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	CANbus, Ethernet, IrDA, MMC/SD, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	79
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K 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-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4e8cb-an

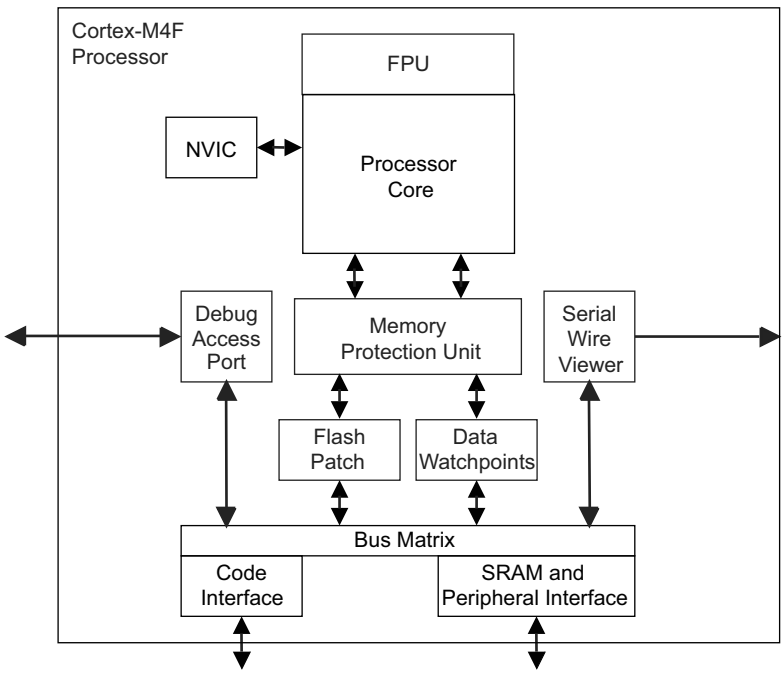
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored on a non-erasable, ROM-based microcontroller to be patched if a small programmable memory, for example flash, is available in the device. During initialization, the application in ROM detects, from the programmable memory, whether a patch is required. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration, which means the program in the non-modifiable ROM can be patched.

11.2 Embedded Characteristics

- Tight integration of system peripherals reduces area and development costs
- Thumb instruction set combines high code density with 32-bit performance
- IEEE754-compliant single-precision FPU
- Code-patch ability for ROM system updates
- Power control optimization of system components
- Integrated sleep modes for low power consumption
- Fast code execution permits slower processor clock or increases sleep mode time
- Hardware division and fast digital-signal-processing oriented multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory Protection Unit (MPU) for safety-critical applications
- Extensive debug and trace capabilities:
  - Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging, tracing, and code profiling.

11.3 Block Diagram

Figure 11-1. Typical Cortex-M4F Implementation



11.4.1.13 Priority Mask Register

Name: PRIMASK

Access: Read/Write

Reset: 0x000000000

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
–							PRIMASK

The PRIMASK register prevents the activation of all exceptions with a configurable priority.

- **PRIMASK**

0: No effect

1: Prevents the activation of all exceptions with a configurable priority.

11.6.7.3 QADD and QSUB

Saturating Add and Saturating Subtract, signed.

Syntax

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

where:

op is one of:

QADD Saturating 32-bit add.

QADD8 Saturating four 8-bit integer additions.

QADD16 Saturating two 16-bit integer additions.

QSUB Saturating 32-bit subtraction.

QSUB8 Saturating four 8-bit integer subtraction.

QSUB16 Saturating two 16-bit integer subtraction.

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

These instructions add or subtract two, four or eight values from the first and second operands and then writes a signed saturated value in the destination register.

The QADD and QSUB instructions apply the specified add or subtract, and then saturate the result to the signed range $-2^{n-1} \le x \le 2^{n-1}-1$, where x is given by the number of bits applied in the instruction, 32, 16 or 8.

If the returned result is different from the value to be saturated, it is called *saturation*. If saturation occurs, the QADD and QSUB instructions set the Q flag to 1 in the APSR. Otherwise, it leaves the Q flag unchanged. The 8-bit and 16-bit QADD and QSUB instructions always leave the Q flag unchanged.

To clear the Q flag to 0, the MSR instruction must be used; see “MSR” .

To read the state of the Q flag, the MRS instruction must be used; see “MRS” .

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not affect the condition code flags.

If saturation occurs, these instructions set the Q flag to 1.

