## Features

- High-performance, Low-power 32-bit Atmel® AVR® Microcontroller
  - Compact Single-cycle RISC Instruction Set Including DSP Instructions
  - Read-modify-write Instructions and Atomic Bit Manipulation
  - Performance
    - Up to 64 DMIPS Running at 50 MHz from Flash (1 Flash Wait State)
    - Up to 36 DMIPS Running at 25 MHz from Flash (0 Flash Wait State)
  - Memory Protection Unit (MPU)
    - Secure Access Unit (SAU) providing User-defined Peripheral Protection
- picoPower® Technology for Ultra-low Power Consumption
- Multi-hierarchy Bus System
  - High-performance Data Transfers on Separate Buses for Increased Performance
  - 12 Peripheral DMA Channels Improve Speed for Peripheral Communication
- Internal High-speed Flash
  - 64 Kbytes, 32 Kbytes, and 16 Kbytes Versions
  - Single-cycle Access up to 25 MHz
  - FlashVault Technology Allows Pre-programmed Secure Library Support for End User Applications
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 100,000 Write Cycles, 15-year Data Retention Capability
  - Flash Security Locks and User-defined Configuration Area
- Internal High-speed SRAM, Single-cycle Access at Full Speed
  - 16 Kbytes (64 Kbytes and 32 Kbytes Flash), or 8 Kbytes (16 Kbytes Flash)
- Interrupt Controller (INTC)
  - Autovectored Low-latency Interrupt Service with Programmable Priority
- External Interrupt Controller (EIC)
- Peripheral Event System for Direct Peripheral to Peripheral Communication
- System Functions
  - Power and Clock Manager
  - SleepWalking Power Saving Control
  - Internal System RC Oscillator (RCSYS)
  - 32 KHz Oscillator
  - Multipurpose Oscillator and Digital Frequency Locked Loop (DFLL)
- Windowed Watchdog Timer (WDT)
- Asynchronous Timer (AST) with Real-time Clock Capability
  - Counter or Calendar Mode Supported
- Frequency Meter (FREQM) for Accurate Measuring of Clock Frequency
- Six 16-bit Timer/Counter (TC) Channels
  - External Clock Inputs, PWM, Capture and Various Counting Capabilities
- 36 PWM Channels (PWMA)
  - 8-bit PWM with a Source Clock up to 150 MHz
- Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Independent Baudrate Generator, Support for SPI Interfaces
  - Support for Hardware Handshaking
- One Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
  - Up to 15 SPI Slaves can be Addressed
- Two Master and Two Slave Two-wire Interface (TWI), 400 kbit/s $I^2C$-compatible
- One 8-channel Analog-to-digital Converter (ADC) with up to 12 Bits Resolution
  - Internal Temperature Sensor

# 32-bit Atmel AVR Microcontroller

# AT32UC3L064
# AT32UC3L032
# AT32UC3L016

# Summary

32099HS–12/2011

- Eight Analog Comparators (AC) with Optional Window Detection
- Capacitive Touch (CAT) Module
  - Hardware-assisted Atmel® AVR® QTouch® and Atmel® AVR® QMatrix Touch Acquisition
  - Supports QTouch and QMatrix Capture from Capacitive Touch Sensors
- QTouch Library Support
  - Capacitive Touch Buttons, Sliders, and Wheels
  - QTouch and QMatrix Acquisition
- On-chip Non-intrusive Debug System
  - Nexus Class 2+, Runtime Control, Non-intrusive Data and Program Trace
  - aWire Single-pin Programming Trace and Debug Interface Muxed with Reset Pin
  - NanoTrace Provides Trace Capabilities through JTAG or aWire Interface
- 48-pin TQFP/QFN/TLLGA (36 GPIO Pins)
- Five High-drive I/O Pins
- Single 1.62-3.6 V Power Supply

# 1. Description

The Atmel® AVR® AT32UC3L016/32/64 is a complete system-on-chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 50MHz. AVR32 UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density, and high performance.

The processor implements a Memory Protection Unit (MPU) and a fast and flexible interrupt controller for supporting modern operating systems and real-time operating systems. The Secure Access Unit (SAU) is used together with the MPU to provide the required security and integrity.

Higher computation capability is achieved using a rich set of DSP instructions.

The AT32UC3L016/32/64 embeds state-of-the-art picoPower technology for ultra-low power consumption. Combined power control techniques are used to bring active current consumption down to 165µA/MHz, and leakage down to 9nA while still retaining a bank of backup registers. The device allows a wide range of trade-offs between functionality and power consumption, giving the user the ability to reach the lowest possible power consumption with the feature set required for the application.

