Microchip Technology - ATSAM4E8CB-ANR Datasheet

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-anr

11.6.3 Instruction Descriptions

11.6.3.1 Operands

An instruction operand can be an ARM register, a constant, or another instruction-specific parameter. Instructions act on the operands and often store the result in a destination register. When there is a destination register in the instruction, it is usually specified before the operands.

Operands in some instructions are flexible, can either be a register or a constant. See "Flexible Second Operand" .

11.6.3.2 Restrictions when Using PC or SP

Many instructions have restrictions on whether the *Program Counter* (PC) or *Stack Pointer* (SP) for the operands or destination register can be used. See instruction descriptions for more information.

Note: Bit[0] of any address written to the PC with a BX, BLX, LDM, LDR, or POP instruction must be 1 for correct execution, because this bit indicates the required instruction set, and the Cortex-M4 processor only supports Thumb instructions.

11.6.3.3 Flexible Second Operand

Many general data processing instructions have a flexible second operand. This is shown as *Operand2* in the descriptions of the syntax of each instruction.

Operand2 can be a:

- "Constant"
- "Register with Optional Shift"

Constant

Specify an Operand2 constant in the form:

```
#constant
```

where *constant* can be:

- Any constant that can be produced by shifting an 8-bit value left by any number of bits within a 32-bit word
- Any constant of the form 0x00XY00XY
- Any constant of the form 0xXY00XY00
- Any constant of the form 0xXYXYXYXY.

Note: In the constants shown above, X and Y are hexadecimal digits.

In addition, in a small number of instructions, *constant* can take a wider range of values. These are described in the individual instruction descriptions.

When an Operand2 constant is used with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to bit[31] of the constant, if the constant is greater than 255 and can be produced by shifting an 8-bit value. These instructions do not affect the carry flag if *Operand2* is any other constant.

Instruction Substitution

The assembler might be able to produce an equivalent instruction in cases where the user specifies a constant that is not permitted. For example, an assembler might assemble the instruction CMP *Rd*, #0xFFFFFFFE as the equivalent instruction CMN *Rd*, #0x2.

Register with Optional Shift

Specify an Operand2 register in the form:

```
Rm {, shift}
```

where:

Rm	is the register holding the data for the second operand.
shift	is an optional shift to be applied to *Rm*. It can be one of:

11.6.6.10 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

In these instructions:

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

Examples

```
SMULBT       R0, R4, R5  ; Multiplies the bottom halfword of R4 with the
                         ; top halfword of R5, multiplies results and
                         ; writes to R0
SMULBB       R0, R4, R5  ; Multiplies the bottom halfword of R4 with the
                         ; bottom halfword of R5, multiplies results and
                         ; writes to R0
SMULTT       R0, R4, R5  ; Multiplies the top halfword of R4 with the top
                         ; halfword of R5, multiplies results and writes
                         ; to R0
SMULTB       R0, R4, R5  ; Multiplies the top halfword of R4 with the
```

11.6.12.7 MSR

Move the contents of a general-purpose register into the specified special register.

Syntax

```
MSR{cond} spec_reg, Rn
```

where:

cond	is an optional condition code, see “Conditional Execution” .
Rn	is the source register.
spec_reg	can be any of: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, or CONTROL.

Operation

The register access operation in MSR depends on the privilege level. Unprivileged software can only access the APSR. See “Application Program Status Register” . Privileged software can access all special registers.

In unprivileged software writes to unallocated or execution state bits in the PSR are ignored.

Note: When the user writes to BASEPRI_MAX, the instruction writes to BASEPRI only if either:
Rn is non-zero and the current BASEPRI value is 0
Rn is non-zero and less than the current BASEPRI value.

See “MRS” .

Restrictions

Rn must not be SP and must not be PC.

Condition Flags

This instruction updates the flags explicitly based on the value in *Rn*.

