfword, bits [15:0] of *Rn* is used.
If *X* is T, then the top halfword, bits [31:16] of *Rn* is used.If *Y* is B, then the bot tom halfword, bits [15:0], of *Rm* is used.
If *Y* is T, then the top halfword, bits [31:16], of *Rm* is used.

SMULW{Y} Signed Multiply (word by halfword).

Y specifies which halfword of the source register *Rm* is used as the second mul tiply operand.
If *Y* is B, then the bottom halfword (bits [15:0]) of *Rm* is used.
If *Y* is T, then the top halfword (bits [31:16]) of *Rm* is used.

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 SMULBB, SMULTB, SMULBT and SMULTT instructions interprets the values from *Rn* and *Rm* as four signed 16-bit integers. These instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from *Rn* and *Rm*.
- Writes the 32-bit result of the multiplication in *Rd.*

The SMULWT and SMULWB instructions interprets the values from *Rn* as a 32-bit signed integer and *Rm* as two halfword 16-bit signed integers. These instructions:

- Multiplies the first operand and the top, T suffix, or the bottom, B suffix, halfword of the second operand.
- Writes the signed most significant 32 bits of the 48-bit result in the destination register.

Restrictions

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the flags.

Examples

```
      SXTAH  R4, R8, R6, ROR #16 ; Rotates R6 right by 16 bits, obtains bottom
                                 ; halfword, sign extends to 32 bits, adds
                                 ; R8,and writes to R4
      UXTAB  R3, R4, R10         ; Extracts bottom byte of R10 and zero extends
                                 ; to 32 bits, adds R4, and writes to R3.
```

12.6.11.1 BKPT

Breakpoint.

Syntax

```
BKPT #imm
```

where:

imm is an expression evaluating to an integer in the range 0–255 (8-bit value).

Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

imm is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The BKPT instruction can be placed inside an IT block, but it executes unconditionally, unaffected by the condition specified by the IT instruction.

Condition Flags

This instruction does not change the flags.

Examples

```
BKPT 0xAB   ; Breakpoint with immediate value set to 0xAB (debugger can
            ; extract the immediate value by locating it using the PC)
```

Note: ARM does not recommend the use of the BKPT instruction with an immediate value set to 0xAB for any purpose other than Semi-hosting.

12.6.11.2 CPS

Change Processor State.

Syntax

```
CPSeffect iflags
```

where:

effect is one of:

- IE Clears the special purpose register.
- ID Sets the special purpose register.

iflags is a sequence of one or more flags:

- i Set or clear PRIMASK.
- f Set or clear FAULTMASK.

