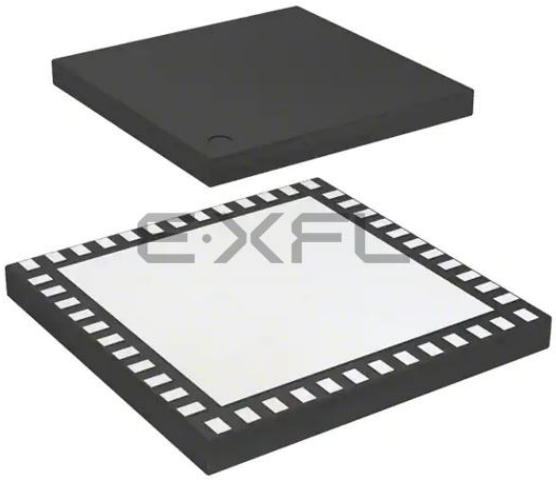


Welcome to E-XFL.COM

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	Obsolete
Core Processor	AVR
Core Size	32-Bit Single-Core
Speed	50MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	36
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-UFLGA Exposed Pad
Supplier Device Package	48-TLLGA (5.5x5.5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3l016-d3ur

4. Processor and Architecture

Rev: 2.1.0.0

This chapter gives an overview of the AVR32UC CPU. AVR32UC is an implementation of the AVR32 architecture. A summary of the programming model, instruction set, and MPU is presented. For further details, see the *AVR32 Architecture Manual* and the *AVR32UC Technical Reference Manual*.

4.1 Features

- **32-bit load/store AVR32A RISC architecture**
  - **15 general-purpose 32-bit registers**
  - **32-bit Stack Pointer, Program Counter and Link Register reside in register file**
  - **Fully orthogonal instruction set**
  - **Privileged and unprivileged modes enabling efficient and secure operating systems**
  - **Innovative instruction set together with variable instruction length ensuring industry leading code density**
  - **DSP extension with saturating arithmetic, and a wide variety of multiply instructions**
- **3-stage pipeline allowing one instruction per clock cycle for most instructions**
  - **Byte, halfword, word, and double word memory access**
  - **Multiple interrupt priority levels**
- **MPU allows for operating systems with memory protection**
- **Secure State for supporting FlashVault technology**

4.2 AVR32 Architecture

AVR32 is a new, high-performance 32-bit RISC microprocessor architecture, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption and high code density. In addition, the instruction set architecture has been tuned to allow a variety of microarchitectures, enabling the AVR32 to be implemented as low-, mid-, or high-performance processors. AVR32 extends the AVR family into the world of 32- and 64-bit applications.

Through a quantitative approach, a large set of industry recognized benchmarks has been compiled and analyzed to achieve the best code density in its class. In addition to lowering the memory requirements, a compact code size also contributes to the core's low power characteristics. The processor supports byte and halfword data types without penalty in code size and performance.

Memory load and store operations are provided for byte, halfword, word, and double word data with automatic sign- or zero extension of halfword and byte data. The C-compiler is closely linked to the architecture and is ableeit 0 T.3(t)-1.(r)5.2(i)2.6byte 5(it 06)5(it .3(w) .2(i)26 5.2(i).ee)5

7.4.4 Peripheral Events

The PDCA peripheral events are connected via the Peripheral Event System. Refer to the Peripheral Event System chapter for details.

7.5 Functional Description

7.5.1 Basic Operation

The PDCA consists of multiple independent PDCA channels, each capable of handling DMA requests in parallel. Each PDCA channels contains a set of configuration registers which must be configured to start a DMA transfer.

In this section the steps necessary to configure one PDCA channel is outlined.

The peripheral to transfer data to or from must be configured correctly in the Peripheral Select Register (PSR). This is performed by writing the Peripheral Identity (PID) value for the corresponding peripheral to the PID field in the PSR register. The PID also encodes the transfer direction, i.e. memory to peripheral or peripheral to memory. See Section 7.5.6.

The transfer size must be written to the Transfer Size field in the Mode Register (MR.SIZE). The size must match the data size produced or consumed by the selected peripheral. See Section 7.5.7.