Examples

```
MSR  CONTROL, R1 ; Read R1 value and write it to the CONTROL register
```

11.10.1 System Timer (SysTick) User Interface

Table 11-35. System Timer (SYST) Register Mapping

Offset	Register	Name	Access	Reset
0xE000E010	SysTick Control and Status Register	SYST_CSR	Read/Write	0x00000000
0xE000E014	SysTick Reload Value Register	SYST_RVR	Read/Write	Unknown
0xE000E018	SysTick Current Value Register	SYST_CVR	Read/Write	Unknown
0xE000E01C	SysTick Calibration Value Register	SYST_CALIB	Read-only	0x00003A98

25.8.7 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Disable Register

Name: DMAC_EBCIDR

Address: 0x400C001C

Access: Write-only

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

23	22	21	20	19	18	17	16
–	–	–	–	ERR3	ERR2	ERR1	ERR0

15	14	13	12	11	10	9	8
–	–	–	–	CBTC3	CBTC2	CBTC1	CBTC0

7	6	5	4	3	2	1	0
–	–	–	–	BTC3	BTC2	BTC1	BTC0

- **BTCx: Buffer Transfer Completed [3:0]**

Buffer transfer completed Disable Interrupt Register. When set, a bit of the BTC field disables the interrupt from the relevant DMAC channel.

- **CBTCx: Chained Buffer Transfer Completed [3:0]**

Chained Buffer transfer completed Disable Register. When set, a bit of the CBTC field disables the interrupt from the relevant DMAC channel.

- **ERRx: Access Error [3:0]**

Access Error Interrupt Disable Register. When set, a bit of the ERR field disables the interrupt from the relevant DMAC channel.

30.5.3 AES Interrupt Enable Register

Name: AES_IER

Address: 0x40004010

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
–	–	–	–	–	–	–	URAD

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

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

0: No effect.

1: Enables the corresponding interrupt.

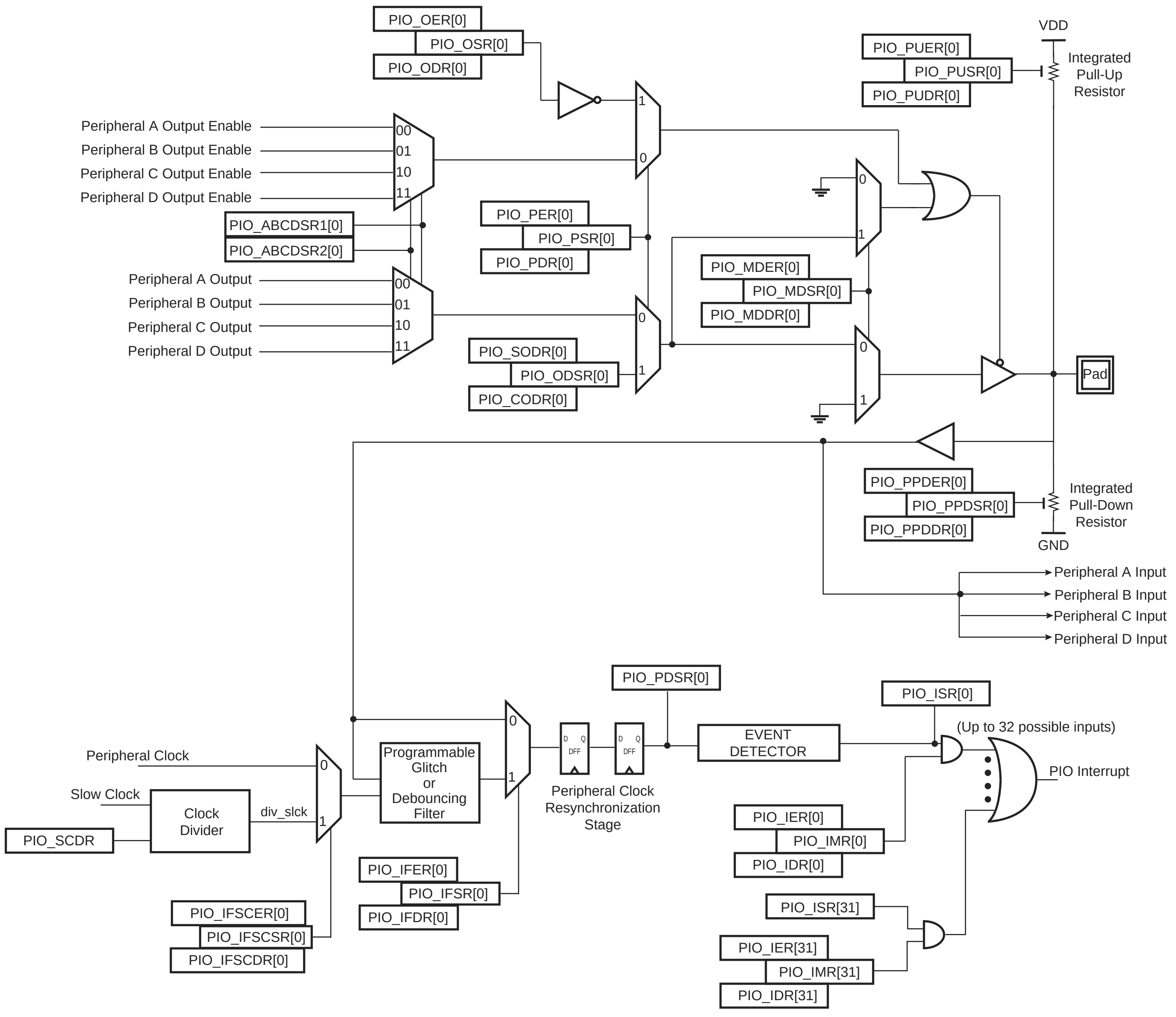
- **DATRDY: Data Ready Interrupt Enable**

