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®-M3
Core Size	32-Bit Single-Core
Speed	84MHz
Connectivity	CANbus, EBI/EMI, Ethernet, I²C, IrDA, LINbus, Memory Card, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	103
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	68K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam3x4ea-au

10.13.4 CLZ

Count Leading Zeros.

10.13.4.1 Syntax

```
CLZ{cond} Rd, Rm
```

where:

cond is an optional condition code, see “Conditional execution” on page 91.

Rd is the destination register.

Rm is the operand register.

10.13.4.2 Operation

The CLZ instruction counts the number of leading zeros in the value in *Rm* and returns the result in *Rd*. The result value is 32 if no bits are set in the source register, and zero if bit[31] is set.

10.13.4.3 Restrictions

Do not use SP and do not use PC.

10.13.4.4 Condition flags

This instruction does not change the flags.

10.13.4.5 Examples

```
CLZ     R4,R9
CLZNE   R2,R3
```

18.5.4 EEFC Flash Result Register

Name: EEFC_FRR

Address: 0x400E0A0C (0), 0x400E0C0C (1)

Access: Read-only

Offset: 0x0C

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

- **FVALUE: Flash Result Value**

The result of a Flash command is returned in this register. If the size of the result is greater than 32 bits, then the next resulting value is accessible at the next register read.

Single-buffer DMAC transfer: Consists of a single buffer.

Multi-buffer DMAC transfer: A DMAC transfer may consist of multiple DMAC buffers. Multi-buffer DMAC transfers are supported through buffer chaining (linked list pointers), auto-reloading of channel registers, and contiguous buffers. The source and destination can independently select which method to use.

- **Linked lists (buffer chaining) –** A descriptor pointer (DSCR) points to the location in system memory where the next linked list item (LLI) exists. The LLI is a set of registers that describe the next buffer (buffer descriptor) and a descriptor pointer register. The DMAC fetches the LLI at the beginning of every buffer when buffer chaining is enabled.
- **Contiguous buffers –** Where the address of the next buffer is selected to be a continuation from the end of the previous buffer.

Channel locking: Software can program a channel to keep the AHB master interface by locking the arbitration for the master bus interface for the duration of a DMAC transfer, buffer, or chunk.

Bus locking: Software can program a channel to maintain control of the AMBA bus by asserting hmastlock for the duration of a DMAC transfer, buffer, or transaction (single or chunk). Channel locking is asserted for the duration of bus locking at a minimum.

22.4.2 Memory Peripherals

Figure 22-3 on page 342 shows the DMAC transfer hierarchy of the DMAC for a memory peripheral. There is no handshaking interface with the DMAC, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request. The alternative to not having a transaction-level handshaking interface is to allow the DMAC to attempt AMBA transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these AMBA transfers, it inserts wait states onto the bus until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus. By using the handshaking interface, the peripheral can signal to the DMAC that it is ready to transmit/receive data, and then the DMAC can access the peripheral without the peripheral inserting wait states onto the bus.

22.4.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or chunk transfers. The operation of the handshaking interface is different and depends on whether the peripheral or the DMAC is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMAC that it is ready to transfer/accept data over the AMBA bus. A non-memory peripheral can request a DMAC transfer through the DMAC using one of two handshaking interfaces:

- Hardware handshaking
- Software handshaking

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

22.4.3.1 Software Handshaking

When the slave peripheral requires the DMAC to perform a DMAC transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller.

The interrupt service routine then uses the software registers to initiate and control a DMAC transaction. These software registers are used to implement the software handshaking interface.

The SRC_H2SEL/DST_H2SEL bit in the DMAC_CFGx channel configuration register must be set to zero to enable software handshaking.

24.6.2 SDRAM Controller Read Cycle

The SDRAM Controller allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAM Controller keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAM controller automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active commands (t_{RP}) and between active and read command (t_{RCD}). These two parameters are set in the configuration register of the SDRAM Controller. After a read command, additional wait states are generated to comply with the CAS latency (1, 2 or 3 clock delays specified in the configuration register).

