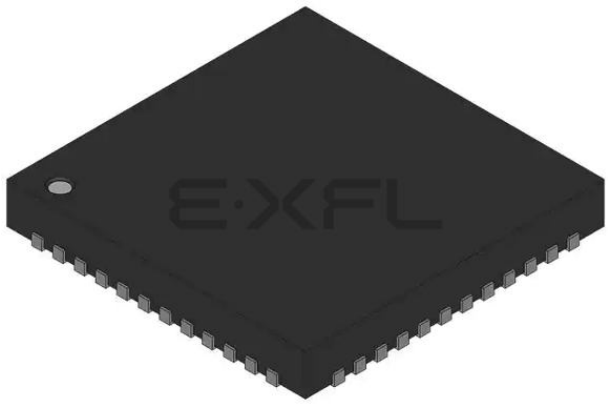


**Welcome to E-XFL.COM**

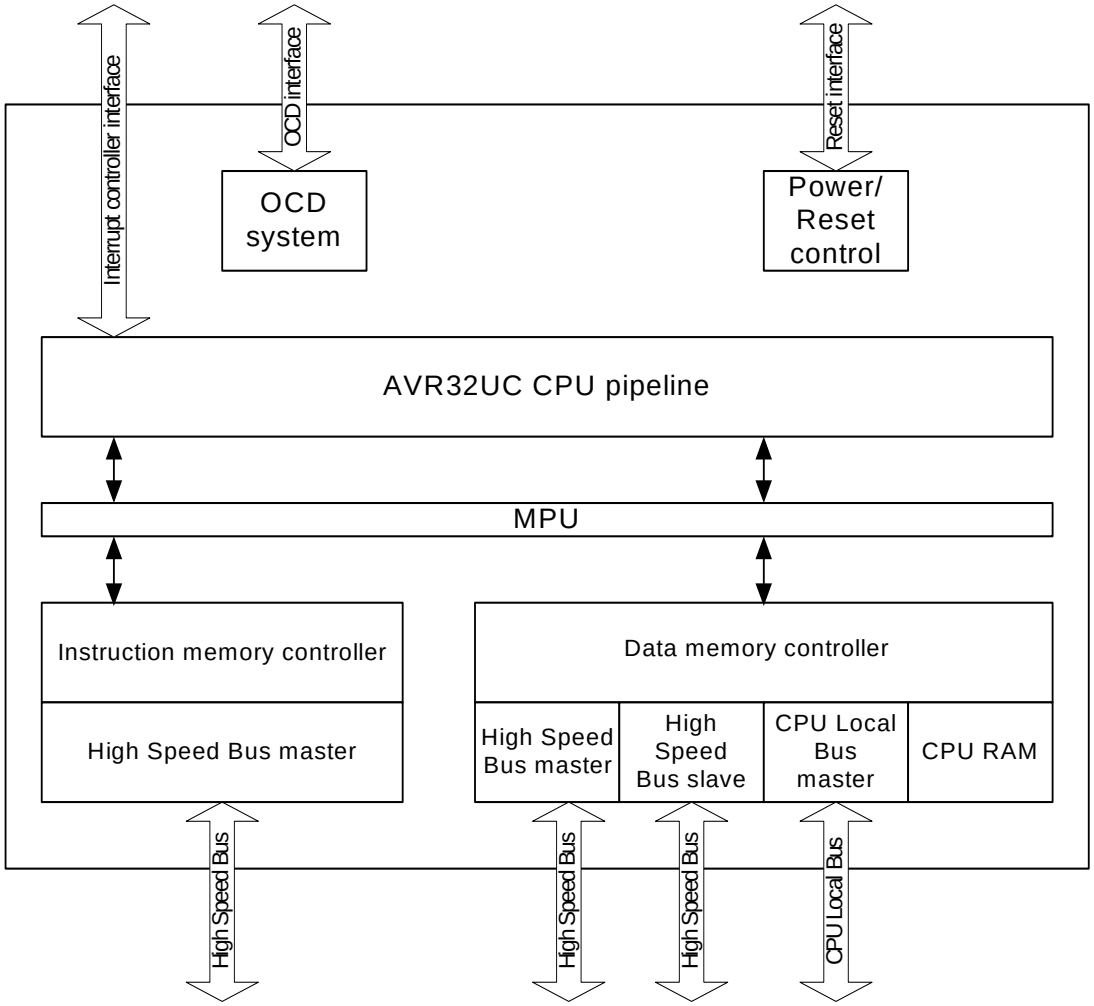
## What is "Embedded - Microcontrollers"?