- **URAD: Unspecified Register Access Detection Interrupt Enable**

33.5 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 33-2. In this description each signal shown represents one of up to 32 possible indexes.

Figure 33-2. I/O Line Control Logic



33.5.1 Pull-up and Pull-down Resistor Control

Each I/O line is designed with an embedded pull-up resistor and an embedded pull-down resistor. The pull-up resistor can be enabled or disabled by writing to the Pull-up Enable Register (PIO_PUER) or Pull-up Disable Register (PIO_PUDR), respectively. Writing to these registers results in setting or clearing the corresponding bit in the Pull-up Status Register (PIO_PUSR). Reading a one in PIO_PUSR means the pull-up is disabled and reading a zero means the pull-up is enabled. The pull-down resistor can be enabled or disabled by writing the Pull-down Enable Register (PIO_PPDER) or the Pull-down Disable Register (PIO_PPDDR), respectively. Writing in these

33.6.7 PIO Input Filter Enable Register

Name: PIO_IFER

Address: 0x400E0E20 (PIOA), 0x400E1020 (PIOB), 0x400E1220 (PIOC), 0x400E1420 (PIOD), 0x400E1620 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

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

- **P0–P31: Input Filter Enable**

0: No effect.

1: Enables the input glitch filter on the I/O line.

33.6.12 PIO Output Data Status Register

Name: PIO_ODSR

Address: 0x400E0E38 (PIOA), 0x400E1038 (PIOB), 0x400E1238 (PIOC), 0x400E1438 (PIOD), 0x400E1638 (PIOE)

Access: Read-only or Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Output Data Status**

0: The data to be driven on the I/O line is 0.

1: The data to be driven on the I/O line is 1.

34.8.6 SPI Interrupt Enable Register

Name: SPI_IER

Address: 0x40088014

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**
–	–	–	–	–	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: No effect.

1: Enables the corresponding interrupt.

- **RDRF: Receive Data Register Full Interrupt Enable**

- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**

- **MODF: Mode Fault Error Interrupt Enable**

- **OVRES: Overrun Error Interrupt Enable**

- **ENDRX: End of Receive Buffer Interrupt Enable**

- **ENDTX: End of Transmit Buffer Interrupt Enable**

- **RXBUFF: Receive Buffer Full Interrupt Enable**

- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

- **NSSR: NSS Rising Interrupt Enable**

- **TXEMPTY: Transmission Registers Empty Enable**

- **UNDES: Underrun Error Interrupt Enable**

- **ONEBIT: Start Frame Delimiter Selector**

0: Start frame delimiter is COMMAND or DATA SYNC.

1: Start frame delimiter is one bit.

38. Timer Counter (TC)

38.1 Description

A Timer Counter (TC) module includes three identical TC channels. The number of implemented TC modules is device-specific.

Each TC channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The TC embeds a quadrature decoder (QDEC) connected in front of the timers and driven by TIOA0, TIOB0 and TIOB1 inputs. When enabled, the QDEC performs the input lines filtering, decoding of quadrature signals and connects to the timers/counters in order to read the position and speed of the motor through the user interface.