Operation

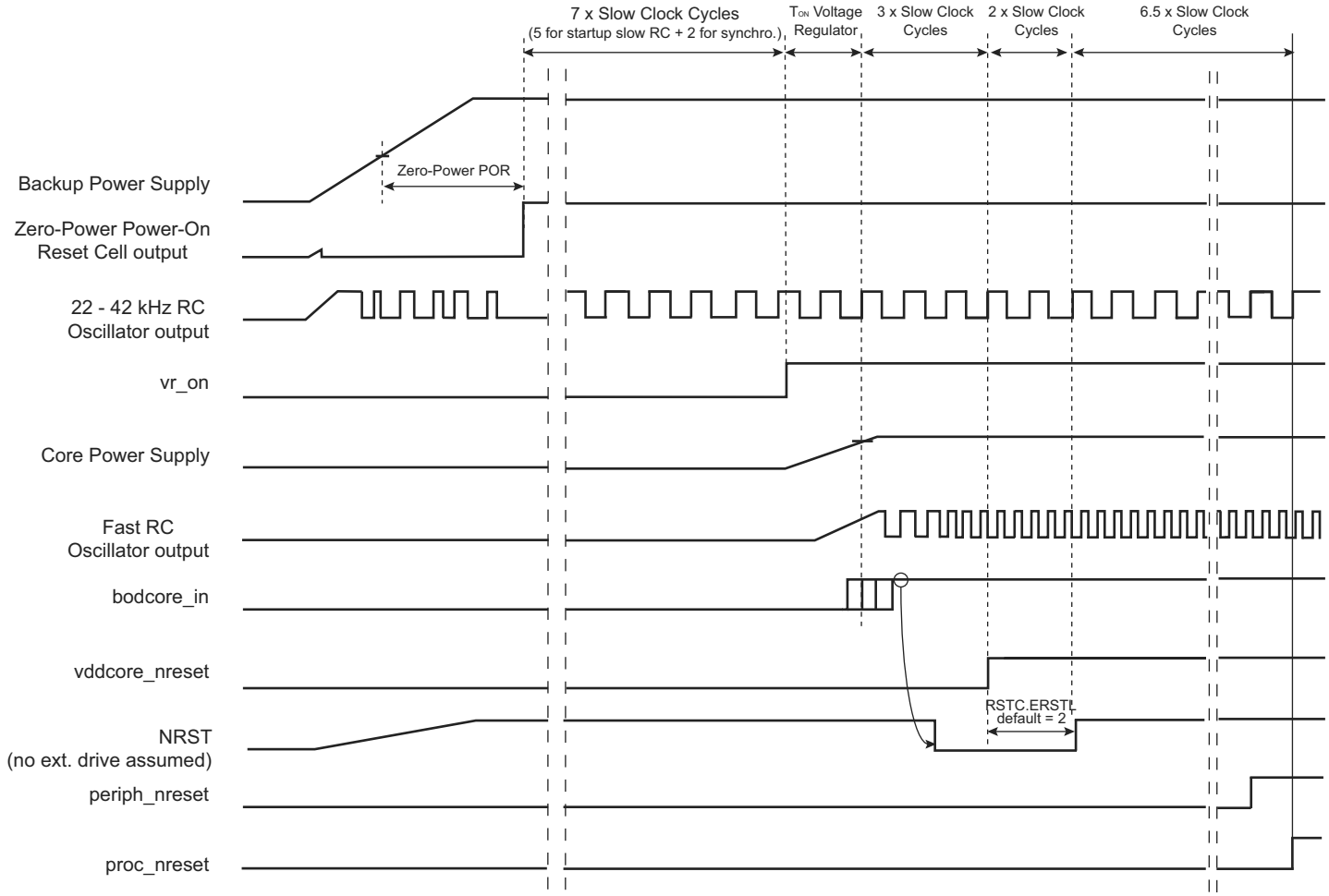
CPS changes the PRIMASK and FAULTMASK special register values. See “Exception Mask Registers” for more information about these registers.

Restrictions

The restrictions are:

- Use CPS only from privileged software, it has no effect if used in unprivileged software
- CPS cannot be conditional and so must not be used inside an IT block.

Figure 18-3. Raising the VDDIO Power Supply



Note: After “proc_nreset” rising, the core starts fetching instructions from Flash at 4 MHz.

18.4.6 Core Reset

The Supply Controller manages the vddcore_nreset signal to the Reset Controller, as described in Section 18.4.5 ”Backup Power Supply Reset”. The vddcore_nreset signal is normally asserted before shutting down the core power supply and released as soon as the core power supply is correctly regulated.

There are two additional sources which can be programmed to activate vddcore_nreset:

- a supply monitor detection
- a brownout detection

18.4.6.1 Supply Monitor Reset

The supply monitor is capable of generating a reset of the system. This is enabled by setting the SMRSTEN bit in SUPC_SMMR.

If SMRSTEN is set and if a supply monitor detection occurs, the vddcore_nreset signal is immediately activated for a minimum of one slow clock cycle.

18.4.6.2 Brownout Detector Reset

The brownout detector provides the bodcore_in signal to the SUPC. This signal indicates that the voltage regulation is operating as programmed. If this signal is lost for longer than 1 slow clock period while the voltage regulator is enabled, the SUPC asserts vddcore_nreset if BODRSTEN is written to 1 in SUPC_MR.

Table 26-2. I/O Lines

SMC	A16	PA20	C
SMC	A17	PA0	C
SMC	A18	PA1	C
SMC	A19	PA23	C
SMC	A20	PA24	C
SMC	A21/NANDALE	PC16	A
SMC	A22/NANDCLE	PC17	A
SMC	A23	PA25	C
SMC	D0	PC0	A
SMC	D1	PC1	A
SMC	D2	PC2	A
SMC	D3	PC3	A
SMC	D4	PC4	A
SMC	D5	PC5	A
SMC	D6	PC6	A
SMC	D7	PC7	A
SMC	NANDOE	PC9	A
SMC	NANDWE	PC10	A
SMC	NCS0	PC14	A
SMC	NCS1	PC15	A
SMC	NCS2	PA22	C
SMC	NCS3	PC12	A
SMC	NRD	PC11	A
SMC	NWAIT	PC13	A
SMC	NWE	PC8	A

