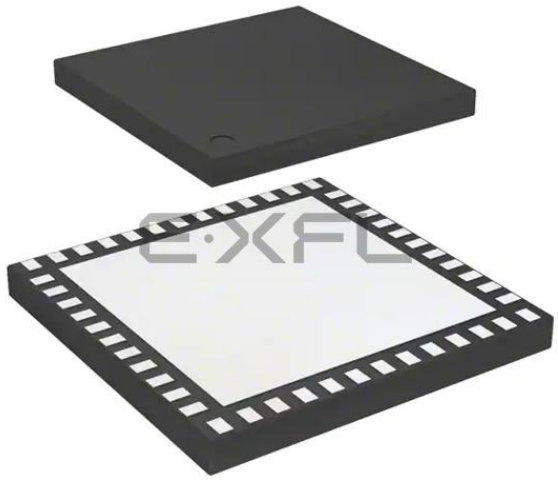


Welcome to E-XFL.COM

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	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	16K 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/at32uc3l032-d3ht

4.4 Programming Model

4.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 4-3. The AVR32UC Register File

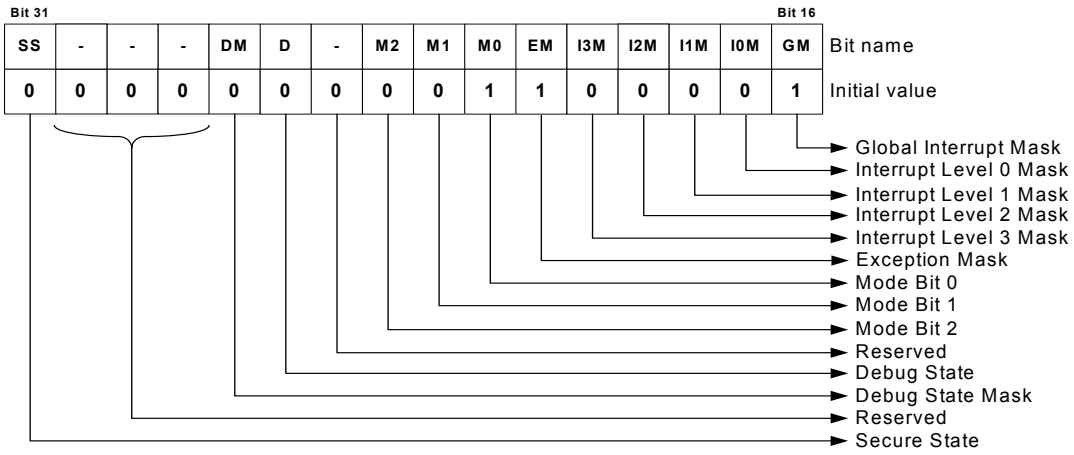
Application	Supervisor	INT0	INT1	INT2	INT3	Exception	NMI	Secure
Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0
PC	PC	PC	PC	PC	PC	PC	PC	PC
LR	LR	LR	LR	LR	LR	LR	LR	LR
SP_APP	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SEC
R12	R12	R12	R12	R12	R12	R12	R12	R12
R11	R11	R11	R11	R11	R11	R11	R11	R11
R10	R10	R10	R10	R10	R10	R10	R10	R10
R9	R9	R9	R9	R9	R9	R9	R9	R9
R8	R8	R8	R8	R8	R8	R8	R8	R8
R7	R7	R7	R7	R7	R7	R7	R7	R7
R6	R6	R6	R6	R6	R6	R6	R6	R6
R5	R5	R5	R5	R5	R5	R5	R5	R5
R4	R4	R4	R4	R4	R4	R4	R4	R4
R3	R3	R3	R3	R3	R3	R3	R3	R3
R2	R2	R2	R2	R2	R2	R2	R2	R2
R1	R1	R1	R1	R1	R1	R1	R1	R1
R0	R0	R0	R0	R0	R0	R0	R0	R0
SR	SR	SR	SR	SR	SR	SR	SR	SR

SS_STATUS
SS_ADRF
SS_ADRR
SS_ADR0
SS_ADR1
SS_SP_SYS
SS_SP_APP
SS_RAR
SS_RSR

4.4.2 Status Register Configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 4-4 and Figure 4-5. The lower word contains the C, Z, N, V, and Q condition code flags and the R, T, and L bits, while the upper halfword contains information about the mode and state the processor executes in. Refer to the *AVR32 Architecture Manual* for details.