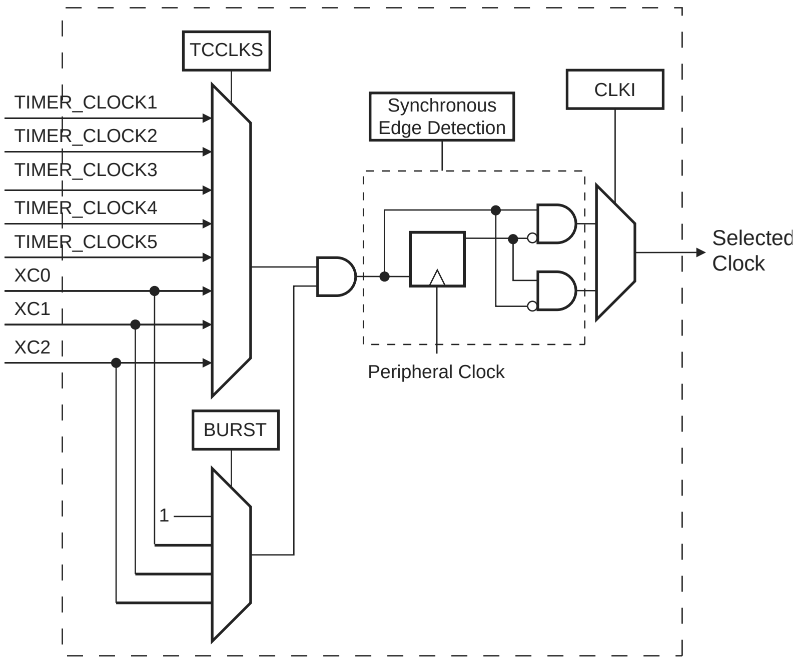
The TC block has two global registers which act upon all TC channels:

- Block Control Register (TC_BCR)—allows channels to be started simultaneously with the same instruction
- Block Mode Register (TC_BMR)—defines the external clock inputs for each channel, allowing them to be chained

38.2 Embedded Characteristics

- Total number of TC channels implemented on this device: nine
- TC channel size: 32-bit
- Wide range of functions including:
  - Frequency measurement
  - Event counting
  - Interval measurement
  - Pulse generation
  - Delay timing
  - Pulse Width Modulation
  - Up/down capabilities
  - Quadrature decoder
  - 2-bit Gray up/down count for stepper motor
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five Internal clock inputs
  - Two multi-purpose input/output signals acting as trigger event
  - Trigger/capture events can be directly synchronized by PWM signals
- Internal interrupt signal
- Read of the Capture registers by the PDC
- Compare event fault generation for PWM
- Register Write Protection