For a single access or an incremented burst of unspecified length, the SDRAM Controller anticipates the next access. While the last value of the column is returned by the SDRAM Controller on the bus, the SDRAM Controller anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

Figure 24-3. Read Burst SDRAM Access

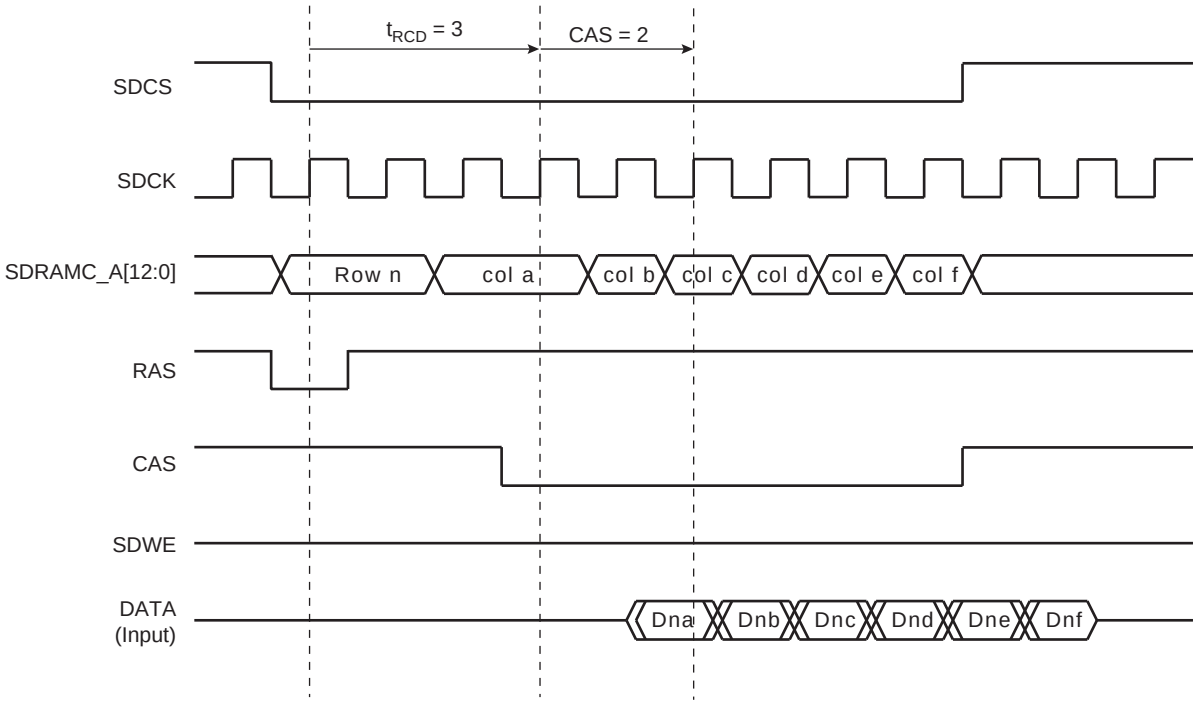
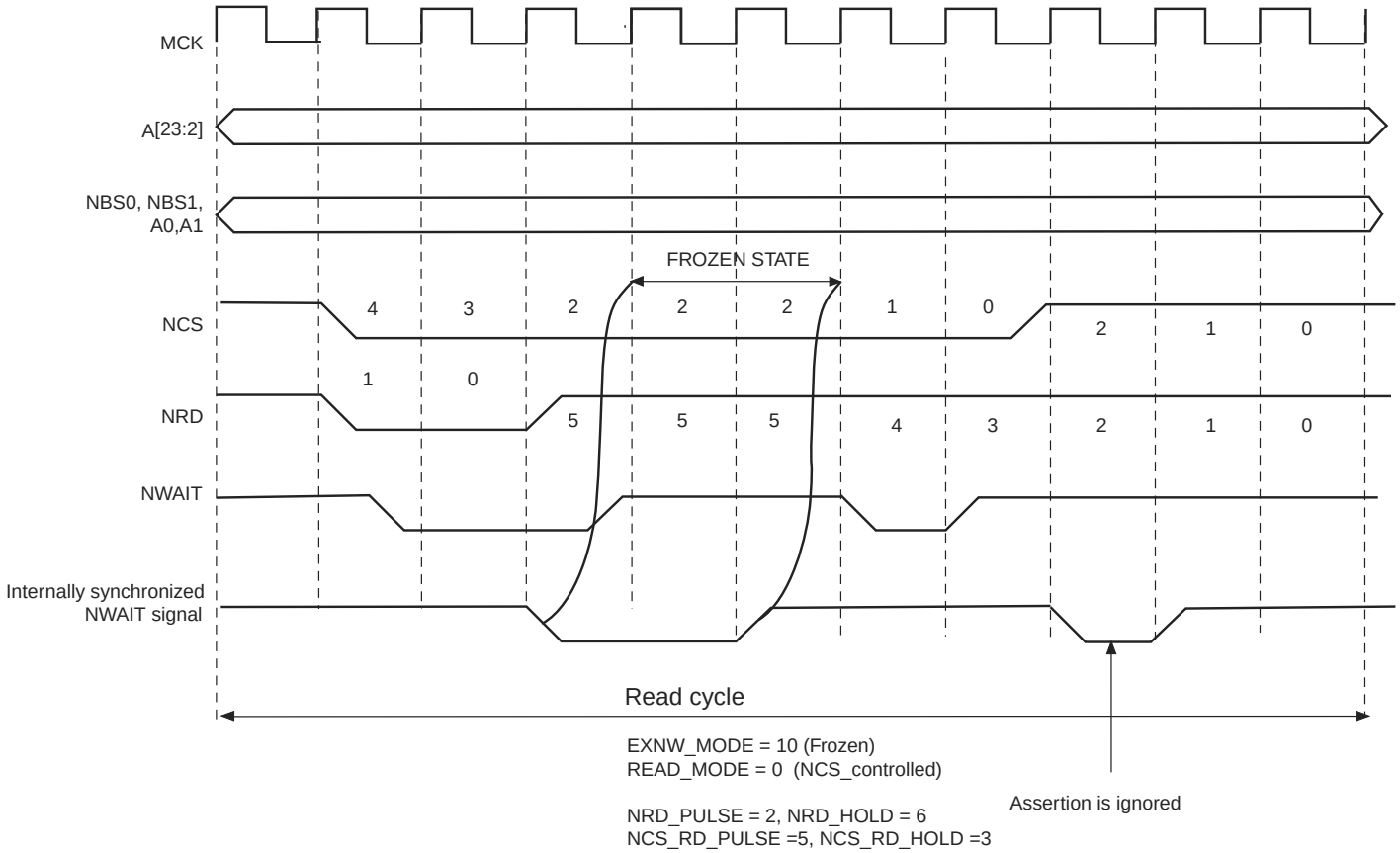