Figure 4-4. The Status Register High Halfword



7.7.23 Performance Channel 0 Write Max Latency

Name: PWLAT0

Access Type: Read/Write

Offset: 0x818

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
LAT[15:8]							

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

- **LAT: Maximum Transfer Initiation Cycles Counted Since Last Reset**
  Clock cycles are counted using the CLK_PDCA_HSB clock
  This counter is saturating. The register is reset only when PCONTROL.CH0RES is written to one.

- **FSZ: Flash Size**
  The size of the flash. Not all device families will provide all flash sizes indicated in the table.

Table 8-10. Flash Size

FSZ	Flash Size	FSZ	Flash Size
0	4 Kbyte	8	192 Kbyte
1	8 Kbyte	9	256 Kbyte
2	16 Kbyte	10	384 Kbyte
3	32 Kbyte	11	512 Kbyte
4	48 Kbyte	12	768 Kbyte
5	64 Kbyte	13	1024 Kbyte
6	96 Kbyte	14	2048 Kbyte
7	128 Kbyte	15	Reserved

9. Secure Access Unit (SAU)

Rev: 1.1.0.3

9.1 Features

- **Remaps registers in memory regions protected by the MPU to regions not protected by the MPU**
- **Programmable physical address for each channel**
- **Two modes of operation: Locked and Open**
  - **In Locked Mode, access to a channel must be preceded by an unlock action**
    - **An unlocked channel remains open only for a specific amount of time, if no access is performed during this time, the channel is relocked**
    - **Only one channel can be open at a time, opening a channel while another one is open locks the first one**
    - **Access to a locked channel is denied, a bus error and optionally an interrupt is returned**
    - **If a channel is relocked due to an unlock timeout, an interrupt can optionally be generated**
  - **In Open Mode, all channels are permanently unlocked**

9.2 Overview

In many systems, erroneous access to peripherals can lead to catastrophic failure. An example of such a peripheral is the Pulse Width Modulator (PWM) used to control electric motors. The PWM outputs a pulse train that controls the motor. If the control registers of the PWM module are inadvertently updated with wrong values, the motor can start operating out of control, possibly causing damage to the application and the surrounding environment. However, sometimes the PWM control registers must be updated with new values, for example when modifying the pulse train to accelerate the motor. A mechanism must be used to protect the PWM control registers from inadvertent access caused by for example:

- Errors in the software code
- Transient errors in the CPU caused by for example electrical noise altering the execution path of the program

To improve the security in a computer system, the AVR32UC implements a Memory Protection Unit (MPU). The MPU can be set up to limit the accesses that can be performed to specific memory addresses. The MPU divides the memory space into regions, and assigns a set of access restrictions on each region. Access restrictions can for example be read/write if the CPU is in supervisor mode, and read-only if the CPU is in application mode. The regions can be of different size, but each region is usually quite large, e.g. protecting 1 kilobyte of address space or more. Furthermore, access to each region is often controlled by the execution state of the CPU, i.e. supervisor or application mode. Such a simple control mechanism is often too inflexible (too coarse-grained chunks) and with too much overhead (often requiring system calls to access protected memory locations) for simple or real-time systems such as embedded microcontrollers.

Usually, the Secure Access Unit (SAU) is used together with the MPU to provide the required security and integrity. The MPU is set up to protect regions of memory, while the SAU is set up to provide a secure channel into specific memory locations that are protected by the MPU. These specific locations can be thought of as fine-grained overrides of the general coarse-grained protection provided by the MPU.

12.7.2 CPU Clock Select

Name: CPUSEL

Access Type: Read/Write

Offset: 0x004

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
CPUDIV	-	-	-	-	CPUSEL		

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

Note that if CPUDIV is written to 0, CPUSEL 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 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.

12.7.9 Interrupt Enable Register

Name: IER

Access Type: Write-only

Offset: 0x0C0

Reset Value: 0x00000000

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

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will set the corresponding bit in IMR.

12.7.19 Configuration Register

Name: CONFIG

Access Type: Read-Only

Offset: 0x3F8

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
HSBPEVC	-	-	-	PBD	PBC	PBB	PBA

This register shows the configuration of the PM.