Figure 38-3. Clock Selection

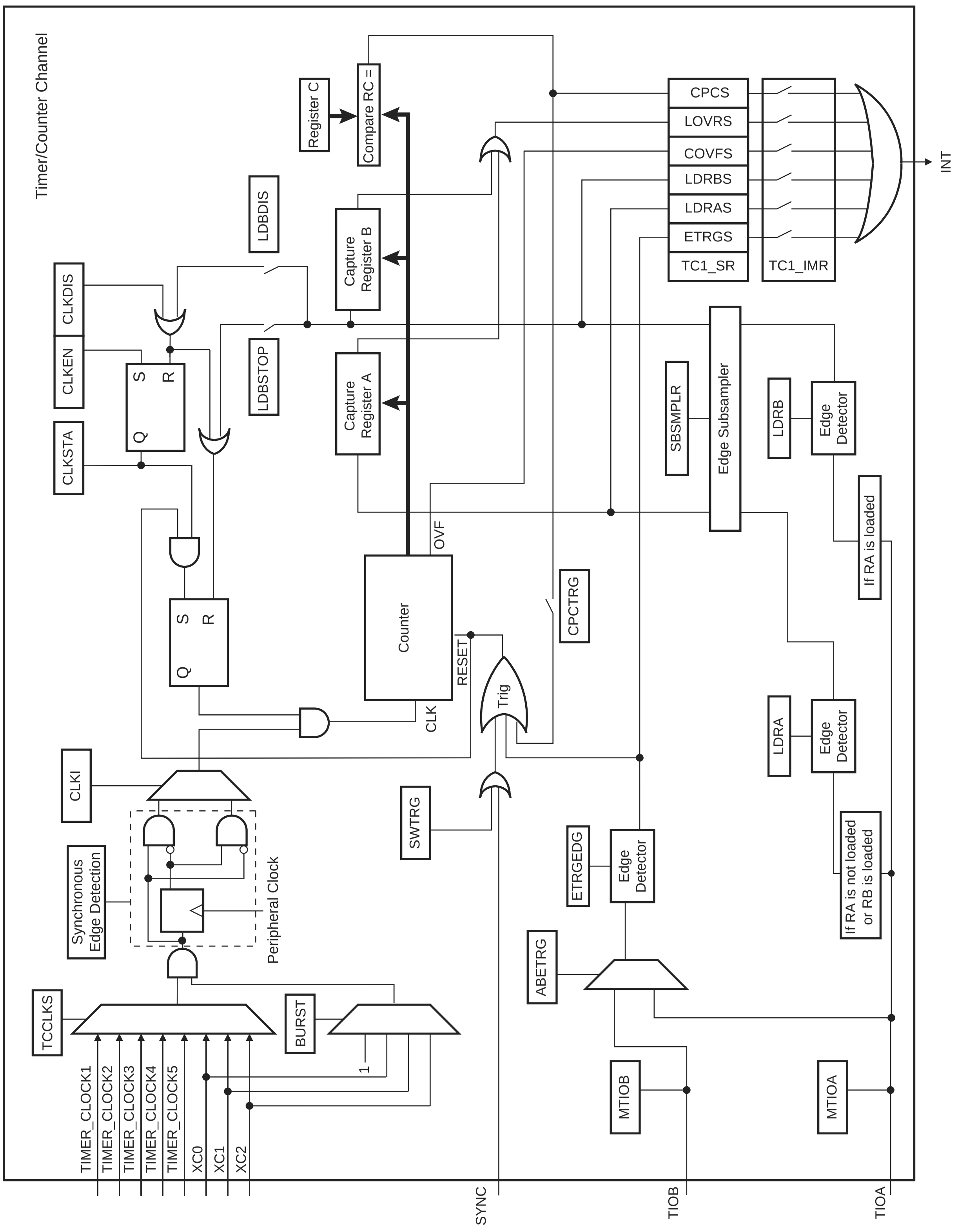


38.6.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 38-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the TC Channel Control Register (TC_CCR). In Capture mode it can be disabled by an RB load event if LDBDIS is set to 1 in the TC_CMR. In Waveform mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the TC_CCR can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the TC_SR.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture mode (LDBSTOP = 1 in TC_CMR) or an RC compare event in Waveform mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands are effective only if the clock is enabled.

Figure 38-6. Capture Mode



39.7.14 PWM Interrupt Enable Register 2

Name: PWM_IER2

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	UNRE	TXBUFE	ENDTX	WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Enable**

- **ENDTX: PDC End of TX Buffer Interrupt Enable**

- **TXBUFE: PDC TX Buffer Empty Interrupt Enable**

- **UNRE: Synchronous Channels Update Underrun Error Interrupt Enable**

- **CMPMx: Comparison x Match Interrupt Enable**

- **CMPUx: Comparison x Update Interrupt Enable**

Table 42-12. Example of Pdelay_Resp Frame in 1588 Version 2 (UDP/IPv6) Format (Continued)

Frame Segment	Value
Dest IP port (Octets 56–57)	013F
Other stuff (Octets 58–61)	—
Message type (Octet 62)	03
Other stuff (Octets 63–93)	—
Version PTP (Octet 94)	02

For the multicast address 011B19000000 sync and delay request frames are recognized depending on the message type field, 00 for sync and 01 for delay request.

Table 42-13. Example of Sync Frame in 1588 Version 2 (Ethernet Multicast) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	011B19000000
SA (Octets 6–11)	—
Type (Octets 12–13)	88F7
Message type (Octet 14)	00
Version PTP (Octet 15)	02

Pdelay request frames need a special multicast address so they can pass through ports blocked by the spanning tree protocol. For the multicast address 0180C200000E sync, Pdelay_Req and Pdelay_Resp frames are recognized depending on the message type field, 00 for sync, 02 for pdelay request and 03 for pdelay response.

