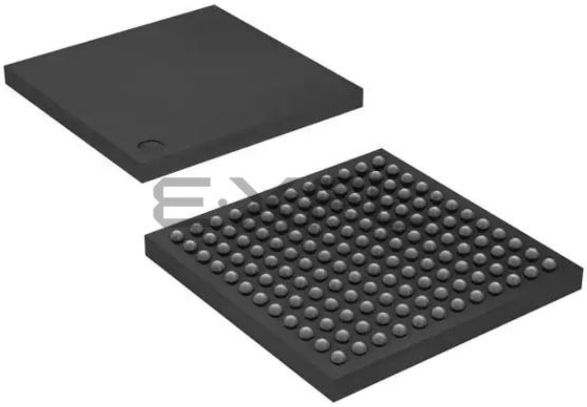
Microchip Technology - ATSAM3X4EA-CU Datasheet



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-LFBGA
Supplier Device Package	144-BGA (13x13)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam3x4ea-cu

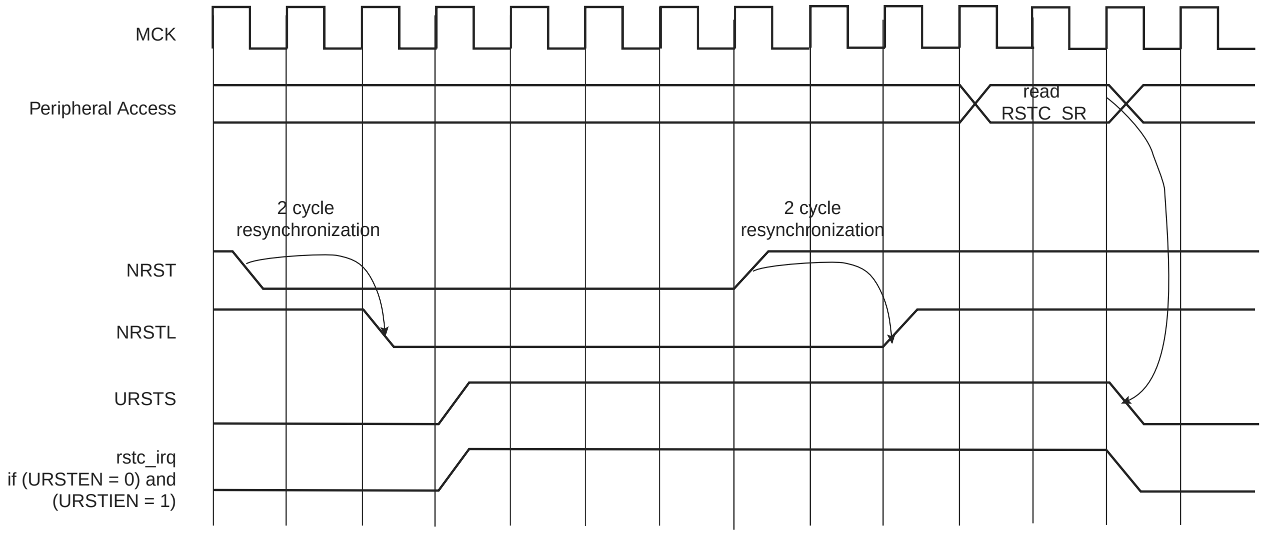
9.3.1 PIO Controller A Multiplexing

Table 9-2. Multiplexing on PIO Controller A (PIOA)