- **HSBPEVC:HSB PEVC Clock Implemented**
  0: HSBPEVC not implemented.
  1: HSBPEVC implemented.
- **PBD: PBD Implemented**
  0: PBD not implemented.
  1: PBD implemented.
- **PBC: PBC Implemented**
  0: PBC not implemented.
  1: PBC implemented.
- **PBB: PBB Implemented**
  0: PBB not implemented.
  1: PBB implemented.
- **PBA: PBA Implemented**
  0: PBA not implemented.
  1: PBA implemented.

13.5.3 Digital Frequency Locked Loop (DFLL) Operation

Rev: 2.0.1.1

The DFLL is controlled by the Digital Frequency Locked Loop Interface (DFLLIF). The DFLL is disabled by default, but can be enabled to provide a high-frequency source clock for synchronous and generic clocks.

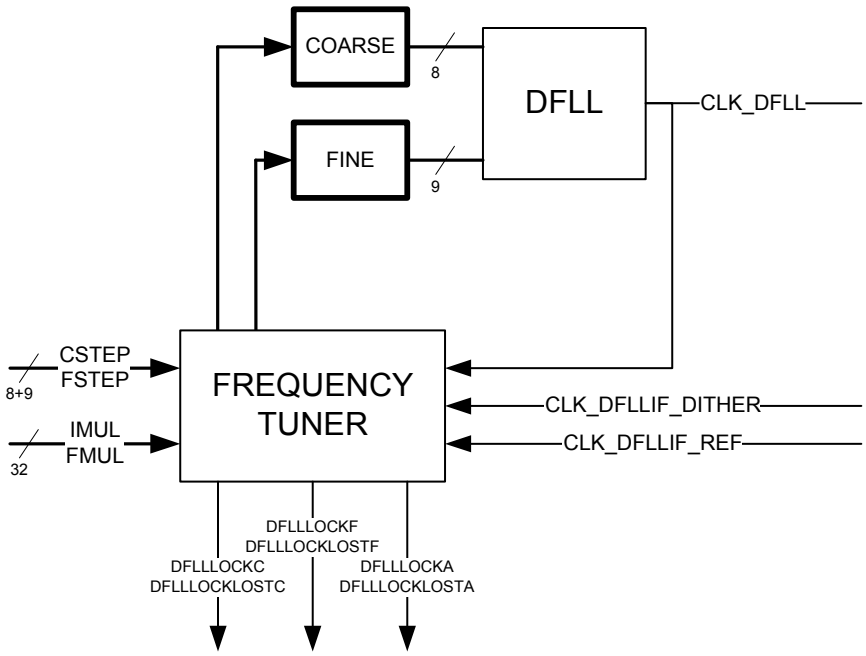
Features:

- Internal oscillator with no external components
- 40-150MHz frequency in closed loop mode
- Can operate standalone as a high-frequency programmable oscillator in open loop mode
- Can operate as an accurate frequency multiplier against a known frequency in closed loop mode
- Optional spread-spectrum clock generation
- Very high-frequency multiplication supported - can generate all frequencies from a 32KHz clock

The DFLL can operate in both open loop mode and closed loop mode. In closed loop mode a low frequency clock with high accuracy can be used as reference clock to get high accuracy on the output clock (CLK_DFLL).

To prevent unexpected writes due to software bugs, write access to the configuration registers is protected by a locking mechanism. For details please refer to the UNLOCK register description.

Figure 13-1. DFLLIF Block Diagram



13.5.3.1 *Enabling the DFLL*

The DFLL is enabled by writing a one to the Enable bit (EN) in the DFLLn Configuration Register (DFLLnCONF). No other bits or fields in DFLLnCONF must be changed simultaneously, or before the DFLL is enabled.

14.5.2.3 Calendar operation

When the CAL bit in the Control Register is one, the counter operates in calendar mode. Before this mode is enabled, the prescaler should be set up to give a pulse every second. The date and time can then be read from or written to the Calendar Value (CALV) register.

Time is reported as seconds, minutes, and hours according to the 24-hour clock format. Date is the numeral date of month (starting on 1). Month is the numeral month of the year (1 = January, 2 = February, etc.). Year is a 6-bit field counting the offset from a software-defined leap year (e.g. 2000). The date is automatically compensated for leap years, assuming every year divisible by 4 is a leap year.

All peripheral events and interrupts work the same way in calendar mode as in counter mode. However, the Alarm Register (ARn) must be written in time/date format for the alarm to trigger correctly.