Table 42-14. Example of Pdelay_Req Frame in 1588 Version 2 (Ethernet Multicast) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	0180C200000E
SA (Octets 6–11)	—
Type (Octets 12–13)	88F7
Message type (Octet 14)	00
Version PTP (Octet 15)	02

42.6.14 Time Stamp Unit

The TSU consists of a timer and registers to capture the time at which PTP event frames cross the message timestamp point. An interrupt is issued when a capture register is updated.

The timer is implemented as a 62-bit register with the upper 32 bits counting seconds and the lower 30 bits counting nanoseconds. The lower 30 bits roll over when they have counted to one second. An interrupt is generated when the seconds increment. The timer value can be read, written and adjusted through the APB interface.

The amount by which the timer increments each clock cycle is controlled by the timer increment registers (GMAC_TI). Bits 7:0 are the default increment value in nanoseconds and an additional 16 bits of sub-nanosecond resolution are available using the Timer Increment Sub-nanoseconds register (GMAC_TISUBN). If the rest of the

Changing the AFE reference voltage (ADVREF pin) requires a new calibration sequence.

For calibration time, offset and gain error after calibration, refer to the AFE Characteristics in the section “Electrical Characteristics”.

43.6.14 Buffer Structure

The PDC read channel is triggered each time a new data is stored in AFEC_LCDR. The same structure of data is repeatedly stored in AFEC_LCDR each time a trigger event occurs. Depending on the user mode of operation (AFEC_MR, AFEC_CHSR, AFEC_SEQ1R, AFEC_SEQ2R) the structure differs. When TAG is cleared, each data transferred to PDC buffer is carried on a half-word (16-bit) and consists of the last converted data right-aligned. When TAG is set, this data is carried on a word buffer (32-bit) and CHNB carries the channel number, thus simplifying post-processing in the PDC buffer and ensuring the integrity of the PDC buffer.

43.6.15 Fault Output

The AFEC internal fault output is directly connected to PWM fault input. Fault output may be asserted depending on the configuration of AFEC_EMR and AFEC_CWR and converted values. When the compare occurs, the AFEC fault output generates a pulse of one peripheral clock cycle to the PWM fault input. This fault line can be enabled or disabled within the PWM. If it is activated and asserted by the AFEC, the PWM outputs are immediately placed in a safe state (pure combinational path). Note that the AFEC fault output connected to the PWM is not the COMPE bit. Thus the Fault Mode (FMOD) within the PWM configuration must be FMOD = 1.

- **EDGETYP: Edge Type**

Value	Name	Description
0	RISING	Only rising edge of comparator output
1	FALLING	Falling edge of comparator output
2	ANY	Any edge of comparator output

- **INV: Invert Comparator Output**

0 (DIS): Analog comparator output is directly processed.

1 (EN): Analog comparator output is inverted prior to being processed.

- **SELFS: Selection Of Fault Source**

0 (CE): The CE flag is used to drive the FAULT output.

1 (OUTPUT): The output of the analog comparator flag is used to drive the FAULT output.

- **FE: Fault Enable**

0 (DIS): The FAULT output is tied to 0.

1 (EN): The FAULT output is driven by the signal defined by SELFS.

Figure 46-8. Active Mode Measurement Setup

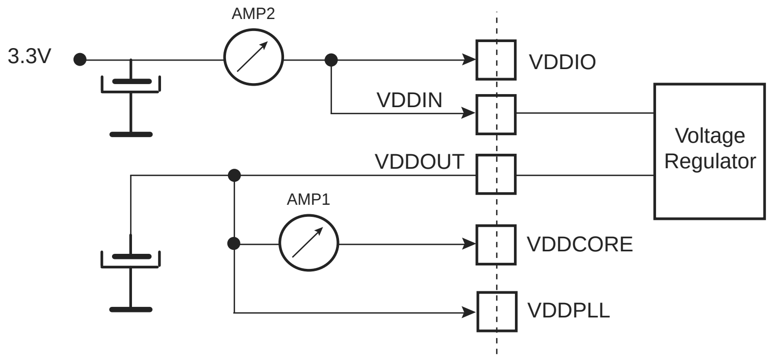


Table 46-13 on page 1365 and Figure 46-14 on page 1366 give the Active Mode Current Consumption in typical conditions.