The memory address to transfer to or from, depending on the PSR, must be written to the Memory Address Register (MAR). For each transfer the memory address is increased by either a one, two or four, depending on the size set in MR. See Section 7.5.2.

The number of data items to transfer is written to the TCR register. If the PDCA channel is enabled, a transfer will start immediately after writing a non-zero value to TCR or the reload version of TCR, TCRR. After each transfer the TCR value is decreased by one. Both MAR and TCR can be read while the PDCA channel is active to monitor the DMA progress. See Section 7.5.3.

The channel must be enabled for a transfer to start. A channel is enable by writing a one to the EN bit in the Control Register (CR).

7.5.2 Memory Pointer

Each channel has a 32-bit Memory Address Register (MAR). This register holds the memory address for the next transfer to be performed. The register is automatically updated after each transfer. The address will be increased by either one, two or four depending on the size of the DMA transfer (byte, halfword or word). The MAR can be read at any time during transfer.

7.5.3 Transfer Counter

Each channel has a 16-bit Transfer Counter Register (TCR). This register must be written with the number of transfers to be performed. The TCR register should contain the number of data items to be transferred independently of the transfer size. The TCR can be read at any time during transfer to see the number of remaining transfers.

7.5.4 Reload Registers

Both the MAR and the TCR have a reload register, respectively Memory Address Reload Register (MARR) and Transfer Counter Reload Register (TCRR). These registers provide the possibility for the PDCA to work on two memory buffers for each channel. When one buffer has completed, MAR and TCR will be reloaded with the values in MARR and TCRR. The reload logic is always enabled and will trigger if the TCR reaches zero while TCRR holds a non-zero value. After reload, the MARR and TCRR registers are cleared.

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: At least one lock region is protected, or BOOTPROT is different from 0. The erase command has been aborted and no page has been erased. A “Unlock region containing given page” (UP) command must be executed to unlock any locked regions.

8.5.3 Region Lock Bits

The flash memory has *p* pages, and these pages are grouped into 16 lock regions, each region containing *p*/16 pages. Each region has a dedicated lock bit preventing writing and erasing pages in the region. After production, the device may have some regions locked. These locked regions are reserved for a boot or default application. Locked regions can be unlocked to be erased and then programmed with another application or other data.

To lock or unlock a region, the commands Lock Region Containing Page (LP) and Unlock Region Containing Page (UP) are provided. Writing one of these commands, together with the number of the page whose region should be locked/unlocked, performs the desired operation.

One error can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that lock bits can also be set/cleared using the commands for writing/erasing general-purpose fuse bits, see Section 8.6. The general-purpose bit being in an erased (1) state means that the region is unlocked.

The lowermost pages in the flash can additionally be protected by the BOOTPROT fuses, see Section 8.6.