Examples

```
QADD16   R7, R4, R2  ; Adds halfwords of R4 with corresponding halfword of
                     ; R2, saturates to 16 bits and writes to
                     ; corresponding halfword of R7
QADD8    R3, R1, R6  ; Adds bytes of R1 to the corresponding bytes of R6,
                     ; saturates to 8 bits and writes to corresponding
                     ; byte of R3
QSUB16   R4, R2, R3  ; Subtracts halfwords of R3 from corresponding
                     ; halfword of R2, saturates to 16 bits, writes to
                     ; corresponding halfword of R4
QSUB8    R4, R2, R5  ; Subtracts bytes of R5 from the corresponding byte
                     ; in R2, saturates to 8 bits, writes to corresponding
                     ; byte of R4.
```

11.8.3.2 Interrupt Clear-enable Registers

Name: NVIC_ICERx [x=0..7]

Access: Read/Write

Reset: 0x000000000

31	30	29	28	27	26	25	24
CLRENA							

23	22	21	20	19	18	17	16
CLRENA							

15	14	13	12	11	10	9	8
CLRENA							

7	6	5	4	3	2	1	0
CLRENA							

These registers disable interrupts, and show which interrupts are enabled.

- **CLRENA: Interrupt Clear-enable**

Write:

0: No effect.

1: Disables the interrupt.

Read:

0: Interrupt disabled.

1: Interrupt enabled.

11.11.2 Memory Protection Unit (MPU) User Interface

Table 11-41. Memory Protection Unit (MPU) Register Mapping

Offset	Register	Name	Access	Reset
0xE000ED90	MPU Type Register	MPU_TYPE	Read-only	0x00000800
0xE000ED94	MPU Control Register	MPU_CTRL	Read/Write	0x00000000
0xE000ED98	MPU Region Number Register	MPU_RNR	Read/Write	0x00000000
0xE000ED9C	MPU Region Base Address Register	MPU_RBAR	Read/Write	0x00000000
0xE000EDA0	MPU Region Attribute and Size Register	MPU_RASR	Read/Write	0x00000000
0xE000EDA4	MPU Region Base Address Register Alias 1	MPU_RBAR_A1	Read/Write	0x00000000
0xE000EDA8	MPU Region Attribute and Size Register Alias 1	MPU_RASR_A1	Read/Write	0x00000000
0xE000EDAC	MPU Region Base Address Register Alias 2	MPU_RBAR_A2	Read/Write	0x00000000
0xE000EDB0	MPU Region Attribute and Size Register Alias 2	MPU_RASR_A2	Read/Write	0x00000000
0xE000EDB4	MPU Region Base Address Register Alias 3	MPU_RBAR_A3	Read/Write	0x00000000
0xE000EDB8	MPU Region Attribute and Size Register Alias 3	MPU_RASR_A3	Read/Write	0x00000000

25.6 Functional Description

25.6.1 Basic Definitions

Source peripheral: Device on an AMBA layer from where the DMAC reads data, which is then stored in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel.

Destination peripheral: Device to which the DMAC writes the stored data from the FIFO (previously read from the source peripheral).

Memory: Source or destination that is always “ready” for a DMAC transfer and does not require a handshaking interface to interact with the DMAC.

Programmable Arbitration Policy: Modified Round Robin and Fixed Priority are available by means of the ARB_CFG bit in the Global Configuration Register (DMAC_GCFG). The fixed priority is linked to the channel number. The highest DMAC channel number has the highest priority.

Channel: Read/write datapath between a source peripheral on one configured AMBA layer and a destination peripheral on the same or different AMBA layer that occurs through the channel FIFO. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.

Master interface: DMAC is a master on the AHB bus reading data from the source and writing it to the destination over the AHB bus.

Slave interface: The APB interface over which the DMAC is programmed. The slave interface in practice could be on the same layer as any of the master interfaces or on a separate layer.

Handshaking interface: A set of signal registers that conform to a protocol and *handshake* between the DMAC and source or destination peripheral to control the transfer of a single or chunk transfer between them. This interface is used to request, acknowledge, and control a DMAC transaction. A channel can receive a request through one of two types of handshaking interface: hardware or software.

Hardware handshaking interface: Uses hardware signals to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral.

Software handshaking interface: Uses software registers to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral. No special DMAC handshaking signals are needed on the I/O of the peripheral. This mode is useful for interfacing an existing peripheral to the DMAC without modifying it.

Transfer hierarchy: Figure 25-2 illustrates the hierarchy between DMAC transfers, buffer transfers, chunk or single, and AMBA transfers (single or burst) for non-memory peripherals. Figure 25-3 shows the transfer hierarchy for memory.

- ENDTX flag is set when the PDC Transmit Counter Register (PERIPH_TCR) reaches zero.
- TXBUFE flag is set when both PERIPH_TCR and the PDC Transmit Next Counter Register (PERIPH_TNCR) reach zero.

These status flags are described in the Transfer Status Register (PERIPH_PTSR).

26.4.4 Data Transfers

The serial peripheral triggers its associated PDC channels' transfers using transmit enable (TXEN) and receive enable (RXEN) flags in the transfer control register integrated in the peripheral's user interface.

When the peripheral receives external data, it sends a Receive Ready signal to its PDC receive channel which then requests access to the Matrix. When access is granted, the PDC receive channel starts reading the peripheral Receive Holding register (RHR). The read data are stored in an internal buffer and then written to memory.