The Peripheral Direct Memory Access (DMA) controller enables data transfers between peripherals and memories without processor involvement. The Peripheral DMA controller drastically reduces processing overhead when transferring continuous and large data streams.

The AT32UC3L016/32/64 incorporates on-chip Flash and SRAM memories for secure and fast access. The FlashVault technology allows secure libraries to be programmed into the device. The secure libraries can be executed while the CPU is in Secure State, but not read by non-secure software in the device. The device can thus be shipped to end customers, who will be able to program their own code into the device to access the secure libraries, but without risk of compromising the proprietary secure code.

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.

The Peripheral Event System allows peripherals to receive, react to, and send peripheral events without CPU intervention. Asynchronous interrupts allow advanced peripheral operation in low power sleep modes.

The Power Manager (PM) improves design flexibility and security. The Power Manager supports SleepWalking functionality, by which a module can be selectively activated based on peripheral events, even in sleep modes where the module clock is stopped. Power monitoring is supported by on-chip Power-on Reset (POR), Brown-out Detector (BOD), and Supply Monitor (SM). The device features several oscillators, such as Digital Frequency Locked Loop (DFLL), Oscillator 0 (OSC0), and system RC oscillator (RCSYS). Either of these oscillators can be used as source for the system clock. The DFLL is a programmable internal oscillator from 40 to 150MHz. It can be tuned to a high accuracy if an accurate oscillator is running, e.g. the 32KHz crystal oscillator.

The Watchdog Timer (WDT) will reset the device unless it is periodically serviced by the software. This allows the device to recover from a condition that has caused the system to be unstable.

The Asynchronous Timer (AST) combined with the 32KHz crystal oscillator supports powerful real-time clock capabilities, with a maximum timeout of up to 136 years. The AST can operate in counter mode or calendar mode.

The Frequency Meter (FREQM) allows accurate measuring of a clock frequency by comparing it to a known reference clock.

The device includes six identical 16-bit Timer/Counter (TC) channels. Each channel can be independently programmed to perform frequency measurement, event counting, interval measurement, pulse generation, delay timing, and pulse width modulation.

The Pulse Width Modulation controller (PWMA) provides 8-bit PWM channels which can be synchronized and controlled from a common timer. One PWM channel is available for each I/O pin on the device, enabling applications that require multiple PWM outputs, such as LCD backlight control. The PWM channels can operate independently, with duty cycles set independently from each other, or in interlinked mode, with multiple channels changed at the same time.

The AT32UC3L016/32/64 also features many communication interfaces for communication intensive applications like USART, SPI, and TWI. The USART supports different communication modes, like SPI Mode and LIN Mode.

A general purpose 8-channel ADC is provided, as well as eight analog comparators (AC). The ADC can operate in 10-bit mode at full speed or in enhanced mode at reduced speed, offering up to 12-bit resolution. The ADC also provides an internal temperature sensor input channel. The analog comparators can be paired to detect when the sensing voltage is within or outside the defined reference window.

The Capacitive Touch (CAT) module senses touch on external capacitive touch sensors, using the QTouch technology. Capacitive touch sensors use no external mechanical components, unlike normal push buttons, and therefore demand less maintenance in the user application. The CAT module allows up to 17 touch sensors, or up to 16 by 8 matrix sensors to be interfaced. One touch sensor can be configured to operate autonomously without software interaction, allowing wakeup from sleep modes when activated.