I/O Line	Peripheral A	Peripheral B	Extra Function	Comments
PA0	CANTX0	PWML3		
PA1	CANRX0	PCK0	WKUP0[1]	
PA2	TIOA1	NANDRDY	AD0[2]	
PA3	TIOB1	PWMFI1	AD1/WKUP1[3]	
PA4	TCLK1	NWAIT	AD2[2]	
PA5	TIOA2	PWMFI0	WKUP2[1]	
PA6	TIOB2	NCS0	AD3[2]	
PA7	TCLK2	NCS1	WKUP3[1]	
PA8	URXD	PWMH0	WKUP4[1]	
PA9	UTXD	PWMH3		
PA10	RXD0	DATRG	WKUP5[1]	
PA11	TXD0	ADTRG	WKUP6[1]	
PA12	RXD1	PWML1	WKUP7[1]	
PA13	TXD1	PWMH2		
PA14	RTS1	TK		
PA15	CTS1	TF	WKUP8[1]	
PA16	SPCK1	TD	AD7[2]	
PA17	TWD0	SPCK0		
PA18	TWCK0	A20	WKUP9[1]	
PA19	MCCK	PWMH1		
PA20	MCCDA	PWML2		
PA21	MCDA0	PWML0		
PA22	MCDA1	TCLK3	AD4[2]	
PA23	MCDA2	TCLK4	AD5[2]	
PA24	MCDA3	PCK1	AD6[2]	
PA25	SPI0_MISO	A18		
PA26	SPI0_MOSI	A19		
PA27	SPI0_SPCK	A20	WKUP10[1]	
PA28	SPI0_NPCS0	PCK2	WKUP11[1]	
PA29	SPI0_NPCS1	NRD		
PA30	SPI0_NPCS2	PCK1		217 pins
PA31	SPI0_NPCS3	PCK2		217 pins

Notes:
1. WKUPx can be used if PIO controller defines the I/O line as "input".
2. To select this extra function, refer to Section 43.5.3 “Analog Inputs”.
3. Analog input has priority over WKUPx pin.

Figure 12-7. Reset Controller Status and Interrupt



Commands and read operations can be performed in parallel only on different memory planes. Code can be fetched from one memory plane while a write or an erase operation is performed on another.

Table 18-2. Set of Commands

Command	Value	Mnemonic
Get Flash Descriptor	0x00	GETD
Write page	0x01	WP
Write page and lock	0x02	WPL
Erase page and write page	0x03	EWP
Erase page and write page then lock	0x04	EWPL
Erase all	0x05	EA
Set Lock Bit	0x08	SLB
Clear Lock Bit	0x09	CLB
Get Lock Bit	0x0A	GLB
Set GPNVM Bit	0x0B	SGPB
Clear GPNVM Bit	0x0C	CGPB
Get GPNVM Bit	0x0D	GGPB
Start Read Unique Identifier	0x0E	STUI
Stop Read Unique Identifier	0x0F	SPUI
Get CALIB Bit	0x10	GCALB

In order to perform one of these commands, the Flash Command Register (EEFC_FCR) has to be written with the correct command using the FCMD field. As soon as the EEFC_FCR register is written, **the FRDY flag and the FVALUE field in the EEFC_FRR register are automatically cleared**. Once the current command is achieved, then the FRDY flag is automatically set. If an interrupt has been enabled by setting the FRDY bit in EEFC_FMR, the corresponding interrupt line of the NVIC is activated. (Note that this is true for all commands except for the STUI Command. The FRDY flag is not set when the STUI command is achieved.)

All the commands are protected by the same keyword, which has to be written in the 8 highest bits of the EEFC_FCR register.

Writing EEFC_FCR with data that does not contain the correct key and/or with an invalid command has no effect on the whole memory plane, but the FCMDE flag is set in the EEFC_FSR register. This flag is automatically cleared by a read access to the EEFC_FSR register.

When the current command writes or erases a page in a locked region, the command has no effect on the whole memory plane, but the FLOCKE flag is set in the EEFC_FSR register. This flag is automatically cleared by a read access to the EEFC_FSR register.