26.4.2 Power Management

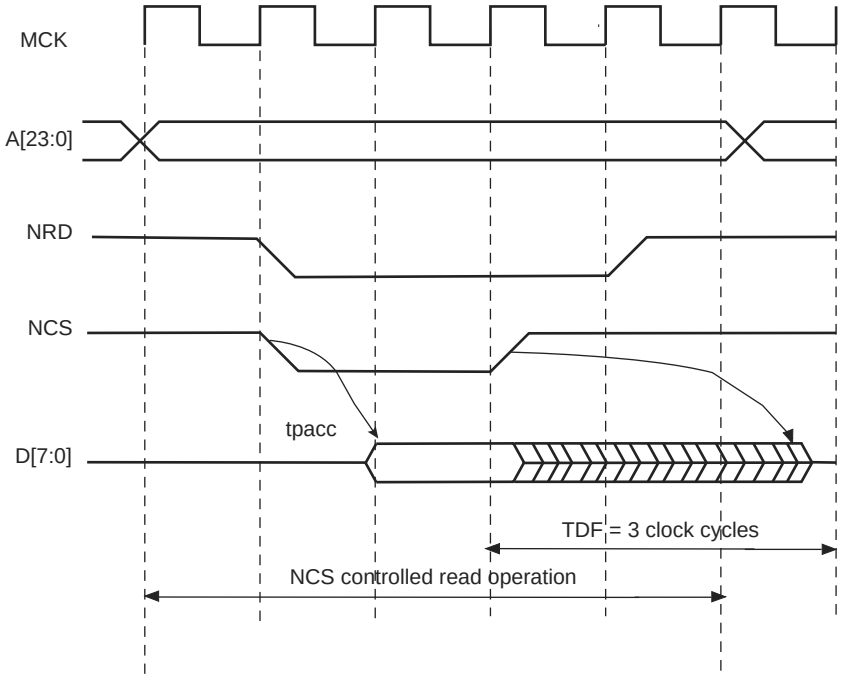
The SMC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SMC clock.

26.5 Multiplexed Signals

Table 26-3. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals		Related Function
A22	NANDCLE	NAND Flash Command Latch Enable
A21	NANDALE	NAND Flash Address Latch Enable

Figure 26-18. TDF Period in NCS Controlled Read Operation (TDF = 3)



26.12.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the SMC_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

Figure 26-19 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

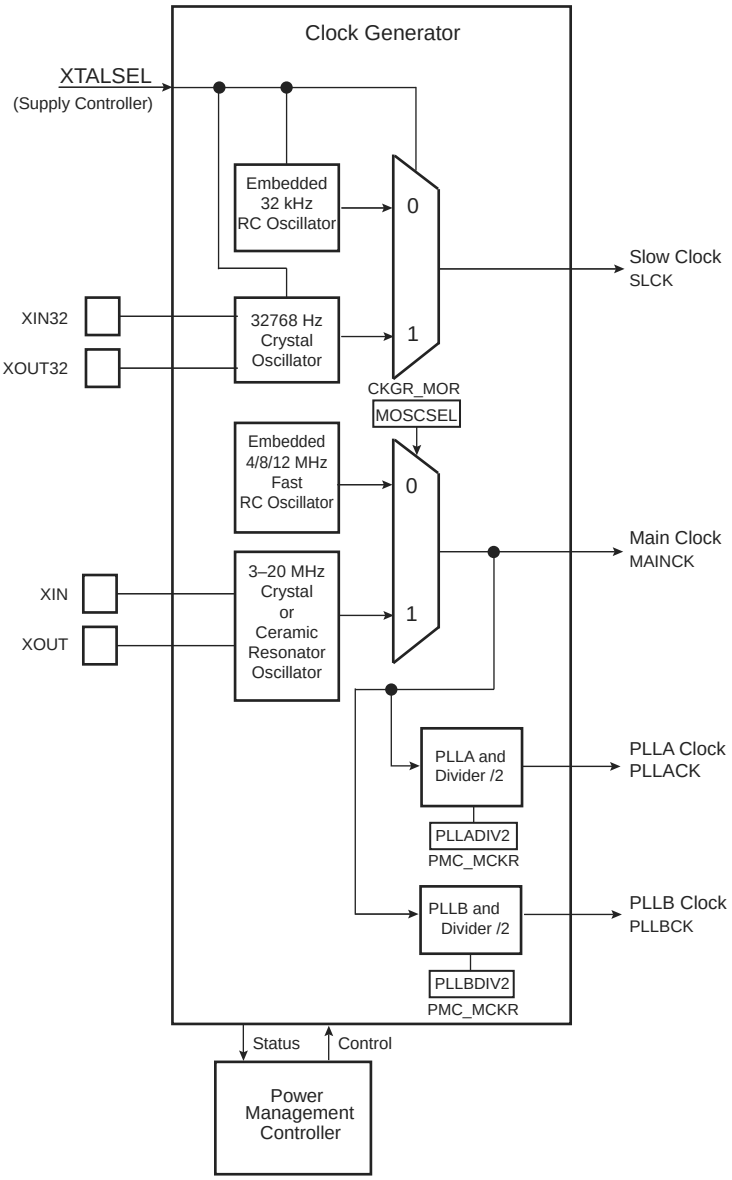
NRD_HOLD = 4; READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; WRITE_MODE = 1 (NWE controlled)