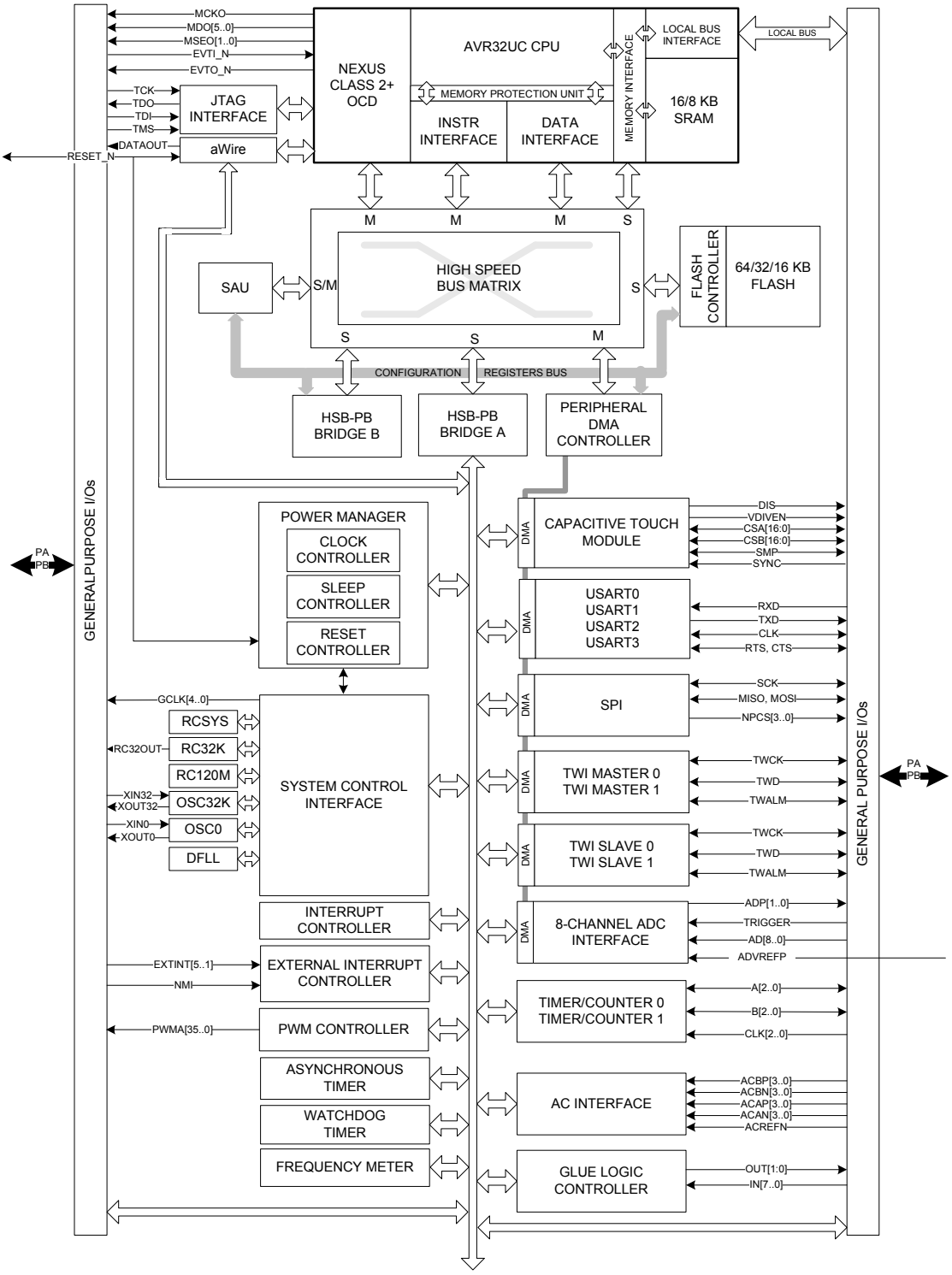
Atmel offers the QTouch library for embedding capacitive touch buttons, sliders, and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys as well as Adjacent Key Suppression® (AKS®) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop, and debug your own touch applications.

The AT32UC3L016/32/64 integrates a class 2+ Nexus 2.0 On-chip Debug (OCD) System, with non-intrusive real-time trace and full-speed read/write memory access, in addition to basic run-time control. The NanoTrace interface enables trace feature for aWire- or JTAG-based debuggers. The single-pin aWire interface allows all features available through the JTAG interface to be accessed through the RESET pin, allowing the JTAG pins to be used for GPIO or peripherals.