8.6 General-purpose Fuse Bits

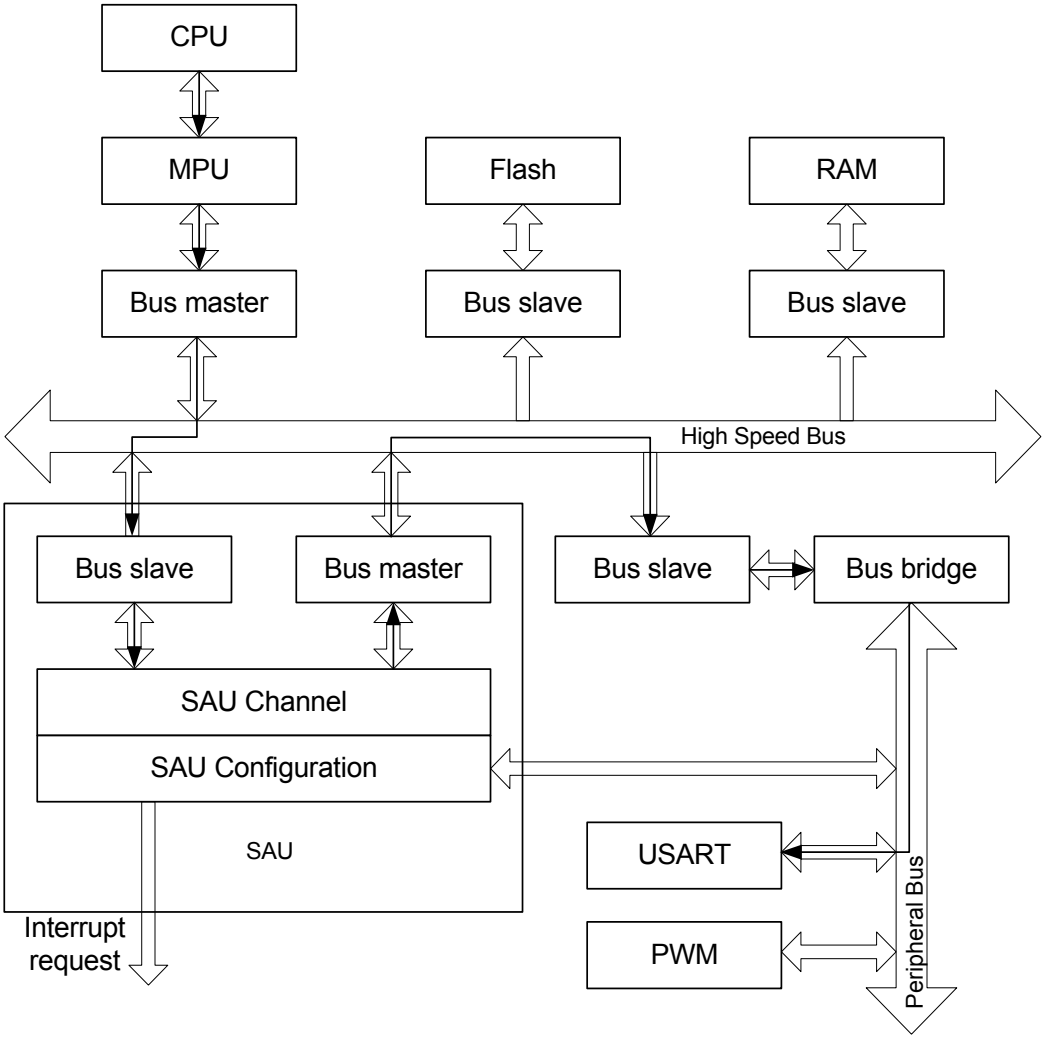
The flash memory has a number of general-purpose fuse bits that the application programmer can use freely. The fuse bits can be written and erased using dedicated commands, and read

9.3 Block Diagram

Figure 9-1 presents the SAU integrated in an example system with a CPU, some memories, some peripherals, and a bus system. The SAU is connected to both the Peripheral Bus (PB) and the High Speed Bus (HSB). Configuration of the SAU is done via the PB, while memory accesses are done via the HSB. The SAU receives an access on its HSB slave interface, remaps it, checks that the channel is unlocked, and if so, initiates a transfer on its HSB master interface to the remapped address.

The thin arrows in Figure 9-1 exemplifies control flow when using the SAU. The CPU wants to read the RX Buffer in the USART. The MPU has been configured to protect all registers in the USART from user mode access, while the SAU has been configured to remap the RX Buffer into a memory space that is not protected by the MPU. This unprotected memory space is mapped into the SAU HSB slave space. When the CPU reads the appropriate address in the SAU, the SAU will perform an access to the desired RX buffer register in the USART, and thereafter return the read results to the CPU. The return data flow will follow the opposite direction of the control flow arrows in Figure 9-1.

Figure 9-1. SAU Block Diagram



- Undefined Length Burst Arbitration

In order to avoid long slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer. A predicted end of burst is used as a defined length burst transfer and can be selected among the following five possibilities:

1. Infinite: No predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. One beat bursts: Predicted end of burst is generated at each single transfer inside the INCP transfer.
3. Four beat bursts: Predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
4. Eight beat bursts: Predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
5. Sixteen beat bursts: Predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the ULBT field in the Master Configuration Registers (MCFG).

- Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (SCFG) and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, halfword, or word transfer.