14.5.3 Interrupts

The AST can generate five separate interrupt requests:

- OVF: OVF
- PER: PER0, PER1
- ALARM: ALARM0, ALARM1
- CLKREADY
- READY

This allows the user to allocate separate handlers and priorities to the different interrupt types.

The generation of the PER interrupt is described in Section 14.5.3.1., and the generation of the ALARM interrupt is described in Section 14.5.3.2. The OVF interrupt is generated when the counter overflows, or when the alarm value is reached, if the Clear on Alarm bit in the Control Register is one. The CLKREADY interrupt is generated when SR.CLKBUSY has a 1-to-0 transition, and indicates that the clock synchronization is completed. The READY interrupt is generated when SR.BUSY has a 1-to-0 transition, and indicates that the synchronization described in Section 14.5.8 is completed.

An interrupt request will be generated if the corresponding bit in the Interrupt Mask Register (IMR) is set. Bits in IMR are set by writing a one to the corresponding bit in the Interrupt Enable Register (IER), and cleared by writing a one to the corresponding bit in the Interrupt Disable Register (IDR). The interrupt request remains active until the corresponding bit in SR is cleared by writing a one to the corresponding bit in the Status Clear Register (SCR).

The AST interrupts can wake the CPU from any sleep mode where the source clock and the interrupt controller is active.

14.5.3.1 Periodic interrupt

The AST can generate periodic interrupts. If the PERn bit in the Interrupt Mask Register (IMR) is one, the AST will generate an interrupt request on the 0-to-1 transition of the selected bit in the

Table 18-2. GPIO Register Memory Map

Offset	Register	Function	Register Name	Access	Reset	Config. Protection	Access Protection
0x0C0	Glitch Filter Enable Register	Read/Write	GFER	Read/Write	-[1]	N	N
0x0C4	Glitch Filter Enable Register	Set	GFERS	Write-only		N	N
0x0C8	Glitch Filter Enable Register	Clear	GFERC	Write-only		N	N
0x0CC	Glitch Filter Enable Register	Toggle	GFERT	Write-only		N	N
0x0D0	Interrupt Flag Register	Read	IFR	Read-only	-[1]	N	N
0x0D4	Interrupt Flag Register	-	-	-		N	N
0x0D8	Interrupt Flag Register	Clear	IFRC	Write-only		N	N
0x0DC	Interrupt Flag Register	-	-	-		N	N
0x180	Event Enable Register	Read	EVER	Read/Write	-[1]	N	N
0x184	Event Enable Register	Set	EVERS	Write-only		N	N
0x188	Event Enable Register	Clear	EVERC	Write-only		N	N
0x18C	Event Enable Register	Toggle	EVERT	Write-only		N	N
0x1A0	Lock Register	Read/Write	LOCK	Read/Write	-[1]	N	Y
0x1A4	Lock Register	Set	LOCKS	Write-only		N	N
0x1A8	Lock Register	Clear	LOCKC	Write-only		N	Y
0x1AC	Lock Register	Toggle	LOCKT	Write-only		N	Y
0x1E0	Unlock Register	Read/Write	UNLOCK	Write-only		N	N
0x1E4	Access Status Register	Read/Write	ASR	Read/Write			N
0x1F8	Parameter Register	Read	PARAMETER	Read-only	-[1]	N	N
0x1FC	Version Register	Read	VERSION	Read-only	-[1]	N	N

Note: 1. The reset values for these registers are device specific. Please refer to the Module Configuration section at the end of this chapter.

21.9.2 Clock Waveform Generator Register

Name: CWGR

Access Type: Read/Write

Offset: 0x04

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	EXP			DATA			

23	22	21	20	19	18	17	16
STASTO							

15	14	13	12	11	10	9	8
HIGH							

7	6	5	4	3	2	1	0
LOW							

- **EXP: Clock Prescaler**
  Used to specify how to prescale the TWCK clock. Counters are prescaled according to the following formula
  $f_{PRESCALER} = \frac{f_{CLK\_TWIM}}{2^{(EXP+1)}}$

- **DATA: Data Setup and Hold Cycles**
  Clock cycles for data setup and hold count. Prescaled by CWGR.EXP. Used to time T_{HD_DAT}, T_{SU_DAT}.
- **STASTO: START and STOP Cycles**
  Clock cycles in clock high count. Prescaled by CWGR.EXP. Used to time T_{HD_STA}, T_{SU_STA}, T_{SU_STO}
- **HIGH: Clock High Cycles**
  Clock cycles in clock high count. Prescaled by CWGR.EXP. Used to time T_{HIGH}.
- **LOW: Clock Low Cycles**
  Clock cycles in clock low count. Prescaled by CWGR.EXP. Used to time T_{LOW}, T_{BUF}.