When the peripheral is about to send data, it sends a Transmit Ready to its PDC transmit channel which then requests access to the Matrix. When access is granted, the PDC transmit channel reads data from memory and transfers the data to the Transmit Holding register (THR) of its associated peripheral. The same peripheral sends data depending on its mechanism.

26.4.5 PDC Flags and Peripheral Status Register

Each peripheral connected to the PDC sends out receive ready and transmit ready flags and the PDC returns flags to the peripheral. All these flags are only visible in the peripheral's Status register.

Depending on whether the peripheral is half- or full-duplex, the flags belong to either one single channel or two different channels.

26.4.5.1 Receive Transfer End

The receive transfer end flag is set when PERIPH_RCR reaches zero and the last data has been transferred to memory.

This flag is reset by writing a non-zero value to PERIPH_RCR or PERIPH_RNCR.

26.4.5.2 Transmit Transfer End

The transmit transfer end flag is set when PERIPH_TCR reaches zero and the last data has been written to the peripheral THR.

This flag is reset by writing a non-zero value to PERIPH_TCR or PERIPH_TNCR.

26.4.5.3 Receive Buffer Full

The receive buffer full flag is set when PERIPH_RCR reaches zero, with PERIPH_RNCR also set to zero and the last data transferred to memory.

This flag is reset by writing a non-zero value to PERIPH_TCR or PERIPH_TNCR.

26.4.5.4 Transmit Buffer Empty

The transmit buffer empty flag is set when PERIPH_TCR reaches zero, with PERIPH_TNCR also set to zero and the last data written to peripheral THR.

This flag is reset by writing a non-zero value to PERIPH_TCR or PERIPH_TNCR.

27.9 Standard Read and Write Protocols

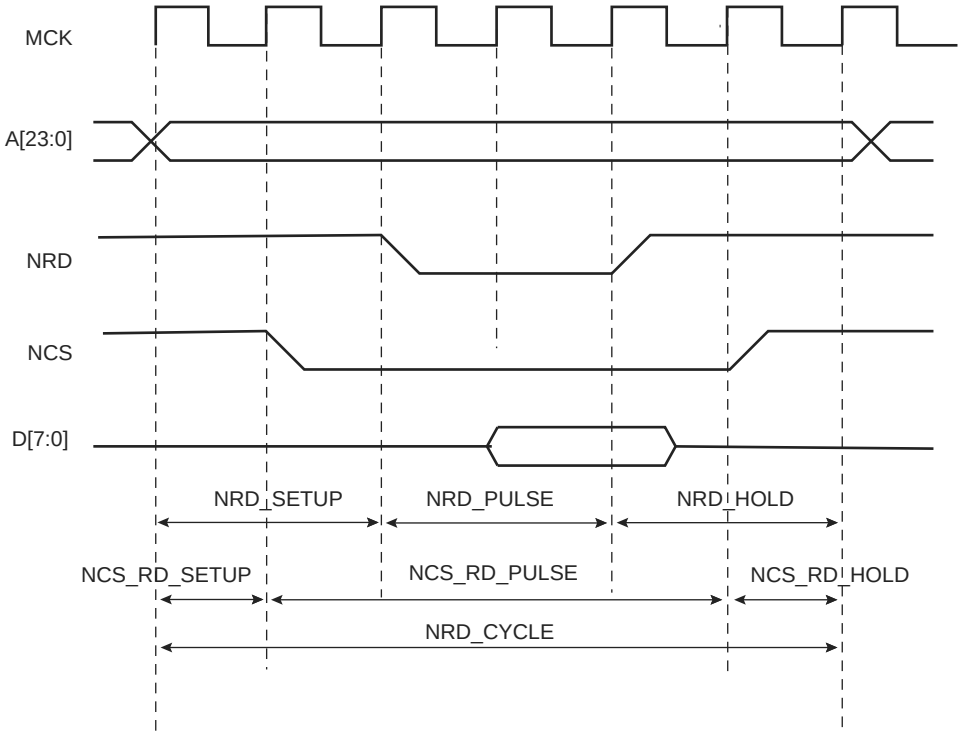
In the following sections, NCS represents one of the NCS[0..3] chip select lines.

27.9.1 Read Waveforms

The read cycle is shown in Figure 27-5.

The read cycle starts with the address setting on the memory address bus.

Figure 27-5. Standard Read Cycle



27.9.1.1 NRD Waveform

The NRD signal is characterized by a setup timing, a pulse width and a hold timing.

- NRD_SETUP— NRD setup time is defined as the setup of address before the NRD falling edge;
- NRD_PULSE—NRD pulse length is the time between NRD falling edge and NRD rising edge;
- NRD_HOLD—NRD hold time is defined as the hold time of address after the NRD rising edge.

27.9.1.2 NCS Waveform

The NCS signal can be divided into a setup time, pulse length and hold time:

- NCS_RD_SETUP—NCS setup time is defined as the setup time of address before the NCS falling edge.
- NCS_RD_PULSE—NCS pulse length is the time between NCS falling edge and NCS rising edge;
- NCS_RD_HOLD—NCS hold time is defined as the hold time of address after the NCS rising edge.