Figure 25-24. Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



28.15.6 PMC Peripheral Clock Status Register 0

Name: PMC_PCSR0

Address: 0x400E0618

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: To get PIDx, refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet. Other peripherals status can be read in PMC_PCSR1 (Section 28.15.25 ”PMC Peripheral Clock Status Register 1”).

30.9.17 SSC Write Protect Mode Register

Name: SSC_WPMR

Address: 0x400040E4

Access: Read-write

Reset: See Table 30-6

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 Protect if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

Protects the registers:

- “SSC Clock Mode Register” on page 593
- “SSC Receive Clock Mode Register” on page 594
- “SSC Receive Frame Mode Register” on page 596
- “SSC Transmit Clock Mode Register” on page 598
- “SSC Transmit Frame Mode Register” on page 600
- “SSC Receive Compare 0 Register” on page 606
- “SSC Receive Compare 1 Register” on page 607

- **WPKEY: Write Protect KEY**

Should be written at value 0x535343 (“SSC” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

33.11.8 TWI Interrupt Disable Register

Name: TWI_IDR

Address: 0x4008C028 (0), 0x40090028 (1)

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Disable**
- **RXRDY: Receive Holding Register Ready Interrupt Disable**
- **TXRDY: Transmit Holding Register Ready Interrupt Disable**
- **SVACC: Slave Access Interrupt Disable**
- **GACC: General Call Access Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **NACK: Not Acknowledge Interrupt Disable**
- **ARBLST: Arbitration Lost Interrupt Disable**
- **SCL_WS: Clock Wait State Interrupt Disable**
- **EOSACC: End Of Slave Access Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

35.8.4 USART Interrupt Disable Register

Name: US_IDR

Address: 0x4009800C (0), 0x4009C00C (1), 0x400A000C (2), 0x400A400C (3)