22.8.7 Wakeup from Sleep Modes by TWI Address Match

The TWIS is able to wake the device up from a sleep mode upon an address match, including sleep modes where CLK_TWIS is stopped. After detecting the START condition on the bus, The TWIS will stretch TWCK until CLK_TWIS has started. The time required for starting CLK_TWIS depends on which sleep mode the device is in. After CLK_TWIS has started, the TWIS releases its TWCK stretching and receives one byte of data on the bus. At this time, only a limited part of the device, including the TWIS, receives a clock, thus saving power. The TWIS goes on to receive the slave address. If the address phase causes a TWIS address match, the entire device is wakened and normal TWIS address matching actions are performed. Normal TWI transfer then follows. If the TWIS is not addressed, CLK_TWIS is automatically stopped and the device returns to its original sleep mode.

22.8.8 Identifying Bus Events

This chapter lists the different bus events, and how these affects the bits in the TWIS registers. This is intended to help writing drivers for the TWIS.

Table 22-5. Bus Events

Event	Effect
Slave transmitter has sent a data byte	SR.THR is cleared. SR.BTF is set. The value of the ACK bit sent immediately after the data byte is given by CR.ACK.
Slave receiver has received a data byte	SR.RHR is set. SR.BTF is set. SR.NAK updated according to value of ACK bit received from master.
Start+Sadr on bus, but address is to another slave	None.
Start+Sadr on bus, current slave is addressed, but address match enable bit in CR is not set	None.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set	Correct address match bit in SR is set. SR.TRA updated according to transfer direction (updating is done one CLK_TWIS cycle after address match bit is set) Slave enters appropriate transfer direction mode and data transfer can commence.
Start+Sadr on bus, current slave is addressed, corresponding address match enable bit in CR set, SR.STREN and SR.SOAM are set.	Correct address match bit in SR is set. SR.TRA updated according to transfer direction (updating is done one CLK_TWIS cycle after address match bit is set). Slave stretches TWCK immediately after transmitting the address ACK bit. TWCK remains stretched until all address match bits in SR have been cleared. Slave enters appropriate transfer direction mode and data transfer can commence.
Repeated Start received after being addressed	SR.REP set. SR.TCOMP unchanged.
Stop received after being addressed	SR.STO set. SR.TCOMP set.

27.9.5 Interrupt Mask Register

Name: IMR

Access Type: Read-only

Offset: 0x18

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	WFINT3	WFINT2	WFINT1	WFINT0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SUTINT7	ACINT7	SUTINT6	ACINT6	SUTINT5	ACINT5	SUTINT4	ACINT4
7	6	5	4	3	2	1	0
SUTINT3	ACINT3	SUTINT2	ACINT2	SUTINT1	ACINT1	SUTINT0	ACINT0

- **WFINT*n*: Window Mode Interrupt Mask**
  0: The corresponding interrupt is disabled.
  1: The corresponding interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.
- **SUTINT*n*: AC*n* Startup Time Interrupt Mask**
  0: The corresponding interrupt is disabled.
  1: The corresponding interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.
- **ACINT*n*: AC*n* Interrupt Mask**
  0: The corresponding interrupt is disabled.
  1: The corresponding interrupt is enabled.
  This bit is cleared when the corresponding bit in IDR is written to one.
  This bit is set when the corresponding bit in IER is written to one.

28.6.4 Autonomous QTouch

For autonomous QTouch, a complete detection algorithm is implemented within the CAT module. The additional parameters needed to control the autonomous QTouch detection algorithm must be specified by the user in the ATCFG2 and ATCFG3 registers.

Autonomous QTouch sensitivity and out-of-touch sensitivity can be adjusted with the SENSE and OUTSENS fields, respectively, in ATCFG2. Each field accepts values from one to 255 where 255 is the least sensitive setting. The value in the OUTSENS field should be smaller than the value in the SENSE field.

To avoid false positives a detect integration filtering technique can be used. The number of successive detects required is specified in the FILTER field of the ATCFG2 register.