TDF_CYCLES = 6; TDF_MODE = 1 (optimization enabled).

28.3 Block Diagram

Figure 28-1. Clock Generator Block Diagram



29.17.10PMC Clock Generator PLLB Register

Name: CKGR_PLLBR

Address: 0x400E042C

Access: Read/Write

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

23	22	21	20	19	18	17	16
MULB							

15	14	13	12	11	10	9	8
–	–	PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

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

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

- **DIVB: PLLB Front-End Divider**

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

1: Divider is bypassed (divide by 1)

2–255: Clock is divided by DIVB

- **PLLBCOUNT: PLLB Counter**

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

- **MULB: PLLB Multiplier**

0: The PLLB is deactivated (PLLB also disabled if DIVB = 0).

7 up to 62: The PLLB Clock frequency is the PLLB input frequency multiplied by MULB + 1.

Unlisted values are forbidden.

30.3.1 Chip ID Register

Name: CHIPID_CIDR

Address: 0x400E0740

Access: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**

Current version of the device.

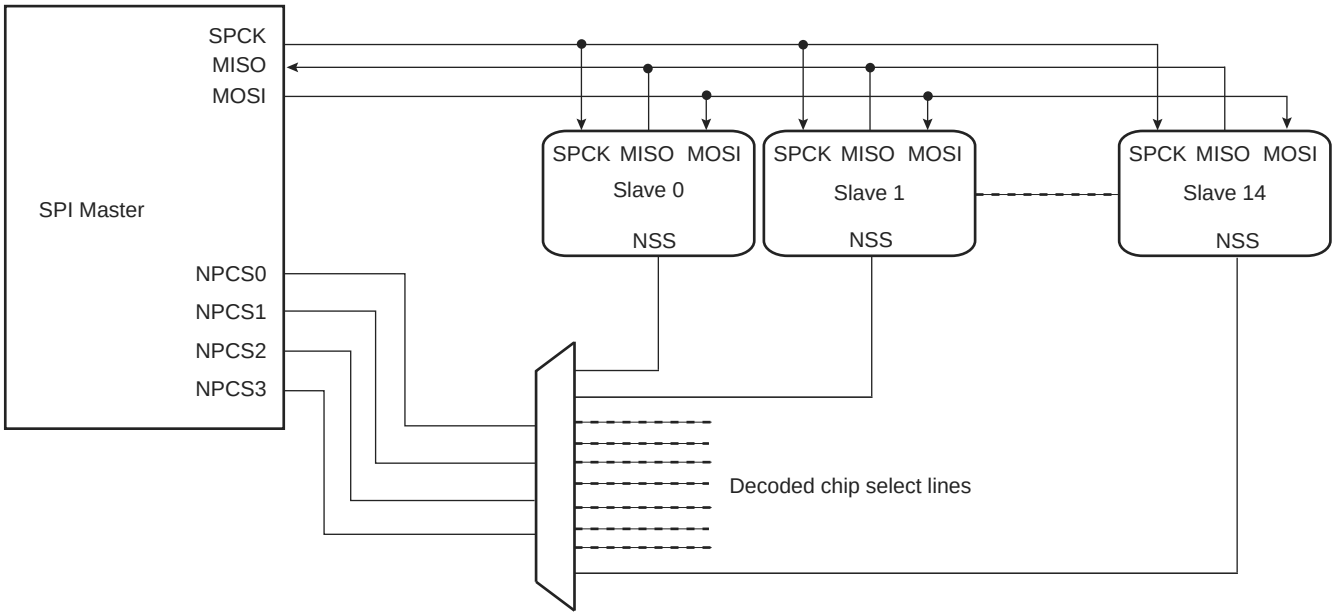
- **EPROC: Embedded Processor**

Value	Name	Description
0	SAM x7	Cortex-M7
1	ARM946ES	ARM946ES
2	ARM7TDMI	ARM7TDMI
3	CM3	Cortex-M3
4	ARM920T	ARM920T
5	ARM926EJS	ARM926EJS
6	CA5	Cortex-A5
7	CM4	Cortex-M4

- **NVPSIZ: Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6	–	Reserved
7	128K	128 Kbytes
8	160K	160 Kbytes
9	256K	256 Kbytes
10	512K	512 Kbytes

Figure 33-11. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation



33.7.3.8 Peripheral Deselection without PDC

During a transfer of more than one unit of data on a Chip Select without the PDC, the SPI_TDR is loaded by the processor, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal Shift register. When this flag is detected high, the SPI_TDR can be reloaded. If this reload by the processor occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. But depending on the application software