10.4.2.2 Round-Robin Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave in a round-robin manner. If two or more master requests arise at the same time, the master with the lowest number is first serviced, then the others are serviced in a round-robin manner.

There are three round-robin algorithms implemented:

1. Round-Robin arbitration without default master
2. Round-Robin arbitration with last default master
3. Round-Robin arbitration with fixed default master

- Round-Robin Arbitration without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

- Round-Robin Arbitration with Last Default Master

This is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. At the end of the cur-

12.7.4 PBx Clock Select

Name: PBxSEL

Access Type: Read/Write

Offset: 0x00C-0x010

Reset Value: 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
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
PBDIV	-	-	-	-	PBSEL		

- **PBDIV, PBSEL: PBx Division and Clock Select**
  PBDIV = 0: PBx clock equals main clock.
  PBDIV = 1: PBx clock equals main clock divided by $2^{(PBSEL+1)}$.

Note that if PBDIV is written to 0, PBSEL should also be written to 0 to ensure correct operation.

Also note that writing this register clears SR.CKRDY. The register must not be re-written until SR.CKRDY is set.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

13.5.10 120 MHz RC Oscillator (RC120M)

Rev: 1.0.1.0

The 120 MHz RC Oscillator can be used as source for the main clock in the device, as described in the Power Manager chapter. The oscillator can also be used as source for the generic clocks, as described in Generic Clock section. The RC120M must be enabled before it is used as a source clock. To enable the clock, the user must write a one to the Enable bit in the 120 MHz RC Oscillator Control Register (RC120MCR.EN), and read back the RC120MCR register until the EN bit reads one. The clock is disabled by writing a zero to RC120MCR.EN. The EN bit must be read back as zero before the RC120M is re-enabled. If not, undefined behavior may occur.

The oscillator is automatically disabled in certain sleep modes to reduce power consumption, as described in the Power Manager chapter.

13.5.11 Backup Registers (BR)

Rev: 1.0.0.1

Four 32-bit backup registers are available to store values when the device is in Shutdown mode. These registers will keep their content even when the VDDCORE supply and the internal regulator supply voltage supplies are removed. The backup registers can be accessed by reading from and writing to the BR0, BR1, BR2, and BR3 registers.

After writing to one of the backup registers the user must wait until the Backup Register Interface Ready bit in tne Power and Clocks Status Register (PCLKSR.BRIFARDY) is set before writing to another backup register. Writes to the backup register while PCLKSR.BRIFARDY is zero will be discarded. An interrupt can be generated on a zero-to-one transition on PCLKSR.BRIFARDY if the BRIFARDY bit in the Interrupt Mask Register (IMR.BRIFARDY) is set. This bit is set by writing a one to the corresponding bit in the Interrupt Enable Register (IER.BRIFARDY).

After powering up the device the Backup Register Interface Valid bit in PCLKSR (PCLKSR.BRIFAVALID) is cleared, indicating that the content of the backup registers has not been written and contains the reset value. After writing to one of the backup registers the PCLKSR.BRIFAVALID bit is set. During writes to the backup registers (when BRIFARDY is zero) BRIFAVALID will be zero. If a reset occurs when BRIFARDY is zero, BRIFAVALID will be cleared after the reset, indicating that the content of the backup registers is not valid. If BRIFARDY is one when a reset occurs, BRIFAVALID will be one and the content is the same as before the reset.

The user must ensure that BRIFAVALID and BRIFARDY are both set before reading the backup register values.

13.5.12 32 kHz RC Oscillator (RC32K)

Rev: 1.0.0.0

The RC32K can be used as source for the generic clocks, as described in The Generic Clocks section.

The 32 kHz RC oscillator (RC32K) is forced on after reset, and output on PA20. The clock is available on the pad until the PPCR.FRC32 bit in the Power Manager has been cleared or a different peripheral function has been chosen on PA20 (PA20 will start with peripheral function F by default). Note that the forcing will only enable the clock output. To be able to use the RC32K normally the oscillator must be enabled as described below.