Access: Write-only

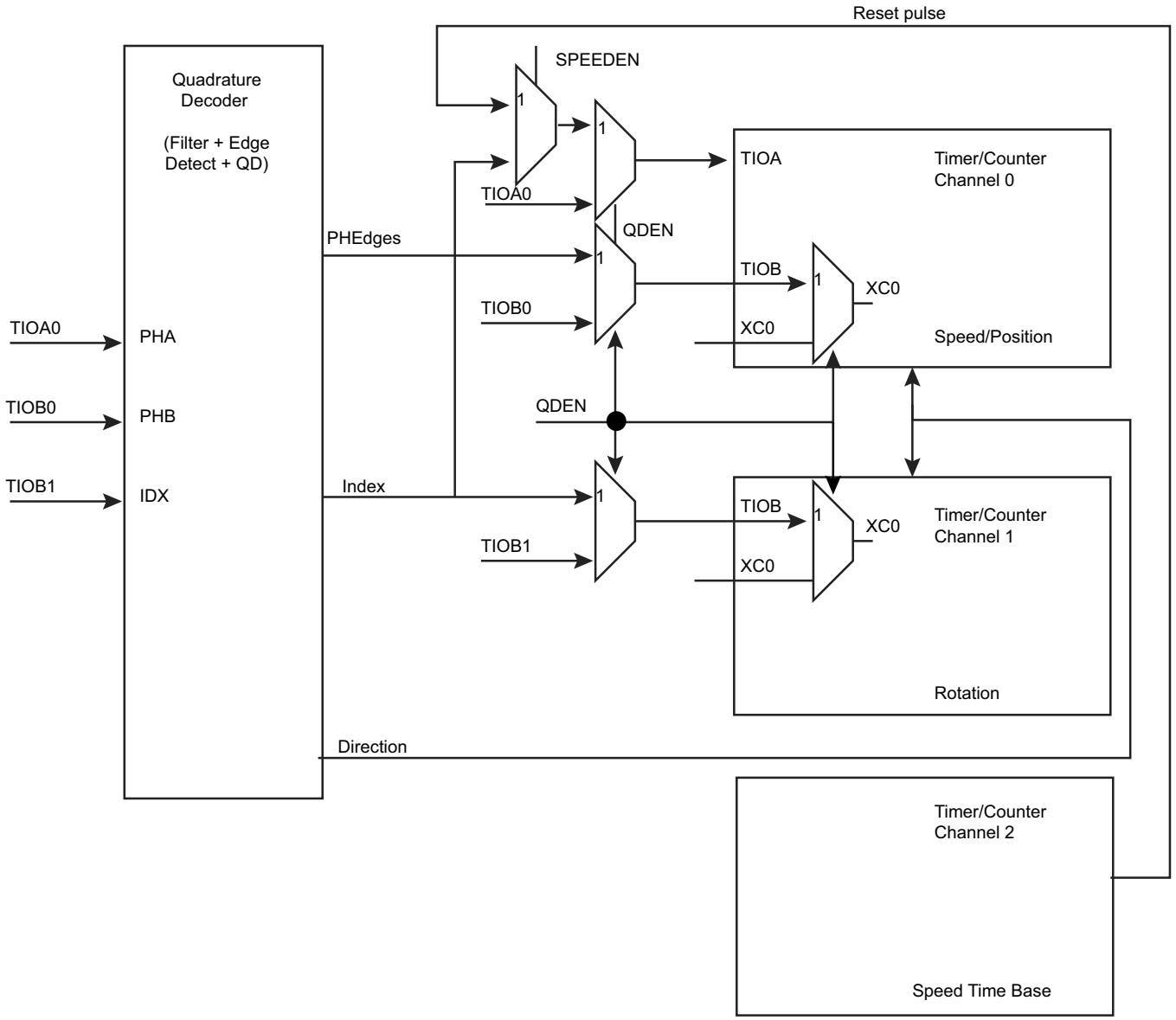
31	30	29	28	27	26	25	24
–	–	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	MANE
23	22	21	20	19	18	17	16
–	–	–		CTSIC	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	NACK/LINBK	RXBUFF	TXBUFE	ITER/UNRE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Transfer Interrupt Disable**
- **ENDTX: End of Transmit Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITER: Max number of Repetitions Reached Disable**
- **UNRE: SPI Underrun Error Disable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Disable**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Disable**

Figure 36-15. Predefined Connection of the Quadrature Decoder with Timer Counters



36.6.14.2 Input Pre-processing

Input pre-processing consists of capabilities to take into account rotary sensor factors such as polarities and phase definition followed by configurable digital filtering.

Each input can be negated and swapping PHA, PHB is also configurable.

The MAXFILT field in the TC_BMR is used to configure a minimum duration for which the pulse is stated as valid. When the filter is active, pulses with a duration lower than MAXFILT +1 $\times$ $t_{peripheral\ clock}$ ns are not passed to downstream logic.

37.14.3 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 “HSMCI Write Protect Mode Register” on page 967.

- **DTOCYC: Data Timeout Cycle Number**

These fields determine 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**

Multiplier is defined by DTOMUL as shown in the following table:

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) raises.

38.7.36 PWM Comparison x Mode Update Register

Name: PWM_CMPMUPDx

Address: 0x4009413C [0], 0x4009414C [1], 0x4009415C [2], 0x4009416C [3], 0x4009417C [4], 0x4009418C [5], 0x4009419C [6], 0x400941AC [7]

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	CPRUPD			

7	6	5	4	3	2	1	0
CTRUPD				–	–	–	CENUPD

This register acts as a double buffer for the CEN, CTR, CPR and CUPR values. This prevents an unexpected comparison x match.