handling the SPI status register flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the SPI_TDR in time to keep the chip select active (low). A null DLYBCT value (delay between consecutive transfers) in the SPI_CSR, gives even less time for the processor to reload the SPI_TDR. With some SPI slave peripherals, if the chip select line must remain active (low) during a full set of transfers, communication errors can occur.

To facilitate interfacing with such devices, the Chip Select registers [CSR0...CSR3] can be programmed with the Chip Select Active After Transfer (CSAAT) bit to 1. This allows the chip select lines to remain in their current state (low = active) until a transfer to another chip select is required. Even if the SPI_TDR is not reloaded, the chip select remains active. To de-assert the chip select line at the end of the transfer, the Last Transfer (LASTXFER) bit in SPI_CR must be set after writing the last data to transmit into SPI_TDR.

33.7.3.9 Peripheral Deselection with PDC

PDC provides faster reloads of the SPI_TDR compared to software. However, depending on the system activity, it is not guaranteed that the SPI_TDR is written with the next data before the end of the current transfer. Consequently, data can be lost by the de-assertion of the NPCS line for SPI slave peripherals requiring the chip select line to remain active between two transfers. The only way to guarantee a safe transfer in this case is the use of the CSAAT and LASTXFER bits.

When the CSAAT bit is configured to 0, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a Chip Select, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal shift register. When this flag is detected, the SPI_TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. This can lead to difficulties to interface with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate

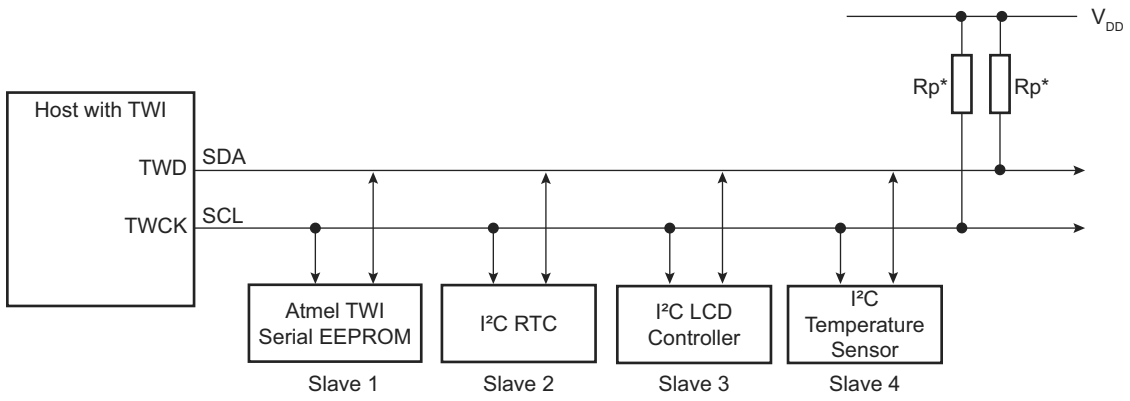
34.7.3 Master Mode

34.7.3.1 Definition

The master is the device that starts a transfer, generates a clock and stops it.