To compensate for changes in capacitance the CAT can recalibrate the autonomous QTouch sensor periodically. The timing of this calibration is done with the NDRIFT and PDRIFT fields in the Configuration register, ATCFG3. It is recommended that the PDRIFT value is smaller than the NDRIFT value.

The autonomous QTouch sensor will also recalibrate if the count value goes too far positive beyond a threshold. This positive recalibration threshold is specified by the PTHR field in the ATCFG3 register.

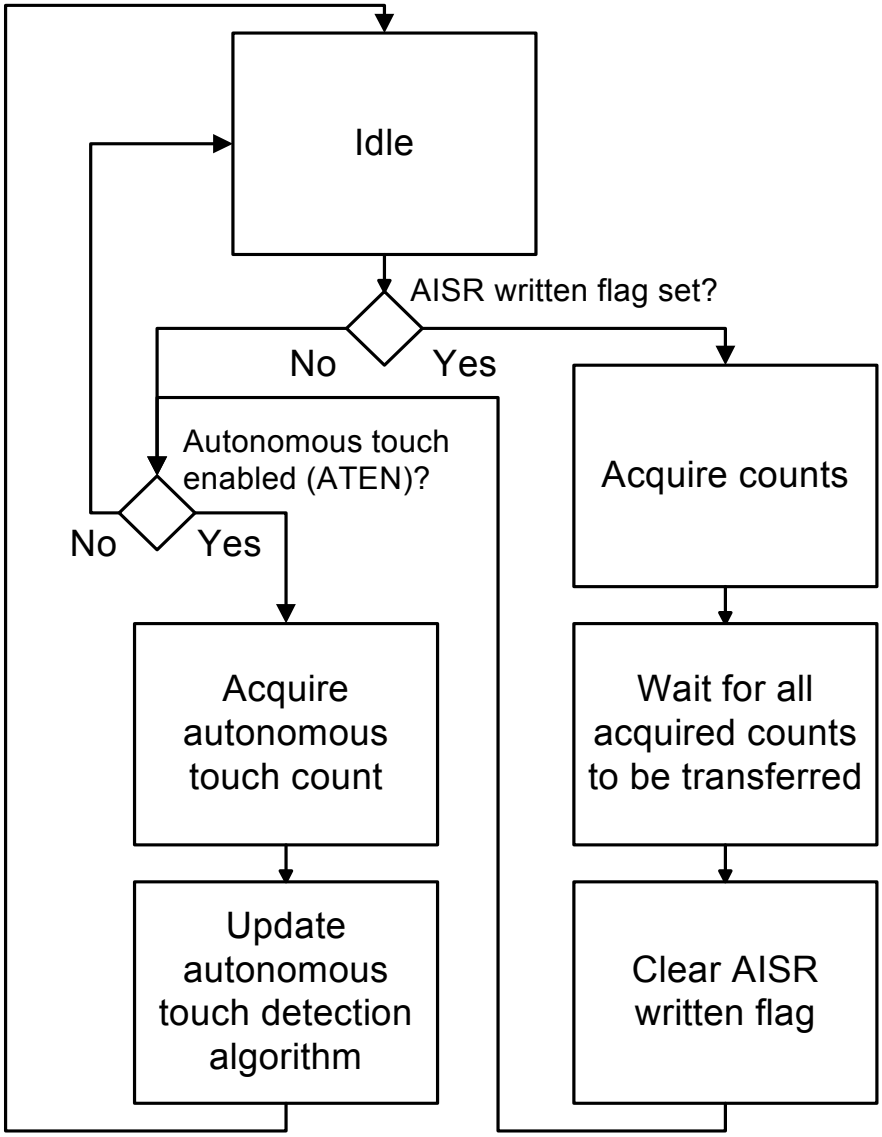
The following block diagram shows the sequence of acquisition and processing operations used by the CAT module. The AISR written bit is internal and not visible in the user interface.

28.6.5 Peripheral Events

The peripheral event from the AST is used to trigger one iteration of autonomous touch detection when the device is in a sleep mode that has disabled CLK_CAT. When CLK_CAT is disabled and the peripheral event from the AST becomes active, a request will automatically be made to enable CLK_CAT. When CLK_CAT is enabled, the CAT will perform one iteration of the autonomous touch detection algorithm. After this, the CLK_CAT will be disabled, and the CAT module will remain in a frozen state until the next AST peripheral event.