13.6.20 Temperature Sensor Configuration Register

Name: TSENS

Access Type: Read/Write

Reset Value: 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
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-			-	-	EN

- **EN: Temperature Sensor Enable**
  0: The Temperature Sensor is disabled.
  1: The Temperature Sensor is enabled.

Note that this register is protected by a lock. To write to this register the UNLOCK register has to be written first. Please refer to the UNLOCK register description for details.

14.5.1.2 Changing the source clock

The CLK_AST_PRSC must be disabled before switching to another source clock. The Clock Busy bit in the Status Register (SR.CLKBUSY) indicates whether the clock is busy or not. This bit is set when the CEN bit in the CLOCK register is changed, and cleared when the CLOCK register can be changed.

To change the clock:

- Write a zero to CLOCK.CEN to disable the clock, without changing CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero
- Write the selected value to CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero
- Write a one to CLOCK.CEN to enable the clock, without changing the CLOCK.CSSEL
- Wait until SR.CLKBUSY reads as zero

14.5.2 Basic Operation

14.5.2.1 Prescaler

When the AST is enabled, the 32-bit prescaler will increment on the rising edge of CLK_AST_PRSC. The prescaler value cannot be read or written, but it can be reset by writing a one to the Prescaler Clear bit in the Control Register (CR.PCLR).

The Prescaler Select field in the Control Register (CR.PSEL) selects the prescaler bit PSEL as source clock for the counter (CLK_AST_CNT). This results in a counter frequency of:

$$f_{CNT} = \frac{f_{PRSC}}{2^{PSEL+1}}$$

where f_{PRSC} is the frequency of the internal prescaler clock CLK_AST_PRSC.

14.5.2.2 Counter operation

When enabled, the AST will increment on every 0-to-1 transition of the selected prescaler tapping. When the Calender bit in the Control Register (CR.CAL) is zero, the counter operates in counter mode. It will increment until it reaches the top value of 0xFFFFFFFF, and then wrap to 0x00000000. This sets the status bit Overflow in the Status Register (SR.OVF). Optionally, the counter can also be reset when an alarm occurs (see Section 14.5.3.2 on page 255. This will also set the OVF bit.

The AST counter value can be read from or written to the Counter Value (CV) register. Note that due to synchronization, continuous reading of the CV register with the lowest prescaler setting will skip every third value. In addition, if CLK_AST_PRSC is as fast as, or faster than, the CLK_AST, the prescaler value must be 3 or higher to be able to read the CV without skipping values.

14.6.1 Control Register

Name: CR

Access Type: Read/Write

Offset: 0x00

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	PSEL				

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CA1	CA0

7	6	5	4	3	2	1	0
-	-	-	-	-	CAL	PCLR	EN

When the SR.BUSY bit is set, writes to this register will be discarded and this register will read as zero.

- **PSEL: Prescaler Select**
  Selects prescaler bit PSEL as source clock for the counter.
- **CAn: Clear on Alarm n**
  0: The corresponding alarm will not clear the counter.
  1: The corresponding alarm will clear the counter.
- **CAL: Calendar Mode**
  0: The AST operates in counter mode.
  1: The AST operates in calendar mode.
- **PCLR: Prescaler Clear**
  Writing a zero to this bit has no effect.
  Writing a one to this bit clears the prescaler.
  This bit always reads as zero.
- **EN: Enable**
  0: The AST is disabled.
  1: The AST is enabled.

16. External Interrupt Controller (EIC)

Rev: 3.0.1.0

16.1 Features

- **Dedicated interrupt request for each interrupt**
- **Individually maskable interrupts**
- **Interrupt on rising or falling edge**
- **Interrupt on high or low level**
- **Asynchronous interrupts for sleep modes without clock**
- **Filtering of interrupt lines**
- **Non-Maskable NMI interrupt**