# 2. Overview

## 2.1 Block Diagram

**Figure 2-1.** Block Diagram



## 2.2 Configuration Summary

**Table 2-1.** Configuration Summary

| Feature | AT32UC3L064 | AT32UC3L032 | AT32UC3L016 |
|---|---|---|---|
| Flash | 64KB | 32KB | 16KB |
| SRAM | 16KB | 16KB | 8KB |
| GPIO | 36 | | |
| High-drive pins | 5 | | |
| External Interrupts | 6 | | |
| TWI | 2 | | |
| USART | 4 | | |
| Peripheral DMA Channels | 12 | | |
| Peripheral Event System | 1 | | |
| SPI | 1 | | |
| Asynchronous Timers | 1 | | |
| Timer/Counter Channels | 6 | | |
| PWM channels | 36 | | |
| Frequency Meter | 1 | | |
| Watchdog Timer | 1 | | |
| Power Manager | 1 | | |
| Secure Access Unit | 1 | | |
| Glue Logic Controller | 1 | | |
| Oscillators | Digital Frequency Locked Loop 40-150MHz (DFLL)<br>Crystal Oscillator 3-16MHz (OSC0)<br>Crystal Oscillator 32KHz (OSC32K)<br>RC Oscillator 120MHz (RC120M)<br>RC Oscillator 115kHz (RCSYS)<br>RC Oscillator 32kHz (RC32K) | | |
| ADC | 8-channel 12-bit | | |
| Temperature Sensor | 1 | | |
| Analog Comparators | 8 | | |
| Capacitive Touch Module | 1 | | |
| JTAG | 1 | | |
| aWire | 1 | | |
| Max Frequency | 50MHz | | |
| Packages | TQFP48/QFN48/TLLGA48 | | |

# 3. Package and Pinout

## 3.1 Package

The device pins are multiplexed with peripheral functions as described in Section 3.2.