The eight peripheral events from the analog comparators are automatically used by the CAT when performing QMatrix acquisition. The CAT will automatically use the negative peripheral events from the AC Interface on every Y pin in QMatrix mode. When QMatrix acquisition is used the analog comparator corresponding to the selected Y pins must be enabled and converting continuously, using the Y pin as the positive reference and the ACREFN as negative reference.

Figure 28-4. CAT Acquisition and Processing Sequence



28.6.6 Spread Spectrum Sensor Drive

To reduce electromagnetic compatibility issues, the capacitive sensors can be driven with a spread spectrum signal. To enable spread spectrum drive for a specific acquisition type, the user must write a one to the SPREAD bit in the appropriate Configuration Register 1 (MGCFG1, ATCFG1, TGACFG1, or TGBCFG1).

During spread spectrum operation, the length of each pulse within a burst is varied in a deterministic pattern, so that the exact same burst pattern is used for a specific burst length. The maximum spread is determined by the MAXDEV field in the Spread Spectrum Configuration Register (SSCFG) register. The prescaler divisor is varied in a sawtooth pattern from (2(DIV+1))-MAXDEV to (2(DIV+1))+MAXDEV and then back to (2(DIV+1))-MAXDEV. For example, if DIV is 2 and MAXDEV is 3, the prescaler divisor will have the following sequence: 6, 7, 8,

Table 31-1. SAB Slaves, Addresses and Descriptions

Slave	Address [35:32]	Description
HSB	0x5	Alternative mapping for HSB space, for compatibility with other 32-bit AVR devices.
Memory Service Unit	0x6	Memory Service Unit registers
Reserved	Other	Unused

31.2.2 SAB Security Restrictions

The Service Access bus can be restricted by internal security measures. A short description of the security measures are found in the table below.

31.2.2.1 Security measure and control location

A security measure is a mechanism to either block or allow SAB access to a certain address or address range. A security measure is enabled or disabled by one or several control signals. This is called the control location for the security measure.

These security measures can be used to prevent an end user from reading out the code programmed in the flash, for instance.

Table 31-2. SAB Security Measures

Security Measure	Control Location	Description
Secure mode	FLASHCDW SECURE bits set	Allocates a portion of the flash for secure code. This code cannot be read or debugged. The User page is also locked.
Security bit	FLASHCDW security bit set	Programming and debugging not possible, very restricted access.
User code programming	FLASHCDW UPROT + security bit set	Restricts all access except parts of the flash and the flash controller for programming user code. Debugging is not possible unless an OS running from the secure part of the flash supports it.

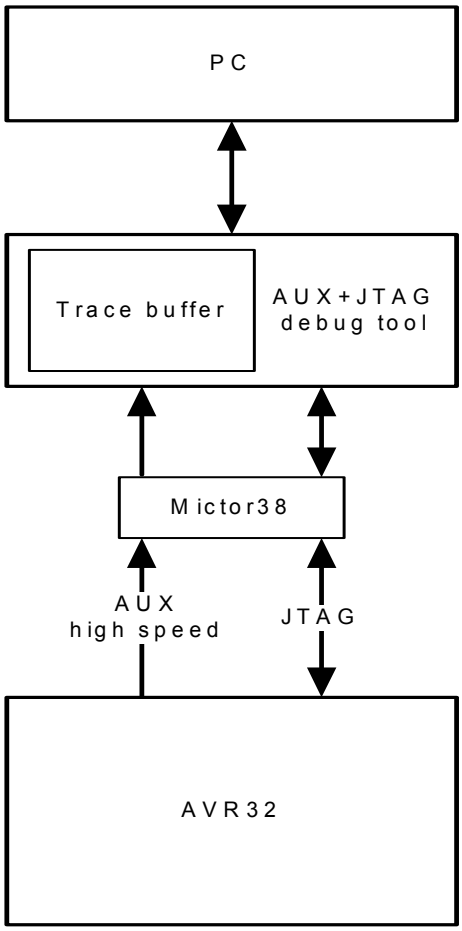
Below follows a more in depth description of what locations are accessible when the security measures are active.

Table 31-3. Secure Mode SAB Restrictions

Name	Address Start	Address End	Access
Secure flash area	0x580000000	0x580000000 + (USERPAGE[15:0] << 10)	Blocked
Secure RAM area	0x500000000	0x500000000 + (USERPAGE[31:16] << 10)	Blocked
User page	0x580800000	0x581000000	Read
Other accesses	-	-	As normal

Note: 1. Second Word of the User Page, refer to the Fuses Settings section for details.