Table 18-3. Flash Descriptor Definition

Symbol	Word Index	Description
FL_ID	0	Flash Interface Description
FL_SIZE	1	Flash size in bytes
FL_PAGE_SIZE	2	Page size in bytes
FL_NB_PLANE	3	Number of planes.
FL_PLANE[0]	4	Number of bytes in the first plane.
...		
FL_PLANE[FL_NB_PLANE-1]	4 + FL_NB_PLANE - 1	Number of bytes in the last plane.
FL_NB_LOCK	4 + FL_NB_PLANE	Number of lock bits. A bit is associated with a lock region. A lock bit is used to prevent write or erase operations in the lock region.
FL_LOCK[0]	4 + FL_NB_PLANE + 1	Number of bytes in the first lock region.
...		

18.4.3.2 Write Commands

Several commands can be used to program the Flash.

Flash technology requires that an erase is done before programming. The full memory plane can be erased at the same time, or several pages can be erased at the same time (refer to Section ”The Partial Programming mode works only with 128-bit (or higher) boundaries. It cannot be used with boundaries lower than 128 bits (8, 16 or 32-bit for example).”). Also, a page erase can be automatically done before a page write using EWP or EWPL commands.

After programming, the page (the whole lock region) can be locked to prevent miscellaneous write or erase sequences. The lock bit can be automatically set after page programming using WPL or EWPL commands.

Data to be written are stored in an internal latch buffer. The size of the latch buffer corresponds to the page size. The latch buffer wraps around within the internal memory area address space and is repeated as many times as the number of pages within this address space.

Note: Writing of 8-bit and 16-bit data is not allowed and may lead to unpredictable data corruption.

Write operations are performed in a number of wait states equal to the number of wait states for read operations.

Data are written to the latch buffer before the programming command is written to the Flash Command Register EEFC_FCR. The sequence is as follows:

- Write the full page, at any page address, within the internal memory area address space.
- Programming starts as soon as the page number and the programming command are written to the Flash Command Register. The FRDY bit in the Flash Programming Status Register (EEFC_FSR) is automatically cleared.
- When programming is completed, the FRDY bit in the Flash Programming Status Register (EEFC_FSR) rises. If an interrupt has been enabled by setting the bit FRDY in EEFC_FMR, the corresponding interrupt line of the NVIC is activated.

Two errors can be detected in the EEFC_FSR register after a programming sequence:

- a Command Error: a bad keyword has been written in the EEFC_FCR register.
- a Lock Error: the page to be programmed belongs to a locked region. A command must be previously run to unlock the corresponding region.

24.7.10 SDRAMC Configuration 1 Register

Name: SDRAMC_CR1

Address: 0x400E0228

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
–	–	–	–	TMRD			

- **TMRD: Load Mode Register Command to Active or Refresh Command**

Reset Value is 2 cycles.

This field defines the delay between a Load mode register command and an active or refresh command in number of cycles. Number of cycles is between 0 and 15.

- **NFSEL: NAND Flash Selection**

If this bit is set to one, the chip select is assigned to NAND Flash write enable and read enable lines drive the Error Correcting Code module.

26.5.6 Receive Next Counter Register

Name: PERIPH_RNCR

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**
RXNCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXNCTR							

- **RXNCTR: Receive Next Counter**

RXNCTR contains next receive buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