**Figure 3-1.** TQFP48/QFN48 Pinout

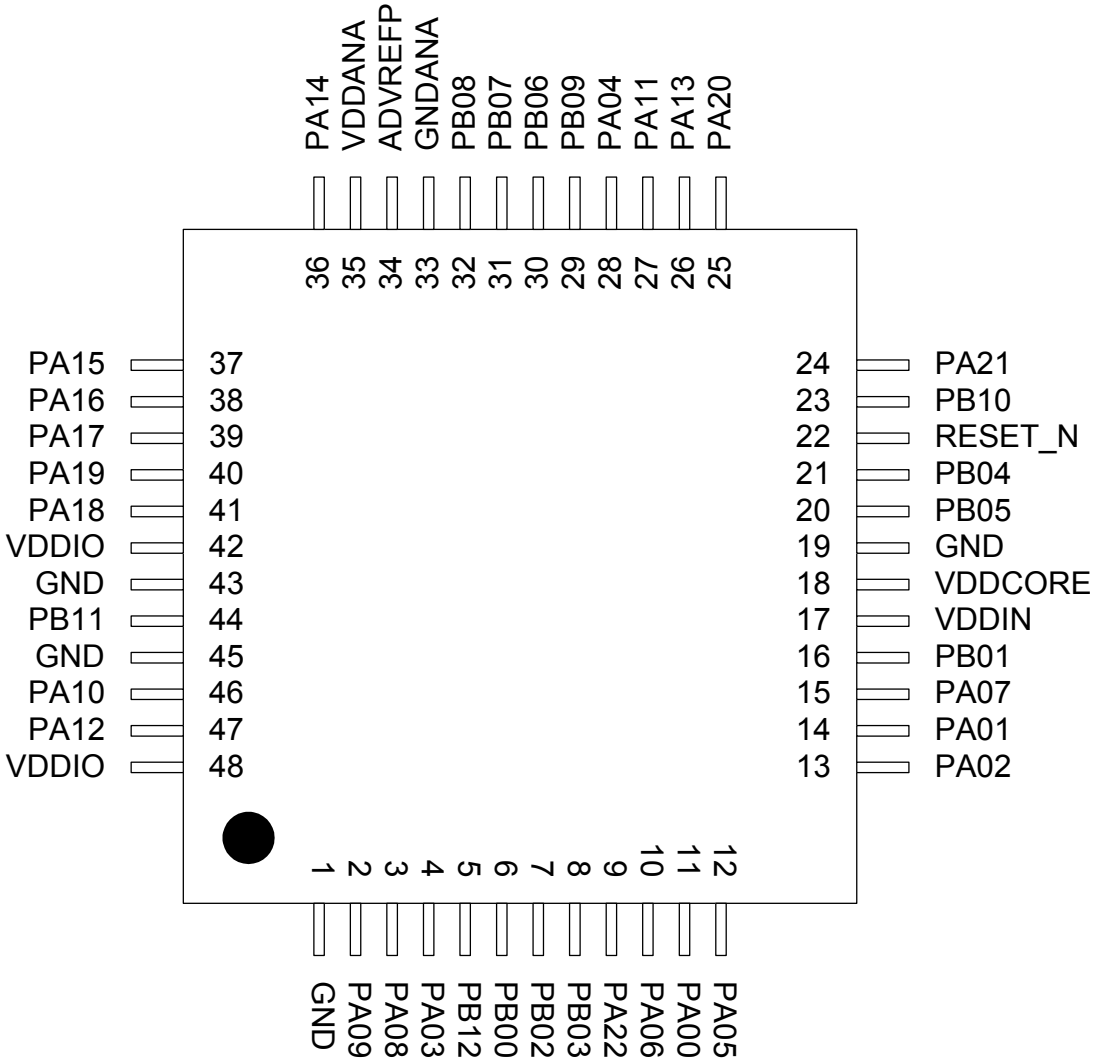
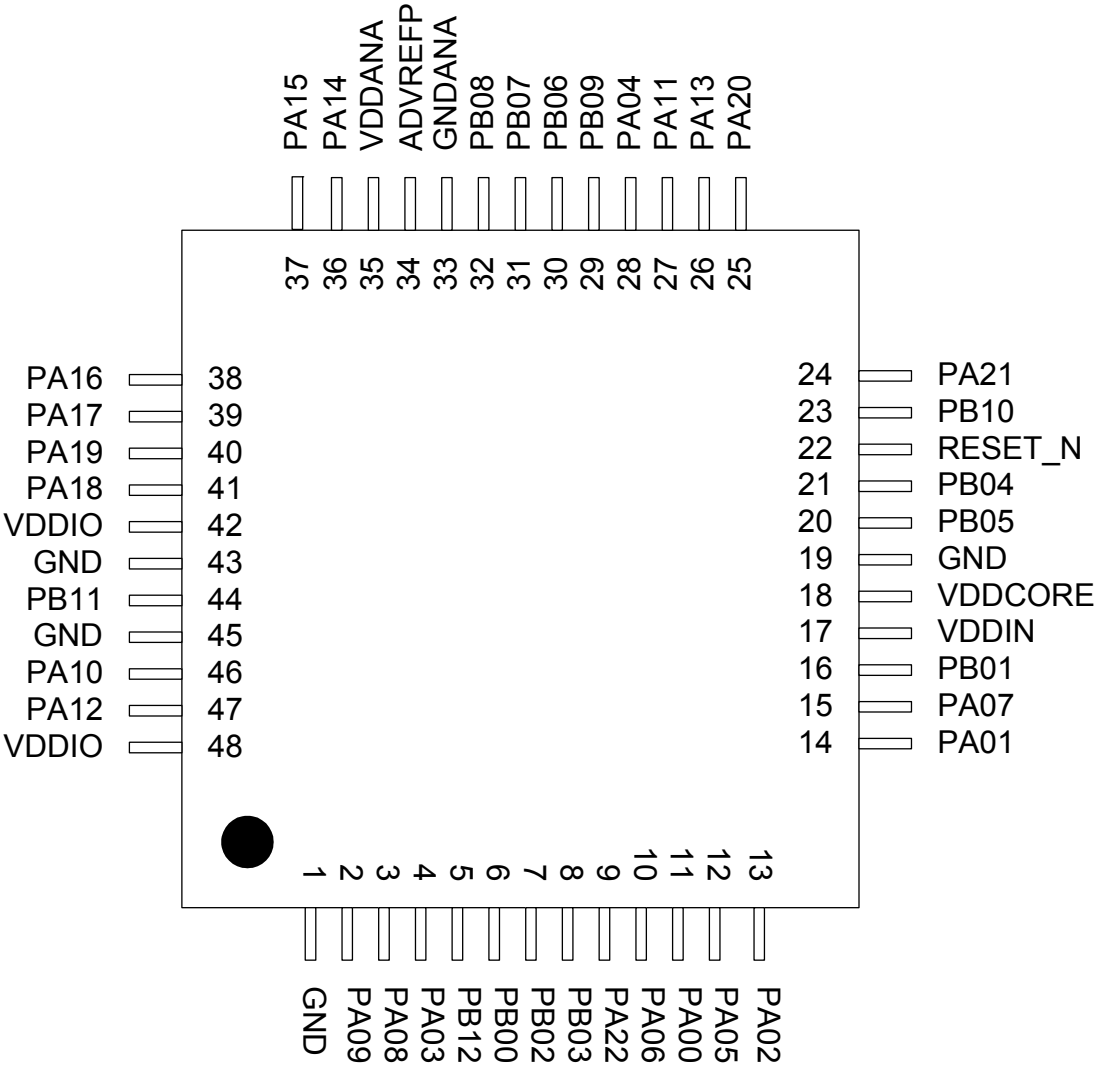


**Figure 3-2.** TLLGA48 Pinout



## 3.2 Peripheral Multiplexing on I/O lines

### 3.2.1 Multiplexed signals

Each GPIO line can be assigned to one of the peripheral functions.The following table describes the peripheral signals multiplexed to the GPIO lines.