34.7.3.2 Application Block Diagram

Figure 34-4. Master Mode Typical Application Block Diagram



* Rp: Pull-up value as given by the I²C Standard

34.7.3.3 Programming Master Mode

The following fields must be programmed before entering Master mode:

1. TWI_MMR.DADR (+ IADRSZ + IADR if a 10-bit device is addressed): The device address is used to access slave devices in Read or Write mode.
2. TWI_CWGR.CKDIV + CHDIV + CLDIV: Clock waveform.
3. TWI_CR.SVDIS: Disables the Slave mode
4. TWI_CR.MSEN: Enables the Master mode

Note: If the TWI is already in Master mode, the device address (DADR) can be configured without disabling the Master mode.

34.7.3.4 Master Transmitter Mode

After the master initiates a START condition when writing into the Transmit Holding register (TWI_THR), it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction—0 in this case (MREAD = 0 in TWI_MMR).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. If the slave does not acknowledge the byte, then the Not Acknowledge flag (NACK) is set in the TWI Status Register (TWI_SR) of the master and a STOP condition is sent. The NACK flag must be cleared by reading the TWI Status Register (TWI_SR) before the next write into the TWI Transmit Holding Register(TWI_THR). As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable register (TWI_IER). If the slave acknowledges the byte, the data written in the TWI_THR is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR.

TXRDY is used as Transmit Ready for the PDC transmit channel.

While no new data is written in the TWI_THR, the serial clock line (SCL) is tied low. When new data is written in the TWI_THR, the TWCK/SCL is released and the data is sent. Setting the STOP bit in TWI_CR generates a STOP condition.

34.8.10 TWI Receive Holding Register

Name: TWI_RHR

Address: 0x40018030 (0), 0x4001C030 (1)

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Master or Slave Receive Holding Data**

When the start frame delimiter is a sync pattern (ONEBIT field to 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the US_RHR and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

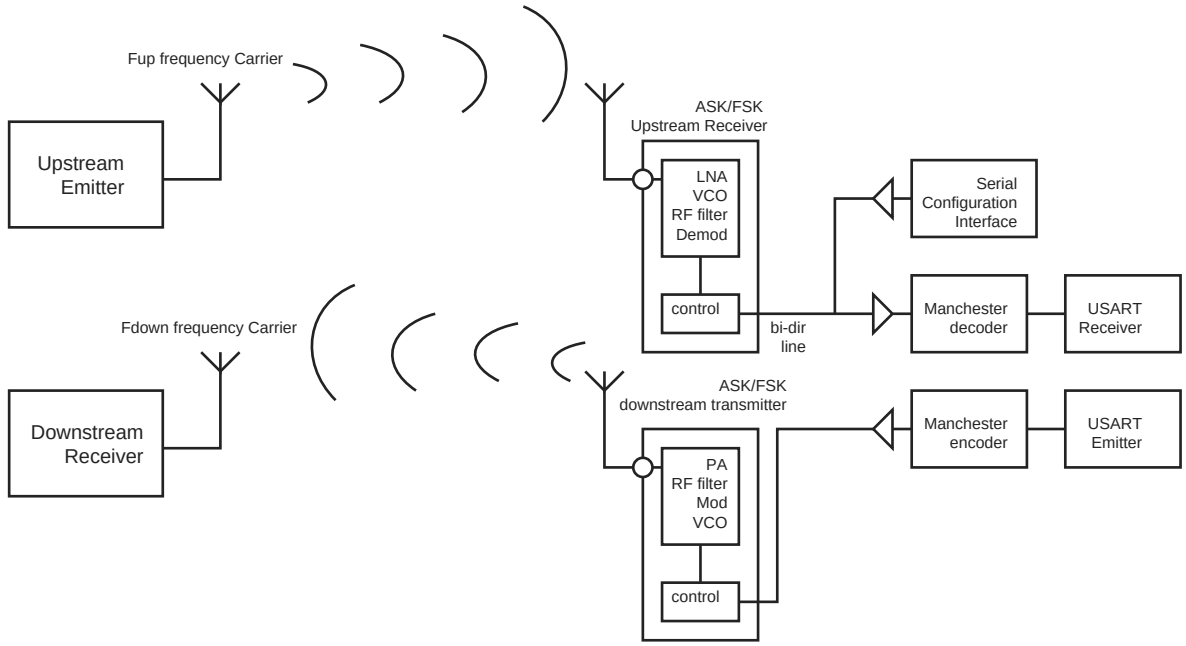
As the decoder is setup to be used in Unipolar mode, the first bit of the frame has to be a zero-to-one transition.