27.9.1.3 Read Cycle

The NRD_CYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is defined as:

NRD_CYCLE = NRD_SETUP + NRD_PULSE + NRD_HOLD,

as well as

NRD_CYCLE = NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

The PLLxCOUNT field specifies the number of slow clock cycles before the LOCKx bit is set in the PMC_SR after CKGR_PLLxR has been written.

Once CKGR_PLLxR has been written, the user must wait for the LOCKx bit to be set in the PMC_SR. This can be done either by polling LOCKx in PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (LOCKx) has been enabled in PMC_IER. All fields in CKGR_PLLxR can be programmed in a single write operation. If at some stage one of the following parameters, MULx or DIVx is modified, the LOCKx bit goes low to indicate that PLLx is not yet ready. When PLLx is locked, LOCKx is set again. The user must wait for the LOCKx bit to be set before using the PLLx output clock.

7. Select the master clock and processor clock

   The master clock and the processor clock are configurable via PMC_MCKR.

   The CSS field is used to select the clock source of the master clock and processor clock dividers. By default, the selected clock source is the main clock.

   The PRES field is used to define the processor clock and master clock prescaler. The user can choose between different values (1, 2, 3, 4, 8, 16, 32, 64). Prescaler output is the selected clock source frequency divided by the PRES value.

   Once the PMC_MCKR has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR. This can be done either by polling MCKRDY in PMC_SR or by waiting for the interrupt line to be raised if the associated interrupt source (MCKRDY) has been enabled in PMC_IER. PMC_MCKR must not be programmed in a single write operation. The programming sequence for PMC_MCKR is as follows:

- If a new value for CSS field corresponds to PLL clock,
  - Program the PRES field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.
  - Program the CSS field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.
- If a new value for CSS field corresponds to main clock or slow clock,
  - Program the CSS field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the PRES field in PMC_MCKR.
  - Wait for the MCKRDY bit to be set in PMC_SR.

If at some stage, parameters CSS or PRES are modified, the MCKRDY bit goes low to indicate that the master clock and the processor clock are not yet ready. The user must wait for MCKRDY bit to be set again before using the master and processor clocks.

Note: IF PLLx clock was selected as the master clock and the user decides to modify it by writing in CKGR_PLLxR, the MCKRDY flag will go low while PLLx is unlocked. Once PLLx is locked again, LOCKx goes high and MCKRDY is set. While PLLx is unlocked, the master clock selection is automatically changed to slow clock for PLLA. For further information, see Section 29.16.2 ”Clock Switching Waveforms”.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The master clock is main clock divided by 2.

8. Select the programmable clocks

   Programmable clocks are controlled via registers, PMC_SCER, PMC_SCDR and PMC_SCSR.