"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 | 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 | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 32K 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/atmel/at32uc3l0128-d3ht |

**Figure 4-1.** Overview of the AVR32UC CPU




### 4.3.1 Pipeline Overview

AVR32UC has three pipeline stages, Instruction Fetch (IF), Instruction Decode (ID), and Instruction Execute (EX). The EX stage is split into three parallel subsections, one arithmetic/logic (ALU) section, one multiply (MUL) section, and one load/store (LS) section.

Instructions are issued and complete in order. Certain operations require several clock cycles to complete, and in this case, the instruction resides in the ID and EX stages for the required number of clock cycles. Since there is only three pipeline stages, no internal data forwarding is required, and no data dependencies can arise in the pipeline.

Figure 4-2 on page 21 shows an overview of the AVR32UC pipeline stages.

# 9. Secure Access Unit (SAU)

Rev: 1.1.1.3

## 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.

### 9.6.1 Control Register

**Name:** CR

**Access Type:** Write-only

**Offset:** 0x00

**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 |
|---|---|---|---|---|---|---|---|
| - | - | BERRDIS | BERREN | SDIS | SEN | DIS | EN |

- **BERRDIS: Bus Error Response Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables Bus Error Response from the SAU.
- **BERREN: Bus Error Response Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables Bus Error Response from the SAU.
- **SDIS: Setup Mode Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit exits setup mode.
- **SEN: Setup Mode Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enters setup mode.
- **DIS: SAU Disable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit disables the SAU.
- **EN: SAU Enable**
  Writing a zero to this bit has no effect.
  Writing a one to this bit enables the SAU.

### 9.6.11 Version Register

**Name:** VERSION

**Access Type:** Write-only

**Offset:** 0x28

**Reset Value:** -

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

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | VERSION[11:8] | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VERSION[7:0] | | | | | | | |

- **VARIANT: Variant Number**
  Reserved. No functionality associated.
- **VERSION: Version Number**
  Version number of the module. No functionality associated.

### 10.5.4 Priority Registers B For Slaves

**Name:** PRBS0...PRBS15

**Access Type:** Read/Write

**Offset:** -

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | M15PR | | - | - | M14PR | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| - | - | M13PR | | - | - | M12PR | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| - | - | M11PR | | - | - | M10PR | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | M9PR | | - | - | M8PR | |

- **MxPR: Master x Priority**

  Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

#### 13.5.8.3 Factory calibration

After a Power-on Reset (POR) the VREGCR.CALIB field is loaded with a factory defined calibration value. This value is chosen so that the normal output voltage of the regulator after a power-up is 1.8V.

Although it is not recommended to override default factory settings, it is still possible to override these default values by writing to VREGCR.CALIB.

If the Flash Calibration Done bit in VREGCR (VREGCR.FCD) is zero, the flash calibration will be redone after any reset, and the VREGCR.FCD bit will be set before program execution starts in the CPU. If VREGCR.FCD is one, the flash calibration will only be redone after a POR.

#### 13.5.8.4 POR33 control

VREGCR includes control bits for the Power-on Reset 3.3V (POR33) detector that monitors the internal regulator supply voltage. The POR33 detector is enabled by default but can be disabled by software to reduce power consumption. The 3.3V Supply Monitor (SM33) can then be used to monitor the regulator power supply.

The POR33 detector is disabled by writing a zero to the POR33 Enable bit (VREGCR.POR33EN). Because of internal synchronisation, the POR33 detector is not immediately enabled or disabled. The actual state of the POR33 detector can be read from the POR33 Status bit (VREGCR.POR33STATUS).

The 32kHz RC oscillator (RC32K) must be enabled before disabling the POR33 detector. Once the POR33 detector has been disabled, the RC32K oscillator can be disabled again.