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,

31.5.2.4 INTEST

This instruction selects the boundary-scan chain as Data Register for testing internal logic in the device. The logic inputs are determined by the boundary-scan chain, and the logic outputs are captured by the boundary-scan chain. The device output pins are driven from the boundary-scan chain.

Starting in Run-Test/Idle, the INTEST instruction is accessed the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. In Update-IR: The data from the boundary-scan chain is applied to the internal logic inputs.
5. Return to Run-Test/Idle.
6. Select the DR Scan path.
7. In Capture-DR: The data on the internal logic is sampled into the boundary-scan chain.
8. In Shift-DR: The boundary-scan chain is shifted by the TCK input.
9. In Update-DR: The data from the boundary-scan chain is applied to internal logic inputs.
10. Return to Run-Test/Idle.

Table 31-15. INTEST Details

Instructions	Details
IR input value	**00100** (0x04)
IR output value	p0001
DR Size	Depending on boundary-scan chain, see BSDL-file.
DR input value	Depending on boundary-scan chain, see BSDL-file.
DR output value	Depending on boundary-scan chain, see BSDL-file.

31.5.2.5 CLAMP

This instruction selects the Bypass register as Data Register. The device output pins are driven from the boundary-scan chain.

Starting in Run-Test/Idle, the CLAMP instruction is accessed the following way:

1. Select the IR Scan path.
2. In Capture-IR: The IR output value is latched into the shift register.
3. In Shift-IR: The instruction register is shifted by the TCK input.
4. In Update-IR: The data from the boundary-scan chain is applied to the output pins.
5. Return to Run-Test/Idle.
6. Select the DR Scan path.
7. In Capture-DR: A logic ‘0’ is loaded into the Bypass Register.
8. In Shift-DR: Data is scanned from TDI to TDO through the Bypass register.

35. Errata

35.1 Rev. E

35.1.1 Processor and Architecture

1. **Hardware breakpoints may corrupt MAC results**
   Hardware breakpoints on MAC instructions may corrupt the destination register of the MAC instruction.
   Fix/Workaround
   Place breakpoints on earlier or later instructions.

2. **Privilege violation when using interrupts in application mode with protected system stack**
   If the system stack is protected by the MPU and an interrupt occurs in application mode, an MPU DTLB exception will occur.
   Fix/Workaround
   Make a DTLB Protection (Write) exception handler which permits the interrupt request to be handled in privileged mode.

35.1.2 FLASHCDW

1. **Flash self programming may fail in one wait state mode**
   Writes in flash and user pages may fail if executing code is located in address space mapped to flash, and the flash controller is configured in one wait state mode (the Flash Wait State bit in the Flash Control Register (FCR.FWS) is one).
   Fix/Workaround
   Solution 1: Configure the flash controller in zero wait state mode (FCR.FWS=0).
   Solution 2: Configure the HMATRIX master 1 (CPU Instruction) to use the unlimited burst length transfer mode (MCFG1.ULBT=0), and the HMATRIX slave 0 (FLASHCDW) to use the maximum slot cycle limit (SCFG0.SLOT_CYCLE=255).

35.1.3 Power Manager

1. **Clock sources will not be stopped in Static mode if the difference between CPU and PBx division factor is larger than 4**
   If the division factor between the CPU/HSB and PBx frequencies is more than 4 when entering a sleep mode where the system RC oscillator (RCSYS) is turned off, the high speed clock sources will not be turned off. This will result in a significantly higher power consumption during the sleep mode.
   Fix/Workaround
   Before going to sleep modes where RCSYS is stopped, make sure the division factor between CPU/HSB and PBx frequencies is less than or equal to 4.

2. **Clock Failure Detector (CFD) can be issued while turning off the CFD**
   While turning off the CFD, the CFD bit in the Status Register (SR) can be set. This will change the main clock source to RCSYS.
   Fix/Workaround
   Solution 1: Enable CFD interrupt. If CFD interrupt is issues after turning off the CFD, switch back to original main clock source.
   Solution 2: Only turn off the CFD while running the main clock on RCSYS.

3. **Sleepwalking in idle and frozen sleep mode will mask all other PB clocks**