16.2 Overview

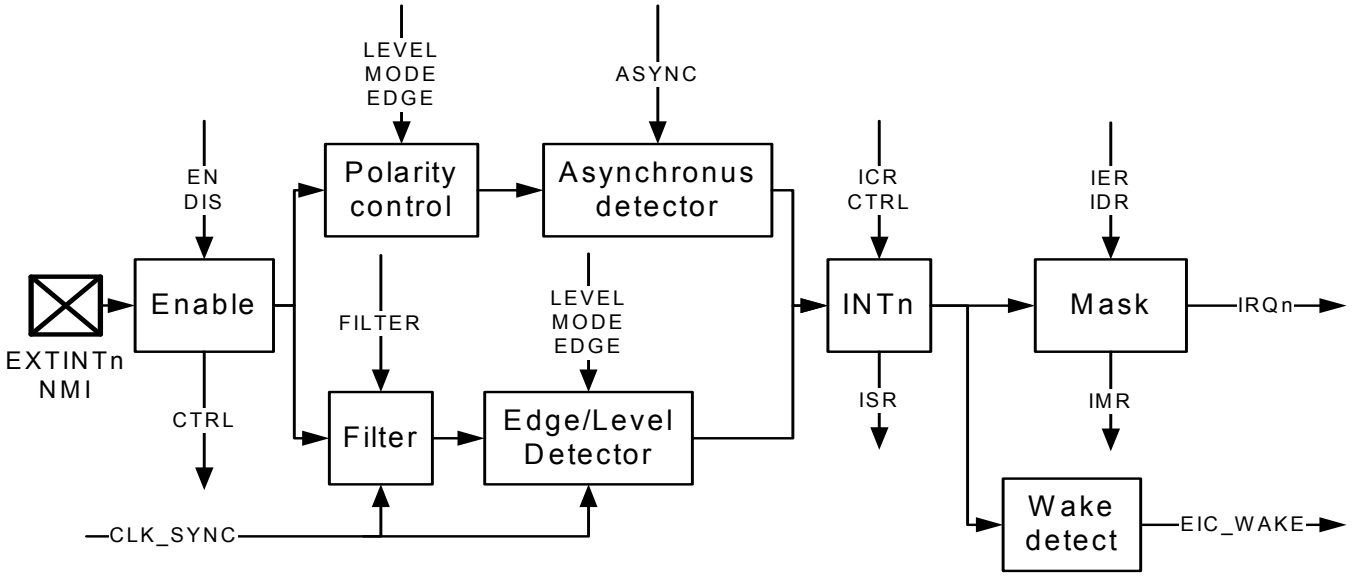
The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked. Each external interrupt can generate an interrupt on rising or falling edge, or high or low level. Every interrupt input has a configurable filter to remove spikes from the interrupt source. Every interrupt pin can also be configured to be asynchronous in order to wake up the part from sleep modes where the CLK_SYNC clock has been disabled.

A Non-Maskable Interrupt (NMI) is also supported. This has the same properties as the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

The EIC can wake up the part from sleep modes without triggering an interrupt. In this mode, code execution starts from the instruction following the sleep instruction.