To avoid spurious resets, it is mandatory to mask the Power-on Reset when enabling or disabling the POR33 detector. The Power-on Reset generated by the POR33 detector can be ignored by writing a one to the POR33 Mask bit (VREGCR.POR33MASK). Because of internal synchronization, the masking is not immediately effective, so software should wait for the VREGCR.POR33MASK to read as a one before assuming the masking is effective.

The output of the POR33 detector is zero if the internal regulator supply voltage is below the POR33 power-on threshold level, and one if the internal regulator supply voltage is above the POR33 power-on threshold level. This output (before masking) can be read from the POR33 Value bit (VREGCR.POR33VALUE).

#### 13.5.8.5 POR18 control

VREGCR includes control bits for the Power-on Reset 1.8V (POR18) detector that monitors the VDDCORE voltage. The POR18 detector is enabled by default but can be disabled by software to reduce power consumption.

The POR18 detector is disabled by writing a zero to the POR18 Enable bit (VREGCR.POR18EN). Because of internal synchronization, the POR18 detector is not immediately enabled or disabled. The actual state of the POR18 detector can be read from the POR18 Status bit (VREGCR.POR18STATUS).

Please note that the POR18 detector cannot be disabled while the JTAG or aWire debug interface is used. Writing a zero to VREGCR.POR18EN bit will have no effect.

To avoid spurious resets, it is mandatory to mask the Power-on Reset when enabling or disabling the POR18 detector. The Power-on Reset generated by the POR18 detector can be ignored by writing a one to the POR18 Mask bit (VREGCR.POR18MASK). Because of internal

### 18.7.11 Pull-up Enable Register

**Name:** PUER

**Access:** Read/Write, Set, Clear, Toggle

**Offset**: 0x070, 0x074, 0x078, 0x07C

**Reset Value:** -

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

- **P0-31: Pull-up Enable**
  Writing a zero to a bit in this register will disable pull-up on the corresponding pin.
  Writing a one to a bit in this register will enable pull-up on the corresponding pin.

### 21.7.3 Clocks

The clock for the TWIM bus interface (CLK_TWIM) 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 TWIM before disabling the clock, to avoid freezing the TWIM in an undefined state.

### 21.7.4 DMA

The TWIM DMA handshake interface is connected to the Peripheral DMA Controller. Using the TWIM DMA functionality requires the Peripheral DMA Controller to be programmed after setting up the TWIM.

### 21.7.5 Interrupts

The TWIM interrupt request lines are connected to the interrupt controller. Using the TWIM interrupts requires the interrupt controller to be programmed first.

### 21.7.6 Debug Operation

When an external debugger forces the CPU into debug mode, the TWIM continues normal operation. If the TWIM is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

Write this bit to one if the command in CMDR performs a repeated start to the same slave address as addressed in the previous transfer in order to enter master receiver mode.

Write this bit to zero otherwise.

- **TENBIT: Ten Bit Addressing Mode**

  0: Use 7-bit addressing mode.

  1: Use 10-bit addressing mode. Must not be used when the TWIM is in SMBus mode.

- **SADR: Slave Address**

  Address of the slave involved in the transfer. Bits 9-7 are don't care if 7-bit addressing is used.

- **READ: Transfer Direction**

  0: Allow the master to transmit data.

  1: Allow the master to receive data.

### 22.9.1 Control Register

**Name:** CR

**Access Type:** Read/Write

**Offset**: 0x00

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | TENBIT | ADR[9:8] | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| ADR[7:0] | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SODR | SOAM | CUP | ACK | PECEN | SMHH | SMDA | SMBALERT |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRST | - | - | STREN | GCMATCH | SMATCH | SMEN | SEN |