29.16.2 Clock Switching Waveforms

Figure 29-6. Switch Master Clock from Slow Clock to PLLx Clock

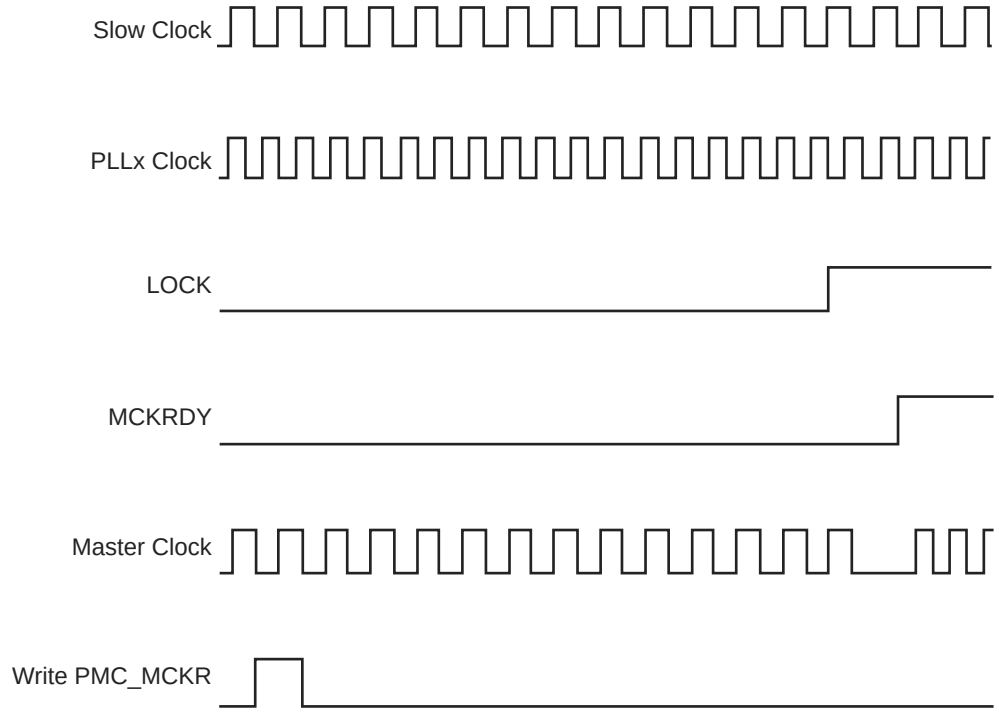
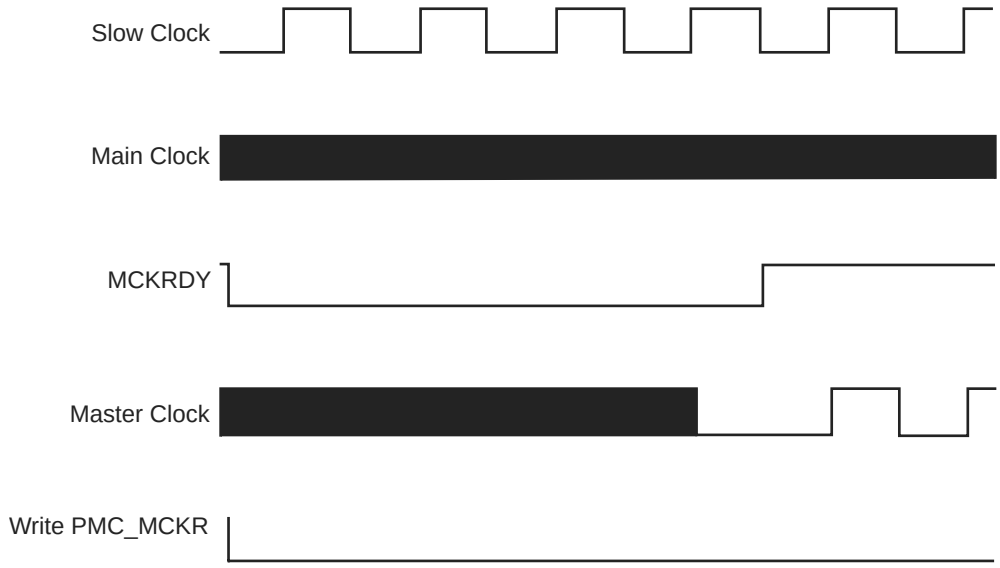


Figure 29-7. Switch Master Clock from Main Clock to Slow Clock



32. Chip Identifier (CHIPID)

32.1 Description

Chip Identifier (CHIPID) registers are used to recognize the device and its revision. These registers provide the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Two CHIPID registers are embedded: Chip ID Register (CHIPID_CIDR) and Chip ID Extension Register (CHIPID_EXID). Both registers contain a hard-wired value that is read-only.

The CHIPID_CIDR register contains the following fields:

- VERSION: Identifies the revision of the silicon
- EPROC: Indicates the embedded ARM processor
- NVPTYP and NVPSIZ: Identify the type of embedded non-volatile memory and the size
- SRAMSIZ: Indicates the size of the embedded SRAM
- ARCH: Identifies the set of embedded peripherals
- EXT: Shows the use of the extension identifier register

The CHIPID_EXID register is device-dependent and reads 0 if CHIPID_CIDR.EXT = 0.

32.2 Embedded Characteristics

- Chip ID Registers
  - Identification of the Device Revision, Sizes of the Embedded Memories, Set of Peripherals, Embedded Processor

Table 32-1. SAM4E Chip ID Registers

Chip Name	CHIPID_CIDR	CHIPID_EXID
SAM4E16E	0xA3CC_0CE0	0x0012_0200
SAM4E8E	0xA3CC_0CE0	0x0012_0208
SAM4E16C	0xA3CC_0CE0	0x0012_0201
SAM4E8C	0xA3CC_0CE0	0x0012_0209

32.3 Chip Identifier (CHIPID) User Interface

Table 32-2. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Chip ID Register	CHIPID_CIDR	Read-only	–
0x4	Chip ID Extension Register	CHIPID_EXID	Read-only	–

34.8.10 SPI Write Protection Mode Register

Name: SPI_WPMR

Address: 0x400880E4

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 Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

1: Enables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

See Section 34.7.5 “Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x535049	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

38.7.14 TC Extended Mode Register

Name: TC_EMRx [x=0..2]

Address: 0x40090030 (0)[0], 0x40090070 (0)[1], 0x400900B0 (0)[2], 0x40094030 (1)[0], 0x40094070 (1)[1], 0x400940B0 (1)[2], 0x40098030 (2)[0], 0x40098070 (2)[1], 0x400980B0 (2)[2]

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
–	–	–	–	–	–	–	NODIVCLK
7	6	5	4	3	2	1	0
–	–	TRIGSRCB		–	–	TRIGSRCA	

- **TRIGSRCA: Trigger Source for Input A**

Value	Name	Description
0	EXTERNAL_TIOAx	The trigger/capture input A is driven by external pin TIOAx
1	PWMx	The trigger/capture input A is driven internally by PWMx

- **TRIGSRCB: Trigger Source for Input B**

Value	Name	Description
0	EXTERNAL_TIOBx	The trigger/capture input B is driven by external pin TIOBx
1	PWMx	The trigger/capture input B is driven internally by the comparator output (see Figure 38-16) of the PWMx.