16.3 Block Diagram

Figure 16-1. EIC Block Diagram



19.6 Functional Description

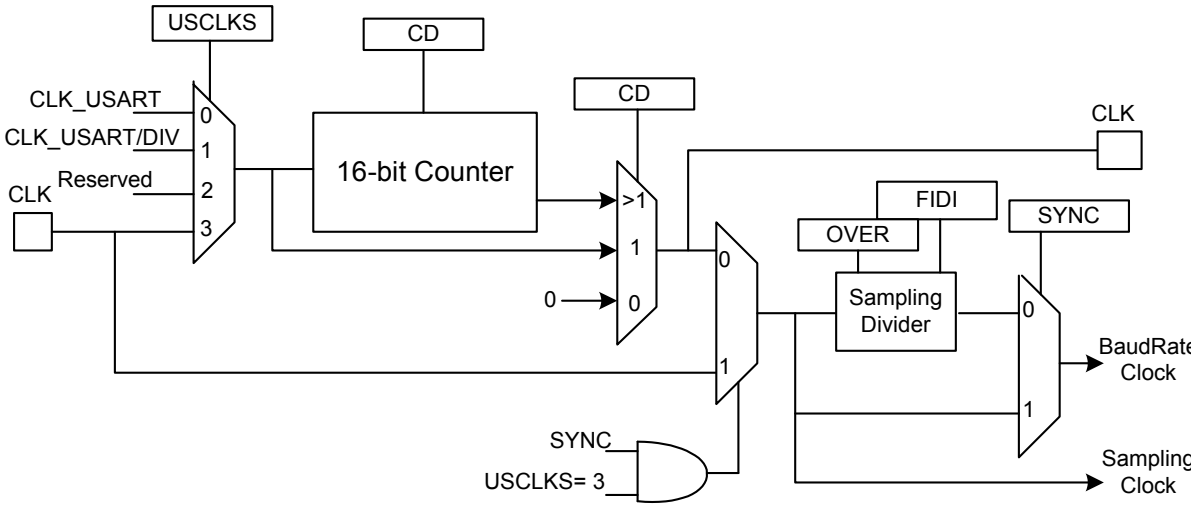
19.6.1 Baud Rate Generator

The baud rate generator provides the bit period clock named the Baud Rate Clock to both receiver and transmitter. It is based on a 16-bit divider, which is specified in the Clock Divider field in the Baud Rate Generator Register (BRGR.CD). A non-zero value enables the generator, and if CD is one, the divider is bypassed and inactive. The Clock Selection field in the Mode Register (MR.USCLKS) selects clock source between:

- CLK_USART (internal clock)
- CLK_USART/DIV (a divided CLK_USART, refer to Module Configuration section)
- CLK (external clock, available on the CLK pin)

If the external CLK clock is selected, the duration of the low and high levels of the signal provided on the CLK pin must be at least 4.5 times longer than those provided by CLK_USART.

Figure 19-2. Baud Rate Generator



19.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is configured to operate in an asynchronous mode, the selected clock is divided by the CD value before it is provided to the receiver as a sampling clock. Depending on the Oversampling Mode bit (MR.OVER) value, the clock is then divided by either 8 (OVER=1), or 16 (OVER=0). The baud rate is calculated with the following formula:

$$BaudRate = \frac{SelectedClock}{(8(2 - OVER)CD)}$$

This gives a maximum baud rate of CLK_USART divided by 8, assuming that CLK_USART is the fastest clock possible, and that OVER is one.

19.6.1.2 *Baud Rate Calculation Example*

Table 19-3 shows calculations based on the CD field to obtain 38400 baud from different source clock frequencies. This table also shows the actual resulting baud rate and error.

23.7.10 Parameter Register

Name: PARAMETER

Access Type: Read-only

Offset: 0x24

Reset Value: -

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
CHANNELS							

- **CHANNELS: Channels Implemented**
  This field contains the number of channels implemented on the device.

Table 27-2. I/O Line Description

Pin Name	Pin Description	Type
ACBPn	Positive reference pin for Analog Comparator B n	Analog
ACBNn	Negative reference pin for Analog Comparator B n	Analog
ACREFN	Reference Voltage for all comparators selectable for INN	Analog

The signal names corresponds to the groups A and B of analog comparators. For normal mode, the mapping from input signal names in the block diagram to the signal names is given in Table 27-3.

Table 27-3. Signal Name Mapping

Pin Name	Channel Number	Normal Mode
ACAP0/ACAN0	0	ACP0/ACN0
ACBP0/ACBN0	1	ACP1/ACN1
ACAP1/ACAN1	2	ACP2/ACN2
ACBP1/ACBN1	3	ACP3/ACN3
ACAP2/ACAN2	4	ACP4/ACN4
ACBP2/ACBN2	5	ACP5/ACN5
ACAP3/ACAN3	6	ACP6/ACN6
ACBP3/ACBN3	7	ACP7/ACN7

27.5 Product Dependencies

In order to use this module, other parts of the system must be configured correctly, as described below.

27.5.1 I/O Lines

The ACIFB pins are multiplexed with other peripherals. The user must first program the I/O Controller to give control of the pins to the ACIFB.

27.5.2 Power Management

If the CPU enters a sleep mode that disables clocks used by the ACIFB, the ACIFB will stop functioning and resume operation after the system wakes up from sleep mode.