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

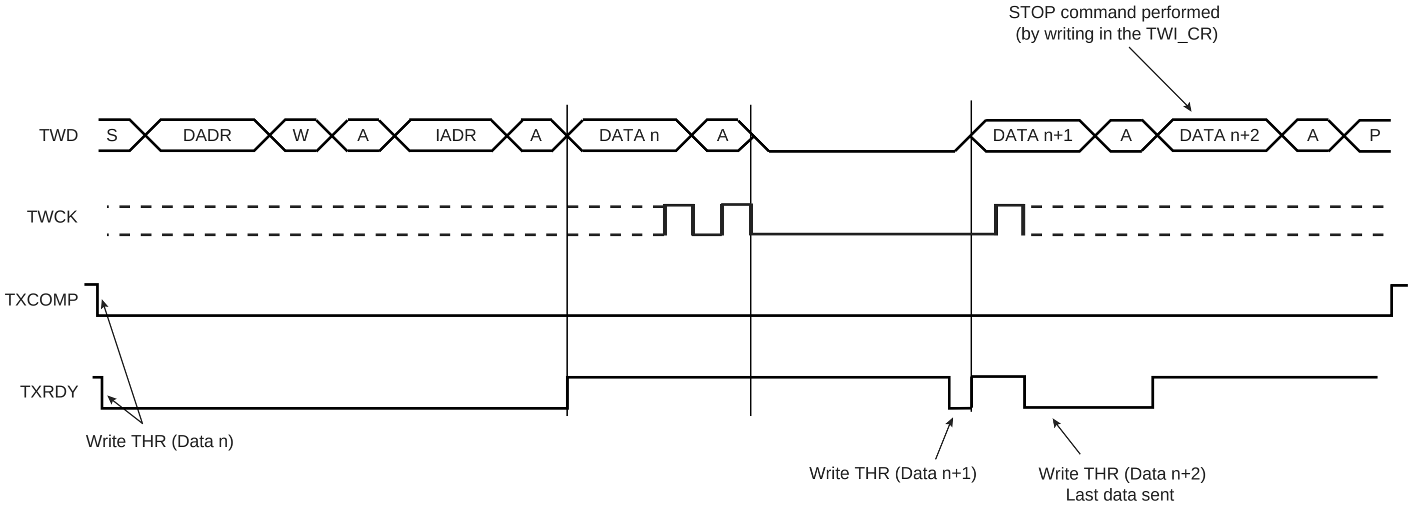
- **WPKEY: Write Protect KEY**

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.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

Figure 33-8. Master Write with One Byte Internal Address and Multiple Data Bytes



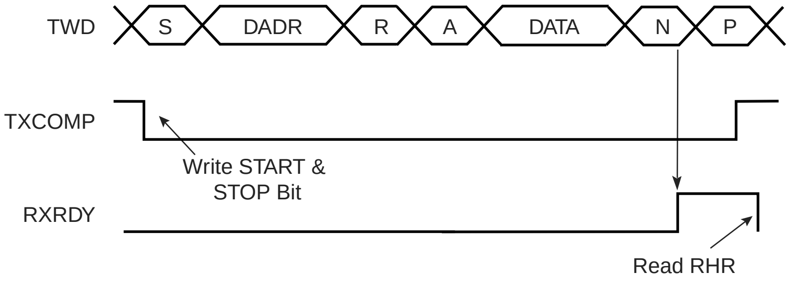
33.8.5 Master Receiver Mode

The read sequence begins by setting the START bit. After the start condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction, 1 in this case (MREAD = 1 in TWI_MMR). 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. The master polls the data line during this clock pulse and sets the **NACK** bit in the status register if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data, after the stop condition. See Figure 33-9. When the RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI_RHR). The RXRDY bit is reset when reading the TWI_RHR.

When a single data byte read is performed, with or without internal address **(IADR)**, the START and STOP bits must be set at the same time. See Figure 33-9. When a multiple data byte read is performed, with or without internal address **(IADR)**, the STOP bit must be set after the next-to-last data received. See Figure 33-10. For Internal Address usage see Section 33.8.6.

Figure 33-9. Master Read with One Data Byte



33.11.9 TWI Interrupt Mask Register

Name: TWI_IMR

Address: 0x4008C02C (0), 0x4009002C (1)

Access: Read-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 Mask**

- **RXRDY: Receive Holding Register Ready Interrupt Mask**

- **TXRDY: Transmit Holding Register Ready Interrupt Mask**

- **SVACC: Slave Access Interrupt Mask**

- **GACC: General Call Access Interrupt Mask**

- **OVRE: Overrun Error Interrupt Mask**

- **NACK: Not Acknowledge Interrupt Mask**

- **ARBLST: Arbitration Lost Interrupt Mask**

- **SCL_WS: Clock Wait State Interrupt Mask**

- **EOSACC: End Of Slave Access Interrupt Mask**

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

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

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

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

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

34.5 UART Operations

The UART operates in asynchronous mode only and supports only 8-bit character handling (with parity). It has no clock pin.

The UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

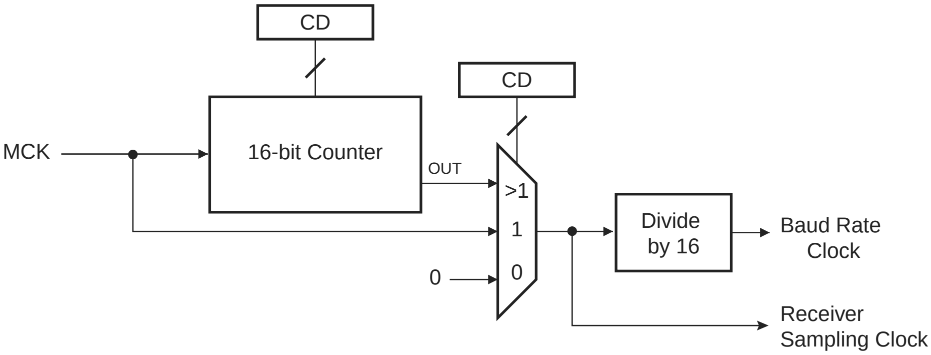
34.5.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in UART_BRGR (Baud Rate Generator Register). If UART_BRGR is set to 0, the baud rate clock is disabled and the UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

$$\text{Baud Rate} = \frac{MCK}{16 \times CD}$$

Figure 34-2. Baud Rate Generator



34.5.2 Receiver

34.5.2.1 Receiver Reset, Enable and Disable

After device reset, the UART receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register UART_CR with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing UART_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing UART_CR with the bit RSTRX at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

–SRC_WIDTH is set to BYTE.

–SCSIZE must be set according to the value of HSMCI_DMA, CHKSIZE field.

–BTSIZE is programmed with *block_length[1:0].* (last 1, 2, or 3 bytes of the buffer).

o. Program LLI_B(n).DMAC_CTRLBx with the following field’s values:
   - DST_INCR is set to INCR.
   - SRC_INCR is set to INCR.
   - FC field is programmed with peripheral to memory flow control mode.
   - Both SRC_DSCR and DST_DSCR are set to 1 (descriptor fetch is disabled) or Next descriptor location points to 0.
   - DIF and SIF are set with their respective layer ID. If SIF is different from DIF, the DMA Controller is able to prefetch data and write HSMCI simultaneously.

p. Program LLI_B(n).DMAC_CFGx memory location for channel x with the following field’s values:
   - FIFOCFG defines the watermark of the DMAC channel FIFO.
   - SRC_H2SEL is set to true to enable hardware handshaking on the destination.
   - SRC_PER is programmed with the hardware handshaking ID of the targeted HSMCI Host Controller

q. Program LLI_B(n).DMAC_DSCR with address of descriptor LLI_W(n+1). If LLI_B(n) is the last descriptor, then program LLI_B(n).DMAC_DSCR with 0.

r. Program DMAC_CTRLBx register for channel x with 0, its content is updated with the LLI Fetch operation.

s. Program DMAC_DSCRx with the address of LLI_W(0) if *block_length* is greater than 4 else with address of LLI_B(0).

t. Enable Channel x writing one to DMAC_CHER[x]. The DMAC is ready and waiting for request.

4. Enable DMADONE interrupt in the HSMCI_IER register.
5. Poll CBTC[x] bit in the DMAC_EBCISR Register.
6. If a new list of buffers shall be transferred, repeat step 7. Check and handle HSMCI errors.
7. Poll FIFOEMPTY field in the HSMCI_SR.
8. Send The STOP_TRANSMISSION command writing HSMCI_ARG then HSMCI_CMDR.
9. Wait for XFRDONE in HSMCI_SR register.