- **NODIVCLK: No Divided Clock**

0: The selected clock is defined by field TCCLKS in TC_CMRx.

1: The selected clock is peripheral clock and TCCLKS field (TC_CMRx) has no effect.

- **CES: Counter Event Selection**

The bit CES defines when the channel counter event occurs when the period is center-aligned (flag CHIDx in PWM Interrupt Status Register 1).

CALG = 0 (Left Alignment):

0/1: The channel counter event occurs at the end of the PWM period.

CALG = 1 (Center Alignment):

0: The channel counter event occurs at the end of the PWM period.

1: The channel counter event occurs at the end of the PWM period and at half the PWM period.

- **UPDS: Update Selection**

When the period is center aligned, the bit UPDS defines when the update of the duty cycle, the polarity value/mode occurs after writing the corresponding update registers.

CALG = 0 (Left Alignment):

0/1: The update always occurs at the end of the PWM period after writing the update register(s).

CALG = 1 (Center Alignment):

0: The update occurs at the next end of the PWM period after writing the update register(s).

1: The update occurs at the next end of the PWM half period after writing the update register(s).

- **TCTS: Timer Counter Trigger Selection**

0: The comparator of the channel x (OCx) is used as the trigger source for the Timer Counter (TC).

1: The counter events of the channel x is used as the trigger source for the Timer Counter (TC).

- **DTE: Dead-Time Generator Enable**

0: The dead-time generator is disabled.

1: The dead-time generator is enabled.

- **DTHI: Dead-Time PWMHx Output Inverted**

0: The dead-time PWMHx output is not inverted.

1: The dead-time PWMHx output is inverted.

- **DTLI: Dead-Time PWMLx Output Inverted**

0: The dead-time PWMLx output is not inverted.

1: The dead-time PWMLx output is inverted.

41.7.2 UDP Global State Register

Name: UDP_GLB_STAT

Address: 0x40084004

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
–	–	–	RMWUPE	RSMINPR	ESR	CONFG	FADDEN

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

- **FADDEN: Function Address Enable**

Read:

0: Device is not in address state

1: Device is in address state

Write:

0: No effect, only a reset can bring back a device to the default state.

1: Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_FADDR register must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **CONFG: Configured**

Read:

0: Device is not in configured state

1: Device is in configured state

Write:

0: Sets device in a non configured state

1: Sets device in configured state

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **ESR: Enable Send Resume**

0: Mandatory value prior to starting any Remote Wakeup procedure

1: Starts the Remote Wakeup procedure if this bit value was 0 and if RMWUPE is enabled

- **RMWUPE: Remote Wakeup Enable**

0: The Remote Wakeup feature of the device is disabled.

1: The Remote Wakeup feature of the device is enabled.

- Encapsulation must be RFC 894 Ethernet Type Encoding or RFC 1042 SNAP Encoding.
- IPv4 packet
- IP header is of a valid length

The GMAC also checks the TCP checksum as per RFC 793, or the UDP checksum as per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- Good IP header checksum (if IPv4)
- No IP fragmentation
- TCP or UDP packet

When an IP, TCP or UDP frame is received, the receive buffer descriptor gives an indication if the GMAC was able to verify the checksums. There is also an indication if the frame had SNAP encapsulation. These indication bits will replace the type ID match indication bits when the receive checksum offload is enabled. For details of these indication bits refer to Table 42-4 “Receive Buffer Descriptor Entry”.

42.6.7 MAC Filtering Block

The filter block determines which frames should be written to the FIFO interface and on to the DMA.

Whether a frame is passed depends on what is enabled in the Network Configuration register, the state of the external matching pins, the contents of the specific address, type and Hash registers and the frame's destination address and type field.

If bit 25 of the Network Configuration register is not set, a frame will not be copied to memory if the GMAC is transmitting in half duplex mode at the time a destination address is received.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, which is the LSB of the first byte of the frame, is the group or individual bit. This is one for multicast addresses and zero for unicast. The all ones address is the broadcast address and a special case of multicast.

The GMAC supports recognition of four specific addresses. Each specific address requires two registers, Specific Address Bottom register and Specific Address Top register. Specific Address Bottom register stores the first four bytes of the destination address and Specific Address Top register contains the last two bytes. The addresses stored can be specific, group, local or universal.

The destination address of received frames is compared against the data stored in the Specific Address registers once they have been activated. The addresses are deactivated at reset or when their corresponding Specific Address Bottom register is written. They are activated when Specific Address Top register is written. If a receive frame address matches an active address, the frame is written to the FIFO interface and on to DMA memory.