36.6.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 36-16.

Figure 36-16. Manchester Encoded Characters RF Transmission



The USART peripheral is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 36-17 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 36-18.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to Receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

36.7.13 USART Receive Holding Register

Name: US_RHR

Address: 0x40024018 (0), 0x40028018 (1)

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
RXSYNH	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last character received is a data.

1: Last character received is a command.

37.7.8 TC Register C

Name: TC_RCx [x=0..2]

Address: 0x4001001C (0)[0], 0x4001005C (0)[1], 0x4001009C (0)[2], 0x4001401C (1)[0], 0x4001405C (1)[1], 0x4001409C (1)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
RC							

23	22	21	20	19	18	17	16
RC							

15	14	13	12	11	10	9	8
RC							

7	6	5	4	3	2	1	0
RC							

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

- **RC: Register C**

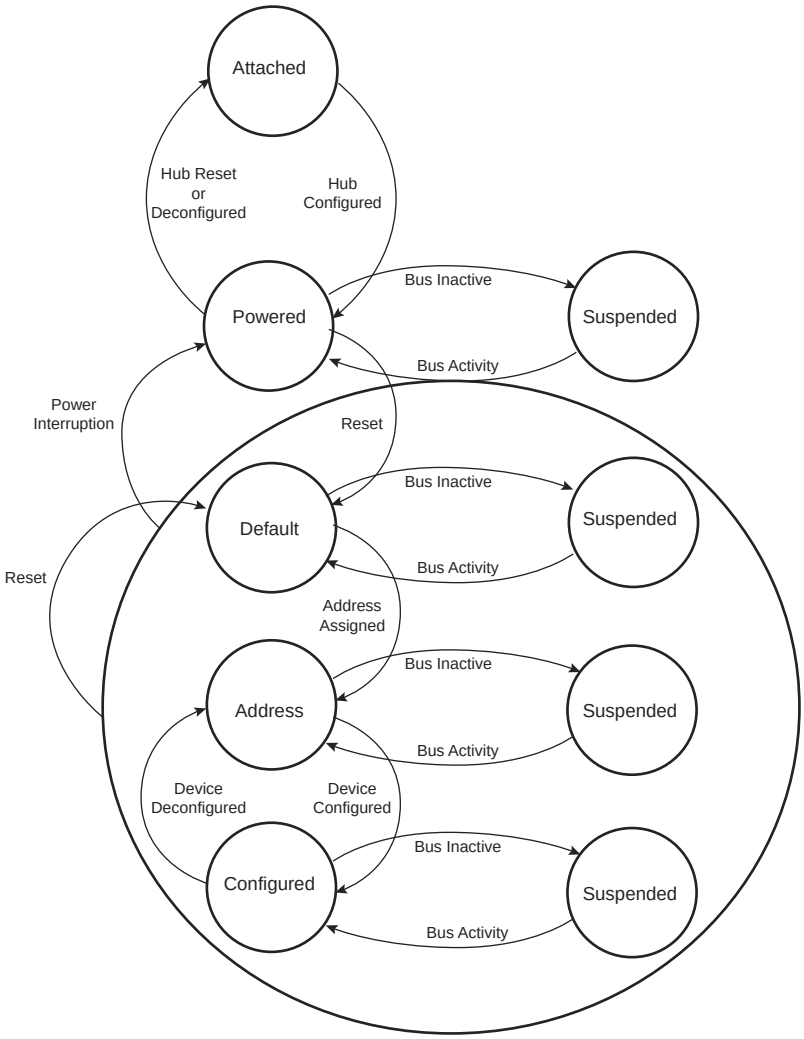
RC contains the Register C value in real time.

IMPORTANT: For 16-bit channels, RC field size is limited to register bits 15:0.

40.6.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 40-14. USB Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices must not consume more than 2.5 mA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wakeup request to the host, e.g., waking up a PC by moving a USB mouse.