37.14.7 HSMCI Block Register

Name: HSMCI_BLKR

Address: 0x40000018

Access: Read-write

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

- **BCNT: MMC/SDIO Block Count - SDIO Byte Count**

This field determines the number of data byte(s) or block(s) to transfer.

The transfer data type and the authorized values for BCNT field are determined by the TRTYP field in the HSMCI Command Register (HSMCI_CMDR):

Value	Name	Description
0	MULTIPLE	MMC/SDCARD Multiple Block From 1 to 65635: Value 0 corresponds to an infinite block transfer.
4	BYTE	SDIO Byte From 1 to 512 bytes: Value 0 corresponds to a 512-byte transfer. Values from 0x200 to 0xFFFF are forbidden.
5	BLOCK	SDIO Block From 1 to 511 blocks: Value 0 corresponds to an infinite block transfer. Values from 0x200 to 0xFFFF are forbidden.

Warning: In SDIO Byte and Block modes, writing to the 7 last bits of BCNT field is forbidden and may lead to unpredictable results.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

This field is also accessible in the HSMCI Mode Register (HSMCI_MR).

Bits 16 and 17 must be set to 0 if FBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

37.14.20 HSMCI FIFOx Memory Aperture

Name: HSMCI_FIFOx[x=0..255]

Address: 0x40000200

Access: Read-write

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

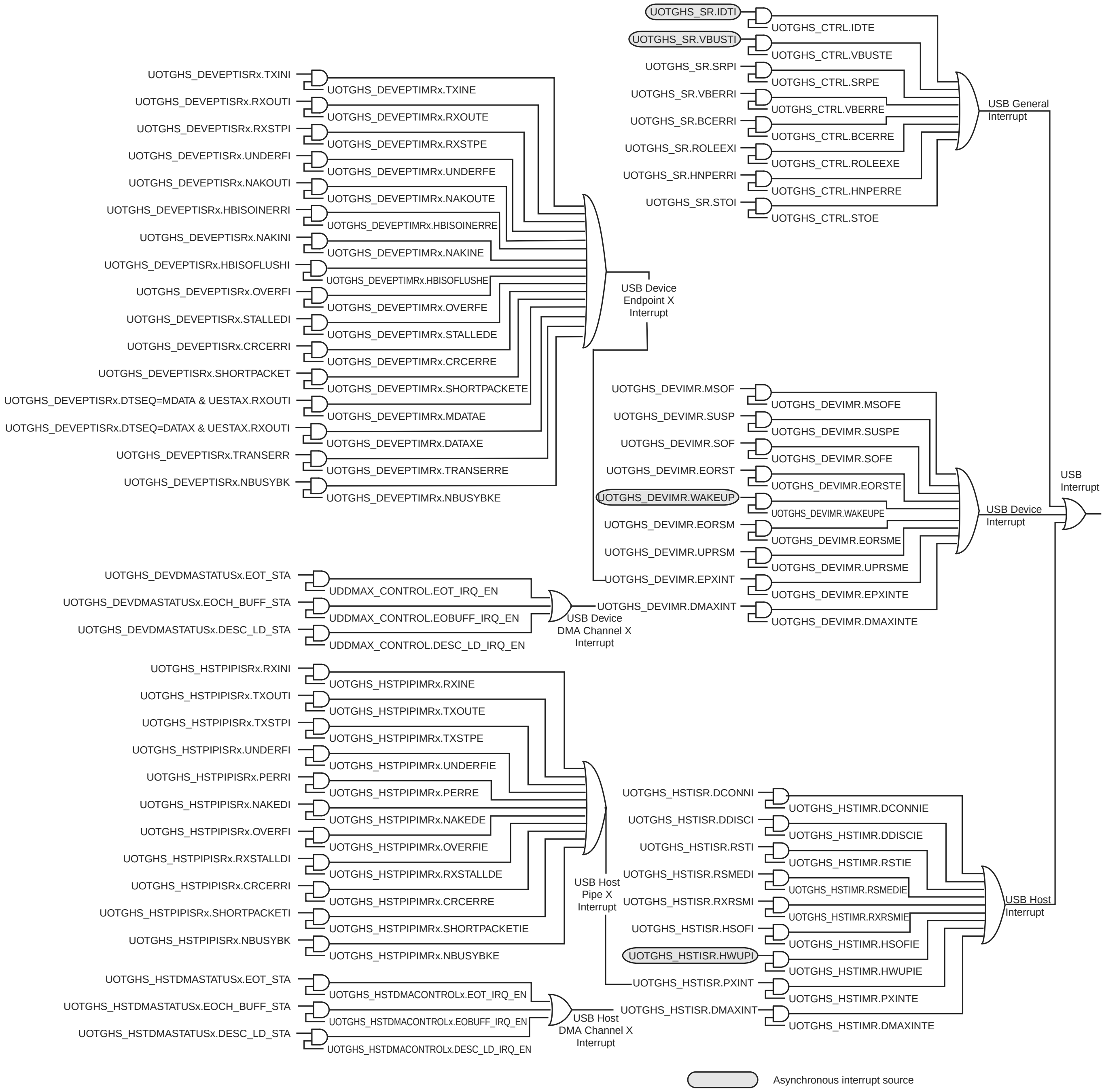
- **DATA: Data to Read or Data to Write**