27.5.3 Clocks

The clock for the ACIFB bus interface (CLK_ACIFB) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. It is recommended to disable the ACIFB before disabling the clock, to avoid freezing the ACIFB in an undefined state.

The ACIFB uses a GCLK as clock source for the Analog Comparators. The user must set up this GCLK at the right frequency. The CLK_ACIFB clock of the interface must be at least 4x the GCLK frequency used in the comparators. The GCLK is used both for measuring the startup time of a comparator, and to give a frequency for the comparisons done in Continuous Measurement Mode, see Section 27.6.

Refer to the Electrical Characteristics chapter for GCLK frequency limitations.

28.7.24 CSB Resistor Control Register

Name: CSBRES

Access Type: Read/Write

Offset: 0x68

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-							

23	22	21	20	19	18	17	16
-							RES[16]

15	14	13	12	11	10	9	8
RES[15:8]							

7	6	5	4	3	2	1	0
RES[7:0]							

- **RES: Resistive Drive Enable**
  When RES[n] is 0, CSB[n] has the same drive properties as normal I/O pads.
  When RES[n] is 1, CSB[n] has a nominal output resistance of 1kOhm during the burst phase.

30.7.2 Status Register

Name: SR

Access Type: Read-only

Offset: 0x04

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

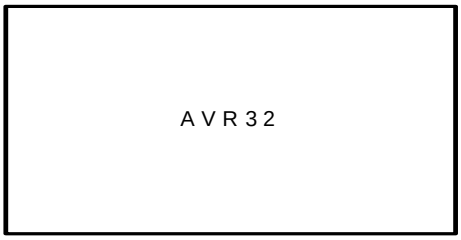
15	14	13	12	11	10	9	8
-	-	TRMIS	-	-	OVERRUN	DREADYINT	READYINT

7	6	5	4	3	2	1	0
-	-	-	-	-	CENABLED	-	BUSY

- **TRMIS: Transmit Mismatch**
  0: No transfers mismatches.
  1: The transceiver was active when receiving.
  This bit is set when the transceiver is active when receiving.
  This bit is cleared when corresponding bit in SCR is written to one.
- **OVERRUN: Data Overrun**
  0: No data overwritten in RHR.
  1: Data in RHR has been overwritten before it has been read.
  This bit is set when data in RHR is overwritten before it has been read.
  This bit is cleared when corresponding bit in SCR is written to one.
- **DREADYINT: Data Ready Interrupt**
  0: No new data in the RHR.
  1: New data received and placed in the RHR.
  This bit is set when new data is received and placed in the RHR.
  This bit is cleared when corresponding bit in SCR is written to one.
- **READYINT: Ready Interrupt**
  0: The interface has not generated an ready interrupt.
  1: The interface has had a transition from busy to not busy.
  This bit is set when the interface has transition from busy to not busy.
  This bit is cleared when corresponding bit in SCR is written to one.

- **CENABLED: Clock Enabled**
  0: The aWire clock is not enabled.
  1: The aWire clock is enabled.

Figure 31-4. AUX+JTAG Based Debugger



31.3.8.1 Trace Operation

Trace features are enabled by writing OCD registers by the debugger. The OCD extracts the trace information from the CPU, compresses this information and formats it into variable-length messages according to the Nexus standard. The messages are buffered in a 16-frame transmit queue, and are output on the AUX port one frame at a time.

The trace features can be configured to be very selective, to reduce the bandwidth on the AUX port. In case the transmit queue overflows, error messages are produced to indicate loss of data. The transmit queue module can optionally be configured to halt the CPU when an overflow occurs, to prevent the loss of messages, at the expense of longer run-time for the program.

31.3.8.2 Program Trace

Program trace allows the debugger to continuously monitor the program execution in the CPU. Program trace messages are generated for every branch in the program, and contains compressed information, which allows the debugger to correlate the message with the source code to identify the branch instruction and target address.

31.3.8.3 Data Trace

Data trace outputs a message every time a specific location is read or written. The message contains information about the type (read/write) and size of the access, as well as the address and data of the accessed location. The AT32UC3L016/32/64 contains two data trace channels,