Frames may be filtered using the type ID field for matching. Four type ID registers exist in the register address space and each can be enabled for matching by writing a one to the MSB (bit 31) of the respective register. When a frame is received, the matching is implemented as an OR function of the various types of match.

The contents of each type ID register (when enabled) are compared against the length/type ID of the frame being received (e.g., bytes 13 and 14 in non-VLAN and non-SNAP encapsulated frames) and copied to memory if a match is found. The encoded type ID match bits (Word 0, Bit 22 and Bit 23) in the receive buffer descriptor status are set indicating which type ID register generated the match, if the receive checksum offload is disabled.

The reset state of the type ID registers is zero, hence each is initially disabled.

- **SFR: PTP Sync Frame Received**
- **DRQFT: PTP Delay Request Frame Transmitted**
- **SFT: PTP Sync Frame Transmitted**
- **PDRQFR: PDelay Request Frame Received**
- **PDRSFR: PDelay Response Frame Received**
- **PDRQFT: PDelay Request Frame Transmitted**
- **PDRSFT: PDelay Response Frame Transmitted**
- **SRI: TSU Seconds Register Increment**
- **WOL: Wake On LAN**

43.7 Analog Front-End Controller (AFEC) User Interface

Table 43-7. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00	AFEC Control Register	AFEC_CR	Write-only	–
0x04	AFEC Mode Register	AFEC_MR	Read/Write	0x00000000
0x08	AFEC Extended Mode Register	AFEC_EMR	Read/Write	0x00000000
0x0C	AFEC Channel Sequence 1 Register	AFEC_SEQ1R	Read/Write	0x00000000
0x10	AFEC Channel Sequence 2 Register	AFEC_SEQ2R	Read/Write	0x00000000
0x14	AFEC Channel Enable Register	AFEC_CHER	Write-only	–
0x18	AFEC Channel Disable Register	AFEC_CHDR	Write-only	–
0x1C	AFEC Channel Status Register	AFEC_CHSR	Read-only	0x00000000
0x20	AFEC Last Converted Data Register	AFEC_LCDR	Read-only	0x00000000
0x24	AFEC Interrupt Enable Register	AFEC_IER	Write-only	–
0x28	AFEC Interrupt Disable Register	AFEC_IDR	Write-only	–
0x2C	AFEC Interrupt Mask Register	AFEC_IMR	Read-only	0x00000000
0x30	AFEC Interrupt Status Register	AFEC_ISR	Read-only	0x00000000
0x34–0x40	Reserved	–	–	–
0x44–0x48	Reserved	–	–	–
0x4C	AFEC Overrun Status Register	AFEC_OVER	Read-only	0x00000000
0x50	AFEC Compare Window Register	AFEC_CWR	Read/Write	0x00000000
0x54	AFEC Channel Gain Register	AFEC_CGR	Read/Write	0x00000000
0x5C	AFEC Channel Calibration DC Offset Register	AFEC_CDOR	Read/Write	0x00000000
0x60	AFEC Channel Differential Register	AFEC_DIFFR	Read/Write	0x00000000
0x64	AFEC Channel Selection Register	AFEC_CSELR	Read/Write	0x00000000
0x68	AFEC Channel Data Register	AFEC_CDR	Read-only	0x00000000
0x6C	AFEC Channel Offset Compensation Register	AFEC_COCR	Read/Write	0x00000000
0x70	AFEC Temperature Sensor Mode Register	AFEC_TEMPMR	Read/Write	0x00000000
0x74	AFEC Temperature Compare Window Register	AFEC_TEMPCWR	Read/Write	0x00000000
0x78–0x90	Reserved	–	–	–
0x94	AFEC Analog Control Register	AFEC_ACR	Read/Write	0x00000100
0x98–0xAC	Reserved	–	–	–
0xB0–0xE0	Reserved	–	–	–
0xE4	AFEC Write Protection Mode Register	AFEC_WPMR	Read/Write	0x00000000
0xE8	AFEC Write Protection Status Register	AFEC_WPSR	Read-only	0x00000000
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Notes: 1. Any offset not listed in Table 43-7 must be considered as “reserved”.

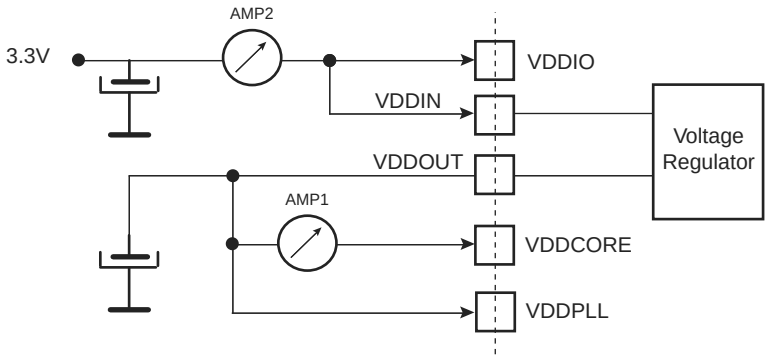
Table 46-9. Typical Power Consumption for Backup Mode Configuration A and B