The UOTGHS can be disabled at any time by writing a zero to UOTGHS_CTRL.USBE. In fact, writing a zero to UOTGHS_CTRL.USBE acts as a hardware reset, except that the UOTGHS_CTRL.OTGPADE, UOTGHS_CTRL.VBUSPO, UOTGHS_CTRL.FRZCLK, UOTGHS_CTRL.UIDE, UOTGHS_CTRL.UIMOD and, UOTGHS_DEVCTRL.LS bits are not reset.

39.5.1.3 Interrupts

One interrupt vector is assigned to the USB interface. Figure 39-6 on page 1059 shows the structure of the USB interrupt system.

Figure 39-6. Interrupt System



See Section 39.5.2.19 and Section 39.5.3.13 for further details about device and host interrupts.

- **MSOFS: Micro Start of Frame Interrupt Set**

Writing a one to this bit will set MSOF bit in UOTGHS_DEVISR, which may be useful for test or debug purposes.

Writing a zero to this bit has no effect.

This bit always reads as zero.

- **SUSPS: Suspend Interrupt Set**

Writing a one to this bit will set SUSP bit in UOTGHS_DEVISR, which may be useful for test or debug purposes.

Writing a zero to this bit has no effect.

This bit always reads as zero.

39.6.3.22 Host DMA Channel x Next Descriptor Address Register

Name: UOTGHS_HSTDMANXTDSCx [x=1..6]

Address: 0x400AC710 [1], 0x400AC720 [2], 0x400AC730 [3], 0x400AC740 [4], 0x400AC750 [5], 0x400AC760 [6], 0x400AC770 [7]

Access: Read-write

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

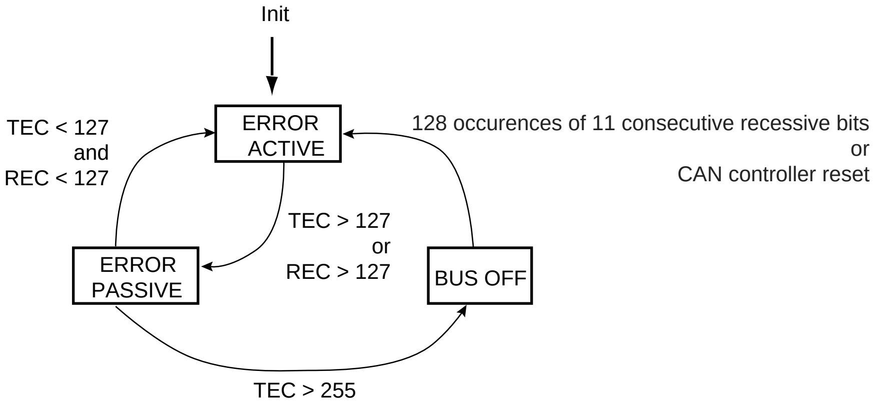
- **NXT_DSC_ADD: Next Descriptor Address**

This field points to the next channel descriptor to be processed. This channel descriptor must be aligned, so bits 0 to 3 of the address must be equal to zero.