- VDDCORE at 1.2V
- T_A = 25°C

46.3.3.1 SAM4E Active Power Consumption

Table 46-13. Active Power Consumption with VDDCORE @ 1.2V running from Embedded Memory (IDDCORE - AMP1)

	Core Mark					
	Cache Enable (CE)		**Cache Disable (CD)**			
Core Clock (MHz)	**128-bit Flash access**(1)	**64-bit Flash access**(1)	**128-bit Flash access**(1)	**64-bit Flash access**(1)	**SRAM**	**Unit**
120	21.1	21.0	25.5	19.0	17.9	mA
100	18.1	18.1	22.5	17.2	15.0	
84	15.5	15.5	20.0	16.1	12.86	
64	11.9	11.9	16.4	13.6	9.9	
48	9.0	9.0	12.7	11.7	7.5	
32	6.2	6.2	9.1	8.9	5.2	
24	4.6	4.6	7	6.8	3.9	
12	2.5	2.4	4.1	3.8	2.2	
8	1.9	1.8	2.9	2.8	1.6	
4	1.2	1.1	1.7	1.7	1.0	
2	0.81	0.79	1	1	0.75	
1	0.46	0.46	0.6	0.6	0.44	
0.5	0.38	0.38	0.47	0.44	0.36	

Note: 1. Flash Wait State (FWS) in EEFC_FMR is adjusted depending on core frequency.

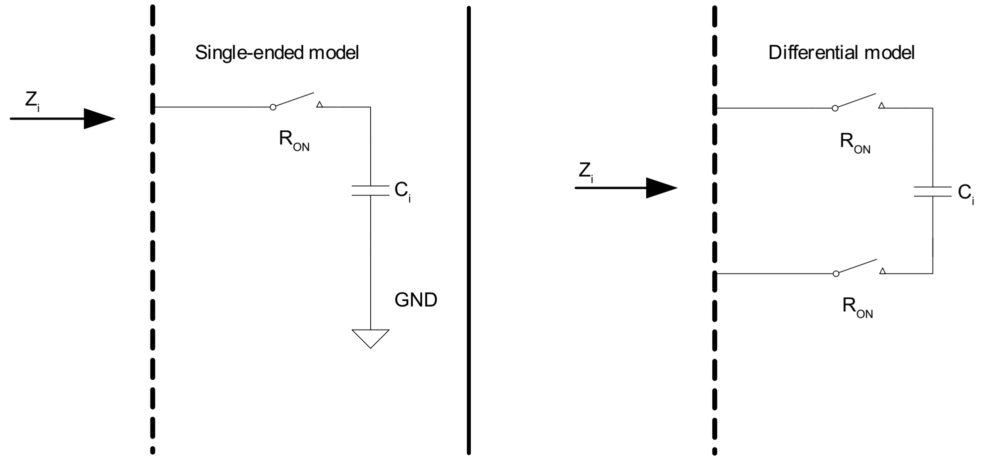
Low Voltage Supply

The ADC operates in 10-bit mode or 12-bit mode. Working at low voltage (V_{DDIN} or/and V_{ADVREF}) between 2 and 2.4V is subject to the following restrictions:

- The field IBCTL must be 00 to reduce the biasing of the ADC under low voltage. See Section 46.7.1.1 "ADC Bias Current".
- In 10-bit mode, the ADC clock should not exceed 5 MHz (max signal bandwidth is 250 kHz).
- In 12-bit mode, the ADC clock should not exceed 2 MHz (max signal bandwidth is 100 kHz).

46.7.5.3 ADC Channel Input Impedance

Figure 46-18. Input Channel Model



where:

- Z_i is input impedance in single-ended or differential mode
- C_i = 1 to 8 pF ±20% depending on the gain value and mode (SE or DIFF); temperature dependency is negligible
- R_{ON} is typical 2 kΩ and 8 kΩ max (worst case process and high temperature)
- R_{ON} is negligible regarding the value of Z_i

The following formula is used to calculate input impedance:

$$Z_i = \frac{1}{f_S \times C_i}$$

where:

- f_S is the sampling frequency of the ADC channel
- Typ values are used to compute ADC input impedance Z_i

Table 46-43. Input Capacitance (C_{IN}) Values

Gain Selection	Single-ended	Differential
0.5	–	2 pF
1	2 pF	4 pF
2	2 pF	8 pF
4	4 pF	–