- **CENUPD: Comparison x Enable Update**

0 = The comparison x is disabled and can not match.

1 = The comparison x is enabled and can match.

- **CTRUPD: Comparison x Trigger Update**

The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.

- **CPRUPD: Comparison x Period Update**

CPR defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.

- **CUPRUPD: Comparison x Update Period Update**

Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.

38.7.41 PWM Channel Period Update Register

Name: PWM_CPRDUPDx [x=0..7]

Address: 0x40094210 [0], 0x40094230 [1], 0x40094250 [2], 0x40094270 [3], 0x40094290 [4], 0x400942B0 [5], 0x400942D0 [6], 0x400942F0 [7]

Access: Write-only

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

23	22	21	20	19	18	17	16
CPRDUPD							

15	14	13	12	11	10	9	8
CPRDUPD							

7	6	5	4	3	2	1	0
CPRDUPD							

This register can only be written if the bits WPSWS3 and WPHWS3 are cleared in “PWM Write Protect Status Register” on page 1039.

This register acts as a double buffer for the CPRD value. This prevents an unexpected waveform when modifying the waveform period.

Only the first 16 bits (channel counter size) are significant.

- **CPRDUPD: Channel Period Update**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRDUPD)}{MCK}$$

– By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPDUPD \times DIVA)}{MCK} \text{ or } \frac{(CRPDUPD \times DIVB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM master clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRDUPD)}{MCK}$$

– By using the PWM master clock (MCK) divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRDUPD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRDUPD \times DIVB)}{MCK}$$

39.6.2.2 Device Global Interrupt Status Register

Name: UOTGHS_DEVISR

Address: 0x400AC004

Access: Read-only

31	30	29	28	27	26	25	24
–	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	PEP_9	PEP_8	PEP_7	PEP_6	PEP_5	PEP_4
15	14	13	12	11	10	9	8
PEP_3	PEP_2	PEP_1	PEP_0	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSM	EORSM	WAKEUP	EORST	SOF	MSOF	SUSP

- **DMA_x: DMA Channel x Interrupt**

This bit is set when an interrupt is triggered by the DMA channel x. This triggers a USB interrupt if DMA_x is one.

This bit is cleared when the UOTGHS_DEVDMASTATUSx interrupt source is cleared.

- **PEP_x: Endpoint x Interrupt**

This bit is set when an interrupt is triggered by the endpoint x (UOTGHS_DEVEPTISRx, UOTGHS_DEVEPTIMRx). This triggers a USB interrupt if UOTGHS_DEVIMR.PEP_x is one.

This bit is cleared when the interrupt source is serviced.

- **UPRSM: Upstream Resume Interrupt**

This bit is set when the UOTGHS sends a resume signal called “Upstream Resume”. This triggers a USB interrupt if UOTGHS_DEVIMR.UPRSME is one.

This bit is cleared when the UOTGHS_DEVICR.UPRSMC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).

- **EORSM: End of Resume Interrupt**

This bit is set when the UOTGHS detects a valid “End of Resume” signal initiated by the host. This triggers a USB interrupt if UOTGHS_DEVIMR.EORSME is one.

This bit is cleared when the UOTGHS_DEVICR.EORSMC bit is written to one to acknowledge the interrupt.

- **WAKEUP: Wake-Up Interrupt**

This bit is set when the UOTGHS is reactivated by a filtered non-idle signal from the lines (not by an upstream resume). This triggers an interrupt if UOTGHS_DEVIMR.WAKEUPE is one.

This bit is cleared when the UOTGHS_DEVICR.WAKEUPC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).

This bit is cleared when the Suspend (SUSP) interrupt bit is set.

This interrupt is generated even if the clock is frozen by the UOTGHS_CTRL.FRZCLK bit.

- **EORST: End of Reset Interrupt**

This bit is set when a USB “End of Reset” has been detected. This triggers a USB interrupt if UOTGHS_DEVIMR.EORSTE is one.

This bit is cleared when the UOTGHS_DEVICR.EORSTC bit is written to one to acknowledge the interrupt.