BACKUP Total Consumption	Typical value				
	@25°C		@85°C	@105°C	
Conditions	(AMP1) Configuration A	(AMP1) Configuration B	(AMP1) Configuration A	(AMP1) Configuration A	Unit
V_{DDIO} = 3.6V	2.0	1.9	7.2	15.4	µA
V_{DDIO} = 3.3V	1.7	1.6	6.9	12.0	
V_{DDIO} = 3.0V	1.5	1.5	6.2	11.2	
V_{DDIO} = 2.5V	1.3	1.2	5.5	9.8	
V_{DDIO} = 1.8V	1	0.9	4.6	8.3	

46.3.2 Sleep and Wait Mode Current Consumption

The Wait mode and Sleep mode configuration and measurements are defined below.

Figure 46-5. Measurement Setup for Sleep Mode



46.3.2.1 Sleep Mode

- Core Clock OFF
- $V_{DDIO} = V_{DDIN} = 3.3V$
- Master Clock (MCK) running at various frequencies with PLLA or the fast RC oscillator
- Fast start-up through pins WKUP0–15
- Current measurement as shown in Figure 46-5
- All peripheral clocks deactivated
- $T_A = 25°C$

Table 46-10 gives current consumption in typical conditions.

Table 51-3. SAM4E Datasheet Rev. 11157F Revision History (Continued)

Doc. Date	Changes
27-Apr-15	Section 38. “Timer Counter (TC)” Modified Section 38.1 “Description”, Section 38.5.2 “Power Management” and Section 38.6.19 “Register Write Protection” Moved Table 38-1, “Timer Counter Clock Assignment” Modified ‘Name’ line in Section 38.7.2 “TC Channel Mode Register: Capture Mode” and Section 38.7.3 “TC Channel Mode Register: Waveform Mode” Modified Section 38.7.10 “TC Status Register”, Section 38.7.14 “TC Extended Mode Register”Section 38.7.16 “TC Block Mode Register”, Section 38.7.20 “TC QDEC Interrupt Status Register” Section 38.7.22 “TC Write Protection Mode Register” Modified Section 38.5.3 “Interrupt Sources” and Section 38.6.14 “Synchronization with PWM”, Section 38.6.16.4 “Position and Rotation Measurement”, Section 38.6.16.5 “Speed Measurement” Section 38.6.16 “Quadrature Decoder”: removed subsection “Missing Pulse Detection and Auto-correction”
	Section 39. “Pulse Width Modulation Controller (PWM)” Modified Section “Method 3: Automatic write of duty-cycle values and automatic trigger of the update” Updated Section 39.2 “Embedded Characteristics” throughout: corrected register name PWM_CPRx to PWM_CPRDx (PWM_CPRx does not exist) Section 39.5.3 “Interrupt Sources” Section 39.6 “Functional Description” Section 39.6.2.9 “Synchronous Channels” Section 39.7.24 “PWM Fault Mode Register”, Section 39.7.25 “PWM Fault Status Register”: changed field descriptions. Section 39.7.26 “PWM Fault Clear Register”, Section 39.7.28 “PWM Fault Protection Enable Register” Section 39.7.40 “PWM Channel Mode Register” Figure 39-17 “Comparison Unit Block Diagram” Table 39-7 “Register Mapping”: deleted reset value from PWM_SCUPUPD (this register is write-only) Added Figure 39-20 “Event Line Generation Waveform (Example)” Section 39.6.4 “PWM Event Lines”: in first sentence, replaced “i.e.” with “e.g.”
	Section 40. “High Speed Multimedia Card Interface (HSMCI)” Modified Section 40.1 “Description”: removed sentence “Only one slot can be selected at a time (slots are multiplexed)” Added Section 40.14.19 “HSMCI FIFOx Memory Aperture” Updated Section 40.14.12 “HSMCI Status Register” Updated Table 40-4, “Bus Topology” (4-bit instead of 8-bit)
	Section 41. “USB Device Port (UDP)” Table 41-6 Register Mapping: update footnote No. 1. Modified Section 41.7.4, “UDP Interrupt Enable Register” Section , “Using Endpoints With Ping-pong Attribute”: Replaced Bank 0 with Bank 1 in step 12.
	Section 42. “Ethernet MAC (GMAC)” Modified Section 42.2 “Embedded Characteristics”, Section 42.5.3 “Interrupt Sources” Modified Section 42.3 “Block Diagram” Added Section 42.5 “Product Dependencies”. Modified Section 42.6.3.1 “Receive AHB Buffers” and Section 42.6.3.2 “Transmit AHB Buffers”, Section 42.6.6.1 “Receiver Checksum Offload”, Section 42.6.15.2 “802.3 Pause Frame Transmission”, Section 42.6.16.2 “PFC Pause Frame Transmission”