- **TENBIT: Ten Bit Address Match**
  0: Disables Ten Bit Address Match.
  1: Enables Ten Bit Address Match.
- **ADR: Slave Address**
  Slave address used in slave address match. Bits 9:0 are used if in 10-bit mode, bits 6:0 otherwise.
- **SODR: Stretch Clock on Data Byte Reception**
  0: Does not stretch bus clock immediately before ACKing a received data byte.
  1: Stretches bus clock immediately before ACKing a received data byte.
- **SOAM: Stretch Clock on Address Match**
  0: Does not stretch bus clock after address match.
  1: Stretches bus clock after address match.
- **CUP: NBYTES Count Up**
  0: Causes NBYTES to count down (decrement) per byte transferred.
  1: Causes NBYTES to count up (increment) per byte transferred.
- **ACK: Slave Receiver Data Phase ACK Value**
  0: Causes a low value to be returned in the ACK cycle of the data phase in slave receiver mode.
  1: Causes a high value to be returned in the ACK cycle of the data phase in slave receiver mode.
- **PECEN: Packet Error Checking Enable**
  0: Disables SMBus PEC (CRC) generation and check.
  1: Enables SMBus PEC (CRC) generation and check.
- **SMHH: SMBus Host Header**
  0: Causes the TWIS not to acknowledge the SMBus Host Header.
  1: Causes the TWIS to acknowledge the SMBus Host Header.
- **SMDA: SMBus Default Address**
  0: Causes the TWIS not to acknowledge the SMBus Default Address.
  1: Causes the TWIS to acknowledge the SMBus Default Address.
- **SMBALERT: SMBus Alert**
  0: Causes the TWIS to release the SMBALERT line and not to acknowledge the SMBus Alert Response Address (ARA).
  1: Causes the TWIS to pull down the SMBALERT line and to acknowledge the SMBus Alert Response Address (ARA).

The current value of the counter is accessible in real time by reading the Channel n Counter Value Register (CVn). The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

### 24.6.1.3 *Clock selection*

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the configurable I/O signals A0, A1 or A2 for chaining by writing to the BMR register. See Figure 24-2 on page 562.

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5. See the Module Configuration Chapter for details about the connection of these clock sources.
- External clock signals: XC0, XC1 or XC2. See the Module Configuration Chapter for details about the connection of these clock sources.

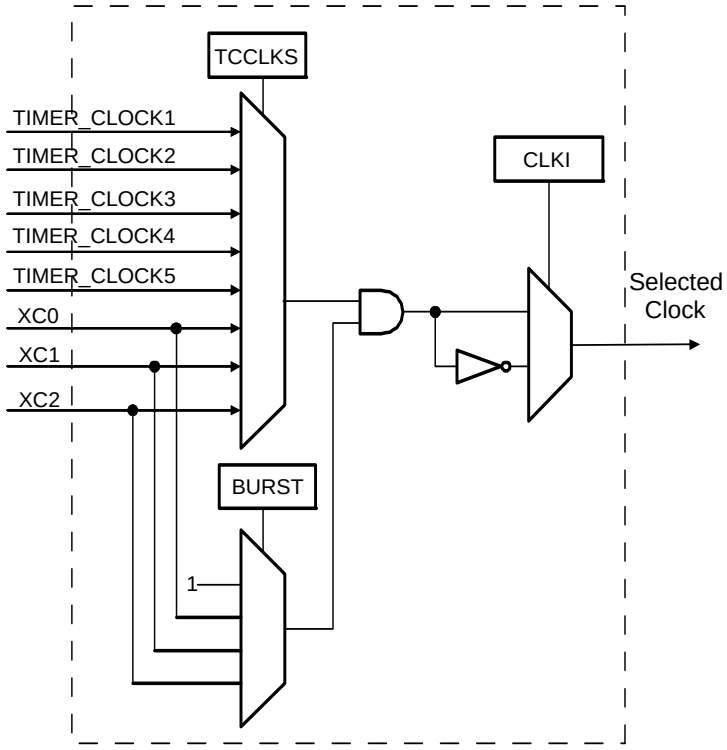
This selection is made by the Clock Selection field in the Channel n Mode Register (CMRn.TCCLKS).

The selected clock can be inverted with the Clock Invert bit in CMRn (CMRn.CLKI). This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The Burst Signal Selection field in the CMRn register (CMRn.BURST) defines this signal.

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the CLK_TC period. The external clock frequency must be at least 2.5 times lower than the CLK_TC.