39.6.2.8 Device Endpoint Register

Name: UOTGHS_DEVEPT

Address: 0x400AC01C

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	EPRST8
23	22	21	20	19	18	17	16
EPRST7	EPRST6	EPRST5	EPRST4	EPRST3	EPRST2	EPRST1	EPRST0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	EPEN8
7	6	5	4	3	2	1	0
EPEN7	EPEN6	EPEN5	EPEN4	EPEN3	EPEN2	EPEN1	EPEN0

- **EPRSTx: Endpoint x Reset**

Writing a one to this bit will reset the endpoint x FIFO prior to any other operation, upon hardware reset or when a USB bus reset has been received. This resets the endpoint x registers (UOTGHS_DEVEPTCFGx, UOTGHS_DEVEPTISRx, UOTGHS_DEVEPTIMRx) but not the endpoint configuration (UOTGHS_DEVEPTCFGx.ALLOC, UOTGHS_DEVEPTCFGx.EPBK, UOTGHS_DEVEPTCFGx.EPSIZE, UOTGHS_DEVEPTCFGx.EPDIR, UOTGHS_DEVEPTCFGx.EPTYPE).

All the endpoint mechanism (FIFO counter, reception, transmission, etc.) is reset apart from the Data Toggle Sequence field (UOTGHS_DEVEPTISRx.DTSEQ) which can be cleared by setting the UOTGHS_DEVEPTIMRx.RSTDT bit (by writing a one to the UOTGHS_DEVEPTIERx.RSTDTS bit).

The endpoint configuration remains active and the endpoint is still enabled.

Writing a zero to this bit will complete the reset operation and start using the FIFO.

This bit is cleared upon receiving a USB reset.

- **EPENx: Endpoint x Enable**

0: The endpoint x is disabled, what forces the endpoint x state to inactive (no answer to USB requests) and resets the endpoint x registers (UOTGHS_DEVEPTCFGx, UOTGHS_DEVEPTISRx, UOTGHS_DEVEPTIMRx) but not the endpoint configuration (UOTGHS_DEVEPTCFGx.ALLOC, UOTGHS_DEVEPTCFGx.EPBK, UOTGHS_DEVEPTCFGx.EPSIZE, UOTGHS_DEVEPTCFGx.EPDIR, UOTGHS_DEVEPTCFGx.EPTYPE).

1: The endpoint x is enabled.

- **MDATAE: MData Interrupt**

This bit is set when UOTGHS_DEVEPTIERx.MDATAES bit is written to one. This will enable the Multiple DATA interrupt. (see DTSEQ bits)

This bit is cleared when UOTGHS_DEVEPTIDRx.MDATAEC bit is written to one. This will disable the Multiple DATA interrupt.

- **SHORTPACKETE: Short Packet Interrupt**

If this bit is set for non-control IN endpoints, a short packet transmission is guaranteed upon ending a DMA transfer, thus signaling an end of isochronous frame or a bulk or interrupt end of transfer, provided that he End of DMA Buffer Output Enable (END_B_EN) bit and the Automatic Switch (AUTOSW) bit are written to one.

This bit is set when UOTGHS_DEVEPTIERx.SHORTPACKETES bit is written to one. This will enable the Short Packet interrupt (UOTGHS_DEVEPTISRx.SHORTPACKET).

This bit is cleared when UOTGHS_DEVEPTIDRx.SHORTPACKETEC bit is written to one. This will disable the Short Packet interrupt (UOTGHS_DEVEPTISRx.SHORTPACKET).

- **STALLEDE: STALLed Interrupt**

This bit is set when UOTGHS_DEVEPTIERx.STALLEDES bit is written to one. This will enable the STALLed interrupt (UOTGHS_DEVEPTISRx.STALLEDI).

This bit is cleared when UOTGHS_DEVEPTIDRx.STALLEDEC bit is written to one. This will disable the STALLed interrupt (UOTGHS_DEVEPTISRx.STALLEDI).

- **CRCERRE: CRC Error Interrupt**

This bit is set when UOTGHS_DEVEPTIERx.CRCERRES bit is written to one. This will enable the CRC Error interrupt (UOTGHS_DEVEPTISRx.CRCERRI).

This bit is cleared when UOTGHS_DEVEPTIDRx.CRCERREC bit is written to one. This will disable the CRC Error interrupt (UOTGHS_DEVEPTISRx.CRCERRI).

- **OVERFE: Overflow Interrupt**

This bit is set when UOTGHS_DEVEPTIERx.OVERFES bit is written to one. This will enable the Overflow interrupt (UOTGHS_DEVEPTISRx.OVERFI).

This bit is cleared when UOTGHS_DEVEPTIDRx.OVERFEC bit is written to one. This will disable the Overflow interrupt (UOTGHS_DEVEPTISRx.OVERFI).

- **NAKINE: NAKed IN Interrupt**

This bit is set when UOTGHS_DEVEPTIERx.NAKINES bit is written to one. This will enable the NAKed IN interrupt (UOTGHS_DEVEPTISRx.NAKINI).