The wakeup feature is not mandatory for all devices and must be negotiated with the host.

40.6.3.1 Not Powered State

Self powered devices can detect 5V VBUS using a PIO as described in the typical connection section. When the device is not connected to a host, device power consumption can be reduced by disabling MCK for the UDP, disabling UDPCK and disabling the transceiver. DDP and DDM lines are pulled down by 330 KΩ resistors.

40.7.6 UDP Interrupt Mask Register

Name: UDP_IMR

Address: 0x40034018

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
–	–	WAKEUP	BIT12	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
EP7INT	EP6INT	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Mask Endpoint 0 Interrupt**
- **EP1INT: Mask Endpoint 1 Interrupt**
- **EP2INT: Mask Endpoint 2 Interrupt**
- **EP3INT: Mask Endpoint 3 Interrupt**
- **EP4INT: Mask Endpoint 4 Interrupt**
- **EP5INT: Mask Endpoint 5 Interrupt**
- **EP6INT: Mask Endpoint 6 Interrupt**
- **EP7INT: Mask Endpoint 7 Interrupt**

0: Corresponding Endpoint Interrupt is disabled

1: Corresponding Endpoint Interrupt is enabled

- **RXSUSP: Mask UDP Suspend Interrupt**

0: UDP Suspend Interrupt is disabled

1: UDP Suspend Interrupt is enabled

- **RXRSM: Mask UDP Resume Interrupt.**

0: UDP Resume Interrupt is disabled

1: UDP Resume Interrupt is enabled

- **SOFINT: Mask Start Of Frame Interrupt**

0: Start of Frame Interrupt is disabled

1: Start of Frame Interrupt is enabled

- **BIT12: UDP_IMR Bit 12**

Bit 12 of UDP_IMR cannot be masked and is always read at 1.

Table 41-1. List of External Analog Data Inputs

Pin Name	Description
AD0..AD7	ACC Analog PLUS inputs
AD0..AD3	ACC Analog MINUS inputs
ADVREF	ADCVoltage reference

41.4 Pin Name List

Table 41-2. ACC Pin List

Pin Name	Description	Type
AD0..AD7	External analog data inputs	Input
TS	On-chip temperature sensor	Input
ADVREF	ADC voltage reference	Input
DAC0, DAC1	On-chip DAC inputs	Input

41.5 Product Dependencies

41.5.1 I/O Lines

The analog input pins (AD0–AD7 and DAC0–1) are multiplexed with digital functions (PIO) on the IO line. By writing the SELMINUS and SELPLUS fields in the ACC Mode Register (ACC_MR), the associated IO lines are set to Analog mode.

41.5.2 Power Management

The ACC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the ACC clock.

Note that the voltage regulator must be activated to use the analog comparator.

41.5.3 Interrupt

The ACC has an interrupt line connected to the Interrupt Controller (IC). In order to handle interrupts, the Interrupt Controller must be programmed before configuring the ACC.

Table 41-3. Peripheral IDs

Instance	ID
ACC	33

41.5.4 Fault Output

The ACC has the FAULT output connected to the FAULT input of PWM. Please refer to chapter Section 41.6.4 ”Fault Mode” and the implementation of the PWM in the product.

42.7.17 ADC Channel Offset Register

Name: ADC_COR

Address: 0x4003804C

Access: Read/Write

31	30	29	28	27	26	25	24
DIFF15	DIFF14	DIFF13	DIFF12	DIFF11	DIFF10	DIFF9	DIFF8
23	22	21	20	19	18	17	16
DIFF7	DIFF6	DIFF5	DIFF4	DIFF3	DIFF2	DIFF1	DIFF0
15	14	13	12	11	10	9	8
OFF15	OFF14	OFF13	OFF12	OFF11	OFF10	OFF9	OFF8
7	6	5	4	3	2	1	0
OFF7	OFF6	OFF5	OFF4	OFF3	OFF2	OFF1	OFF0

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

- **OFFx: Offset for Channel x**

0: No offset.

1: Centers the analog signal on $V_{ADVREF}/2$ before the gain scaling. The offset applied is

$(G-1)V_{ADVREF}/2$

where G is the gain applied (see Section 42.7.16 “ADC Channel Gain Register”).

- **DIFFx: Differential Inputs for Channel x**

0: Single-ended mode.

1: Fully differential mode.