**Figure 24-2.** Clock Selection

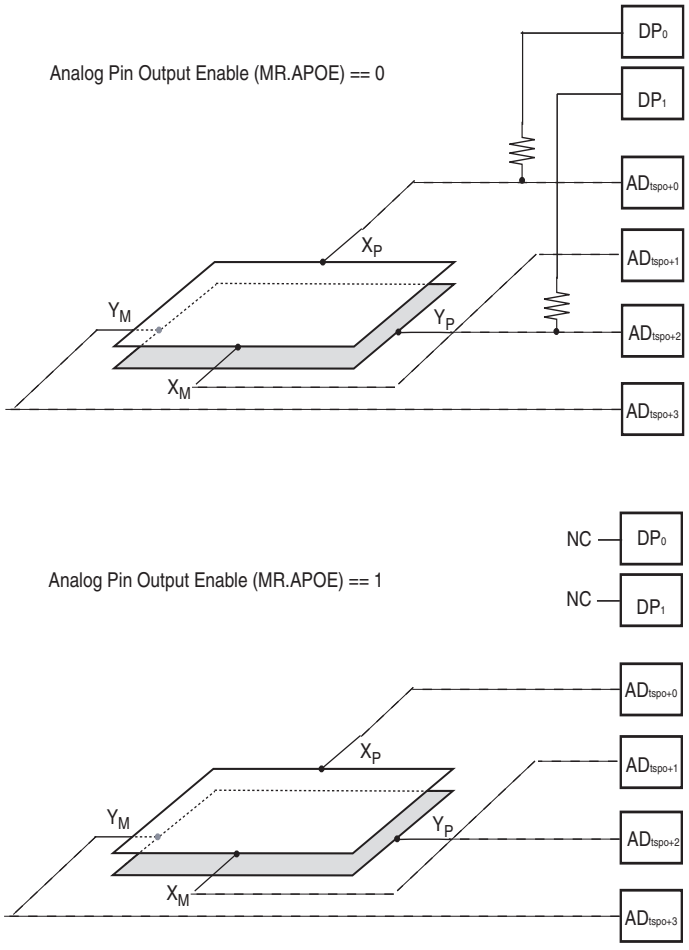


$$V(AD_{tspo+2}) = \frac{R_{filmy}}{R_{filmy} + R_{resistory}} \cdot V(DP_1)$$

The ADP pins are used by default, as the APOE bit is zero after reset. Writing a one to the APOE bit instructs the ADCIFB Resistive Touch Screen Sequencer to use the already connected ADtspo+0 and ADtspo+2 pins to drive VDD to $X_P$ and $Y_P$ signals directly. In this mode the ADP pins can be used as general purpose I/O pins.

Before writing a one to the APOE bit the user must make sure that the I/O voltage is compatible with the ADC input voltage. If the I/O voltage is higher than the maximum input voltage of the ADC, permanent damage may occur. Refer to the Electrical Characteristics chapter for details.

**Figure 26-3.** Resistive Touch Screen Pin Connections



### 26.9.9 Interrupt Enable Register

**Name:** IER

**Access Type:** Write-only

**Offset:** 0x20

**Reset Value:** 0x00000000

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

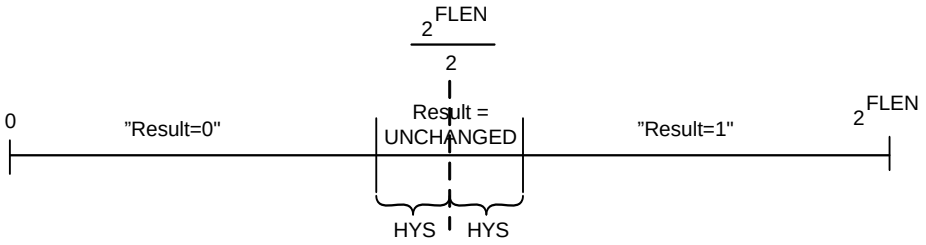
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| - | CELSE | CGT | CLT | - | - | BUSY | READY |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | NOCNT | PENCNT | - | - | OVRE | DRDY |

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.

**Figure 27-3.** The Filtering Algorithm




## 27.7 Peripheral Event Triggers

Peripheral events from other modules can trigger comparisons in the ACIFB. All channels that are set up in Event Triggered Single Measurement Mode will be started simultaneously when a peripheral event is received. Channels that are operating in Continuous Measurement Mode or User Triggered Single Measurement Mode will be unaffected by the received event. The software can still operate these channels independently of channels in Event Triggered Single Measurement Mode.

A peripheral event will trigger one or more comparisons, in normal or window mode.