This bit is cleared when UOTGHS_DEVEPTIDRx.NAKINEC bit is written to one. This will disable the NAKed IN interrupt (UOTGHS_DEVEPTISRx.NAKINI).

- **HBISOFLUSHE: High Bandwidth Isochronous IN Flush Interrupt**

This bit is set when UOTGHS_DEVEPTIERx.HBISOFLUSHES bit is written to one. This will enable the HBISOFLUSHI interrupt.

This bit is cleared when UOTGHS_DEVEPTIDRx.HBISOFLUSHEC bit disable the HBISOFLUSHI interrupt.

41.4 Functional Description

The MACB has several clock domains:

- System bus clock (AHB and APB): DMA and register blocks
- Transmit clock: transmit block
- Receive clock: receive and address checker block

The system bus clock must run at least as fast as the receive clock and transmit clock (25 MHz at 100 Mbps, and 2.5 MHZ at 10 Mbps).

Figure 41-1 illustrates the different blocks of the EMAC module.

The control registers drive the MDIO interface, setup up DMA activity, start frame transmission and select modes of operation such as full- or half-duplex.

The receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the address checking block and DMA interface.

The transmit block takes data from the DMA interface, adds preamble and, if necessary, pad and FCS, and transmits data according to the CSMA/CD (carrier sense multiple access with collision detect) protocol. The start of transmission is deferred if CRS (carrier sense) is active.

If COL (collision) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. CRS and COL have no effect in full duplex mode.

The DMA block connects to external memory through its AHB bus interface. It contains receive and transmit FIFOs for buffering frame data. It loads the transmit FIFO and empties the receive FIFO using AHB bus master operations. Receive data is not sent to memory until the address checking logic has determined that the frame should be copied. Receive or transmit frames are stored in one or more buffers. Receive buffers have a fixed length of 128 bytes. Transmit buffers range in length between 0 and 2047 bytes, and up to 128 buffers are permitted per frame. The DMA block manages the transmit and receive framebuffer queues. These queues can hold multiple frames.

41.4.1 Clock

Synchronization module in the EMAC requires that the bus clock (hclk) runs at the speed of the macb_tx/rx_clk at least, which is 25 MHz at 100 Mbps, and 2.5 MHz at 10 Mbps.

41.4.2 Memory Interface

Frame data is transferred to and from the EMAC through the DMA interface. All transfers are 32-bit words and may be single accesses or bursts of 2, 3 or 4 words. Burst accesses do not cross sixteen-byte boundaries. Bursts of 4 words are the default data transfer; single accesses or bursts of less than four words may be used to transfer data at the beginning or the end of a buffer.

The DMA controller performs six types of operation on the bus. In order of priority, these are:

1. Receive buffer manager write
2. Receive buffer manager read
3. Transmit data DMA read
4. Receive data DMA write
5. Transmit buffer manager read
6. Transmit buffer manager write

41.4.2.1 FIFO

The FIFO depths are 128 bytes for receive and 128 bytes for transmit and are a function of the system clock speed, memory latency and network speed.

41.6 Ethernet MAC 10/100 (EMAC) User Interface

Table 41-6. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Network Control Register	EMAC_NCR	Read-write	0
0x04	Network Configuration Register	EMAC_NCFGR	Read-write	0x800
0x08	Network Status Register	EMAC_NSR	Read-only	-
0x0C	Reserved			
0x10	Reserved			
0x14	Transmit Status Register	EMAC_TSR	Read-write	0x0000_0000
0x18	Receive Buffer Queue Pointer Register	EMAC_RBQP	Read-write	0x0000_0000
0x1C	Transmit Buffer Queue Pointer Register	EMAC_TBQP	Read-write	0x0000_0000
0x20	Receive Status Register	EMAC_RSR	Read-write	0x0000_0000
0x24	Interrupt Status Register	EMAC_ISR	Read-write	0x0000_0000
0x28	Interrupt Enable Register	EMAC_IER	Write-only	-
0x2C	Interrupt Disable Register	EMAC_IDR	Write-only	-
0x30	Interrupt Mask Register	EMAC_IMR	Read-only	0x0000_3FFF
0x34	Phy Maintenance Register	EMAC_MAN	Read-write	0x0000_0000
0x38	Pause Time Register	EMAC_PTR	Read-write	0x0000_0000
0x3C	Pause Frames Received Register	EMAC_PFR	Read-write	0x0000_0000
0x40	Frames Transmitted Ok Register	EMAC_FTO	Read-write	0x0000_0000
0x44	Single Collision Frames Register	EMAC_SCF	Read-write	0x0000_0000
0x48	Multiple Collision Frames Register	EMAC_MCF	Read-write	0x0000_0000
0x4C	Frames Received Ok Register	EMAC_FRO	Read-write	0x0000_0000
0x50	Frame Check Sequence Errors Register	EMAC_FCSE	Read-write	0x0000_0000
0x54	Alignment Errors Register	EMAC_ALE	Read-write	0x0000_0000
0x58	Deferred Transmission Frames Register	EMAC_DTF	Read-write	0x0000_0000
0x5C	Late Collisions Register	EMAC_LCOL	Read-write	0x0000_0000
0x60	Excessive Collisions Register	EMAC_ECOL	Read-write	0x0000_0000
0x64	Transmit Underrun Errors Register	EMAC_TUND	Read-write	0x0000_0000
0x68	Carrier Sense Errors Register	EMAC_CSE	Read-write	0x0000_0000
0x6C	Receive Resource Errors Register	EMAC_RRE	Read-write	0x0000_0000
0x70	Receive Overrun Errors Register	EMAC_ROV	Read-write	0x0000_0000
0x74	Receive Symbol Errors Register	EMAC_RSE	Read-write	0x0000_0000
0x78	Excessive Length Errors Register	EMAC_ELE	Read-write	0x0000_0000
0x7C	Receive Jabbers Register	EMAC_RJA	Read-write	0x0000_0000
0x80	Undersize Frames Register	EMAC_USF	Read-write	0x0000_0000
0x84	SQE Test Errors Register	EMAC_STE	Read-write	0x0000_0000
0x88	Received Length Field Mismatch Register	EMAC_RLE	Read-write	0x0000_0000