Fault Confinement

To distinguish between temporary and permanent failures, every CAN controller has two error counters: REC (Receive Error Counter) and TEC (Transmit Error Counter). The two counters are incremented upon detected errors and are decremented upon correct transmissions or receptions, respectively. Depending on the counter values, the state of the node changes: the initial state of the CAN controller is Error Active, meaning that the controller can send Error Active flags. The controller changes to the Error Passive state if there is an accumulation of errors. If the CAN controller fails or if there is an extreme accumulation of errors, there is a state transition to Bus Off.

Figure 40-7. Line Error Mode



An error active unit takes part in bus communication and sends an active error frame when the CAN controller detects an error.

An error passive unit cannot send an active error frame. It takes part in bus communication, but when an error is detected, a passive error frame is sent. Also, after a transmission, an error passive unit waits before initiating further transmission.

A bus off unit is not allowed to have any influence on the bus.

For fault confinement, two errors counters (TEC and REC) are implemented. These counters are accessible via the CAN_ECR register. The state of the CAN controller is automatically updated according to these counter values. If the CAN controller is in Error Active state, then the ERRA bit is set in the CAN_SR register. The corresponding interrupt is pending while the interrupt is not masked in the CAN_IMR register. If the CAN controller is in Error Passive Mode, then the ERRP bit is set in the CAN_SR register and an interrupt remains pending while the ERRP bit is set in the CAN_IMR register. If the CAN is in Bus Off Mode, then the BOFF bit is set in the CAN_SR register. As for ERRP and ERRA, an interrupt is pending while the BOFF bit is set in the CAN_IMR register.

When one of the error counters values exceeds 96, an increased error rate is indicated to the controller through the WARN bit in CAN_SR register, but the node remains error active. The corresponding interrupt is pending while the interrupt is set in the CAN_IMR register.

Refer to the Bosch CAN specification v2.0 for details on fault confinement.

Error Interrupt Handler

WARN, BOFF, ERRA and ERRP (CAN_SR) represent the current status of the CAN bus and are not latched. They reflect the current TEC and REC (CAN_ECR) values as described in Section "Fault Confinement" on page 1196.

Based on that, if these bits are used as an interrupt, the user can enter into an interrupt and not see the corresponding status register if the TEC and REC counter have changed their state. When entering Bus Off Mode, the only way to exit from this state is 128 occurrences of 11 consecutive recessive bits or a CAN controller reset.

43.7.12 ADC Interrupt Status Register

Name: ADC_ISR

Address: 0x400C0030

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	RXBUFF	ENDRX	COMPE	GOVRE	DRDY
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EOC15	EOC14	EOC13	EOC12	EOC11	EOC10	EOC9	EOC8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished. This flag is cleared when reading the corresponding ADC_CDRx registers.

1 = Corresponding analog channel is enabled and conversion is complete.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_ISR.

1 = At least one General Overrun Error has occurred since the last read of ADC_ISR.

- **COMPE: Comparison Error**

0 = No Comparison Error since the last read of ADC_ISR.

1 = At least one Comparison Error has occurred since the last read of ADC_ISR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in ADC_RCR or ADC_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in ADC_RCR or ADC_RNCR.

- **RXBUFF: RX Buffer Full**

0 = ADC_RCR or ADC_RNCR have a value other than 0.

1 = Both ADC_RCR and ADC_RNCR have a value of 0.