## 27.8 AC Test mode

By writing the Analog Comparator Test Mode (CR.ACTEST) bit to one, the outputs from the ACs are overridden by the value in the Test Register (TR), see Figure 27-1. This is useful for software development.

### 27.9.9 Parameter Register

**Name:** PARAMETER

**Access Type:** Read-only

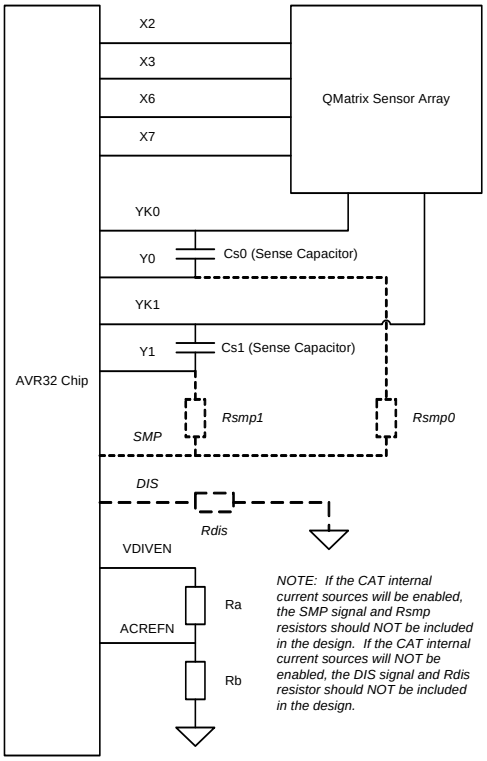
**Offset:** 0x30

**Reset Value:** -

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | - | - | WIMPL3 | WIMPL2 | WIMPL1 | WIMPL0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | - | - | - | - | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACIMPL7 | ACIMPL6 | ACIMPL5 | ACIMPL4 | ACIMPL3 | ACIMPL2 | ACIMPL1 | ACIMPL0 |

- **WIMPLn: Window Pair n Implemented**
  0: Window Pair not implemented.
  1: Window Pair implemented.
- **ACIMPLn: Analog Comparator n Implemented**
  0: Analog Comparator not implemented.
  1: Analog Comparator implemented.

**Figure 28-3.** CAT Matrix Connections




In order to use the autonomous QTouch detection capability, the user must first set up the Autonomous Touch Pin Select Register (ATPINS) and Autonomous/DMA Touch Configuration Registers (ATCFG0 through 3) with appropriate values. The module can then be enabled using the Control Register (CTRL). After the module is enabled, the module will acquire data from the autonomous QTouch sensor and use it to determine whether the sensor is activated. The active/inactive status of the autonomous QTouch sensor is reported in the Status Register (SR), and it is also possible to configure the CAT to generate an interrupt whenever the status changes. The module will continue acquiring autonomous QTouch sensor data and updating autonomous QTouch status until the module is disabled or reset.

In order to use the DMATouch capability, it is first necessary to set up the pin mode registers (PINMODE0, PINMODE1, and PINMODE2) so that the desired pins are specified as DMA-Touch. The Autonomous/DMA Touch Configuration Registers (ATCFG0 through 3) must also be configured with appropriate values. One channel of the Peripheral DMA Controller must be set up to transfer state words from a block of memory to the DMATSW register, and another channel must be set up to transfer state words from the DMATSR register back to the same block of memory. The module can then be enabled using the CTRL register. After the module is enabled, the module will acquire count values from each DMATouch sensor. Once the module has acquired a count value for a sensor, it will use a handshake interface to signal the Peripheral DMA controller to transfer a state word to the DMATSW register. The module will use the count value to update the state word, and then the updated state word will be transferred to the DMATSR register. Another handshake interface will signal the Peripheral DMA controller to transfer the contents of the DMATSR register back to memory. The status of the DMATouch sensors can be determined at any time by reading the DMATouch Sensor Status Register (DMATSS).