45.10.6.2 Write Timings

Table 45-52. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	Min		Max		Unit
	VDDIO supply	1.8V Domain	3.3V Domain	1.8V Domain	3.3V Domain	
HOLD or NO HOLD SETTINGS (NWE_HOLD ≠ 0, NWE_HOLD = 0)						
SMC_{15}	Data Out Valid before NWE High	NWE_PULSE × t_{CPMCK} - 7.5	NWE_PULSE × t_{CPMCK} - 7			ns
SMC_{16}	NWE Pulse Width	NWE_PULSE × t_{CPMCK} - 3	NWE_PULSE × t_{CPMCK} - 3			ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NWE low	NWE_SETUP × t_{CPMCK} + 6	NWE_SETUP × t_{CPMCK} + 6			ns
SMC_{18}	NCS low before NWE high	(NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × t_{CPMCK} + 10	(NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × t_{CPMCK} + 12			ns
HOLD SETTINGS (NWE_HOLD ≠ 0)						
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 change	NWE_HOLD × t_{CPMCK} - 2.4	NWE_HOLD × t_{CPMCK} - 1.8			ns
SMC_{20}	NWE High to NCS Inactive(1)	(NWE_HOLD - NCS_WR_HOLD) × t_{CPMCK} - 0.3	(NWE_HOLD - NCS_WR_HOLD) × t_{CPMCK} - 0.3			ns
NO HOLD SETTINGS (NWE_HOLD = 0)						
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25, NCS change(1)	4	4			ns

Notes: 1. Hold length = total cycle duration - setup duration - pulse duration. “Hold length” is for “NCS_WR_HOLD length” or “NWE_HOLD length”.

Table 45-53. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min		Max		Unit
	VDDIO supply	1.8V Domain	3.3V Domain	1.8V Domain	3.3V Domain	
SMC_{22}	Data Out Valid before NCS High	NCS_WR_PULSE × t_{CPMCK} - 1	NCS_WR_PULSE × t_{CPMCK} - 1			ns
SMC_{23}	NCS Pulse Width	NCS_WR_PULSE × t_{CPMCK} - 6	NCS_WR_PULSE × t_{CPMCK} - 6			ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2–A25 valid before NCS low	NCS_WR_SETUP × t_{CPMCK} - 5	NCS_WR_SETUP × t_{CPMCK} - 5			ns
SMC_{25}	NWE low before NCS high	(NCS_WR_SETUP - NWE_SETUP + NCS pulse) × t_{CPMCK} + 1.3	(NCS_WR_SETUP - NWE_SETUP + NCS pulse) × t_{CPMCK} + 1.3			ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2–A25, change	NCS_WR_HOLD × t_{CPMCK} - 8.2	NCS_WR_HOLD × t_{CPMCK} - 9.6			ns
SMC_{27}	NCS High to NWE Inactive	(NCS_WR_HOLD - NWE_HOLD) × t_{CPMCK} - 6.2	(NCS_WR_HOLD - NWE_HOLD) × t_{CPMCK} - 9			ns

Figure 45-38. EMAC MII Mode