### 28.7.21 Discharge Current Source Register

**Name:** DICS

**Access Type:** Read/Write

**Offset:** 0x5C

**Reset Value:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| FSOURCES[7:0] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GLEN | - | - | - | - | - | INTVREFSEL | INTREFSEL |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| - | - | - | TRIM | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOURCES[7:0] | | | | | | | |

- **FSOURCES: Force Discharge Current Sources**
  When FSOURCES[n] is 0, the corresponding discharge current source behavior depends on SOURCES[n].
  When FSOURCES[n] is 1, the corresponding discharge current source is forced to be enabled continuously. This is useful for testing or debugging but should not be done during normal acquisition.
- **GLEN: Global Enable**
  0: The current source module is globally disabled.
  1: The current source module is globally enabled.
- **INTVREFSEL: Internal Voltage Reference Select**
  0: The voltage for the reference resistor is generated from the internal band gap circuit.
  1: The voltage for the reference resistor is VDDIO/3.
- **INTREFSEL: Internal Reference Select**
  0: The reference current flows through an external resistor on the DIS pin.
  1: The reference current flows through the internal reference resistor.
- **TRIM: Reference Current Trimming**
  This field is used to trim the discharge current. 0x00 corresponds to the minimum current value, and 0x1F corresponds to the maximum current value.
- **SOURCES: Enable Discharge Current Sources**
  When SOURCES[n] is 0, the corresponding discharge current source is disabled.
  When SOURCES[n] is 1, the corresponding discharge current source is enabled at appropriate times during acquisition.

### 30.7.3 Status Clear Register

**Name:** SCR

**Access Type:** Write-only

**Offset:** 0x08

**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 |
| - | - | - | - | - | - | - | - |

Writing a zero to a bit in this register has no effect.
Writing a one to a bit in this register will clear the corresponding bit in SR and the corresponding interrupt request.

### 30.7.9 Baud Rate Register

**Name:** BRR

**Access Type:** Read/Write

**Offset:** 0x20

**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 |
|---|---|---|---|---|---|---|---|
| BR[15:8] | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BR[7:0] | | | | | | | |

- **BR: Baud Rate**

  The baud rate ($f_{br}$) of the transmission, calculated using the following formula ($f_{aw}$ is the RC120M frequency):

$$f_{br} = \frac{8 f_{aw}}{BR}$$

  BR should not be set to a value smaller than 32.
  Writing a value to this field will update the baud rate of the transmission.
  Reading this field will give the current baud rate of the transmission.

## 31.3 On-Chip Debug

Rev: 2.1.2.0

### 31.3.1 Features

- **Debug interface in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 2+**
- **JTAG or aWire access to all on-chip debug functions**
- **Advanced Program, Data, Ownership, and Watchpoint trace supported**
- **NanoTrace aWire- or JTAG-based trace access**
- **Auxiliary port for high-speed trace information**
- **Hardware support for 6 Program and 2 Data breakpoints**
- **Unlimited number of software breakpoints supported**
- **Automatic CRC check of memory regions**

### 31.3.2 Overview

Debugging on the AT32UC3L0128/256 is facilitated by a powerful On-Chip Debug (OCD) system. The user accesses this through an external debug tool which connects to the JTAG or aWire port and the Auxiliary (AUX) port if implemented. The AUX port is primarily used for trace functions, and an aWire- or JTAG-based debugger is sufficient for basic debugging.

The debug system is based on the Nexus 2.0 standard, class 2+, which includes:

- Basic run-time control
- Program breakpoints
- Data breakpoints
- Program trace
- Ownership trace
- Data trace

In addition to the mandatory Nexus debug features, the AT32UC3L0128/256 implements several useful OCD features, such as:

- Debug Communication Channel between CPU and debugger
- Run-time PC monitoring
- CRC checking
- NanoTrace
- Software Quality Assurance (SQA) support

The OCD features are controlled by OCD registers, which can be accessed by the debugger, for instance when the NEXUS_ACCESS JTAG instruction is loaded. The CPU can also access OCD registers directly using mtdr/mfdr instructions in any privileged mode. The OCD registers are implemented based on the recommendations in the Nexus 2.0 standard, and are detailed in the AVR32UC Technical Reference Manual.

### 31.3.3 I/O Lines Description

The OCD AUX trace port contains a number of pins, as shown in Table 31-6 on page 741. These are multiplexed with I/O Controller lines, and must explicitly be enabled by writing OCD registers before the debug session starts. The AUX port is mapped to two different locations,