**Table 3-1.** GPIO Controller Function Multiplexing

| 48-pin | PIN | GPIO | Supply | Pin Type | GPIO Function | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | A | B | C | D | E | F | G | H |
| 11 | PA00 | 0 | VDDIO | Normal I/O | USART0 TXD | USART1 RTS | SPI NPCS[2] | | PWMA PWMA[0] | | SCIF GCLK[0] | CAT CSA[2] |
| 14 | PA01 | 1 | VDDIO | Normal I/O | USART0 RXD | USART1 CTS | SPI NPCS[3] | USART1 CLK | PWMA PWMA[1] | ACIFB ACAP[0] | TWIMS0 TWALM | CAT CSA[1] |
| 13 | PA02 | 2 | VDDIO | High-drive I/O | USART0 RTS | ADCIFB TRIGGER | USART2 TXD | TC0 A0 | PWMA PWMA[2] | ACIFB ACBP[0] | USART0 CLK | CAT CSA[3] |
| 4 | PA03 | 3 | VDDIO | Normal I/O | USART0 CTS | SPI NPCS[1] | USART2 TXD | TC0 B0 | PWMA PWMA[3] | ACIFB ACBN[3] | USART0 CLK | CAT CSB[3] |
| 28 | PA04 | 4 | VDDIO | Normal I/O | SPI MISO | TWIMS0 TWCK | USART1 RXD | TC0 B1 | PWMA PWMA[4] | ACIFB ACBP[1] | | CAT CSA[7] |
| 12 | PA05 | 5 | VDDIO | Normal I/O (TWI) | SPI MOSI | TWIMS1 TWCK | USART1 TXD | TC0 A1 | PWMA PWMA[5] | ACIFB ACBN[0] | TWIMS0 TWD | CAT CSB[7] |
| 10 | PA06 | 6 | VDDIO | High-drive I/O, 5V tolerant | SPI SCK | USART2 TXD | USART1 CLK | TC0 B0 | PWMA PWMA[6] | | SCIF GCLK[1] | CAT CSB[1] |
| 15 | PA07 | 7 | VDDIO | Normal I/O (TWI) | SPI NPCS[0] | USART2 RXD | TWIMS1 TWALM | TWIMS0 TWCK | PWMA PWMA[7] | ACIFB ACAN[0] | EIC EXTINT[0] | CAT CSB[2] |
| 3 | PA08 | 8 | VDDIO | High-drive I/O | USART1 TXD | SPI NPCS[2] | TC0 A2 | ADCIFB ADP[0] | PWMA PWMA[8] | | | CAT CSA[4] |
| 2 | PA09 | 9 | VDDIO | High-drive I/O | USART1 RXD | SPI NPCS[3] | TC0 B2 | ADCIFB ADP[1] | PWMA PWMA[9] | SCIF GCLK[2] | EIC EXTINT[1] | CAT CSB[4] |
| 46 | PA10 | 10 | VDDIO | Normal I/O | TWIMS0 TWD | | TC0 A0 | | PWMA PWMA[10] | ACIFB ACAP[1] | SCIF GCLK[2] | CAT CSA[5] |
| 27 | PA11 | 11 | VDDIN | Normal I/O | | | | | PWMA PWMA[11] | | | |
| 47 | PA12 | 12 | VDDIO | Normal I/O | ADCIFB PRND | USART2 CLK | TC0 CLK1 | CAT SMP | PWMA PWMA[12] | ACIFB ACAN[1] | SCIF GCLK[3] | CAT CSB[5] |
| 26 | PA13 | 13 | VDDIN | Normal I/O | GLOC OUT[0] | GLOC IN[7] | TC0 A0 | SCIF GCLK[2] | PWMA PWMA[13] | CAT SMP | EIC EXTINT[2] | CAT CSA[0] |
| 36 | PA14 | 14 | VDDIO | Normal I/O | ADCIFB AD[0] | TC0 CLK2 | USART2 RTS | CAT SMP | PWMA PWMA[14] | | SCIF GCLK[4] | CAT CSA[6] |
| 37 | PA15 | 15 | VDDIO | Normal I/O | ADCIFB AD[1] | TC0 CLK1 | | GLOC IN[6] | PWMA PWMA[15] | CAT SYNC | EIC EXTINT[3] | CAT CSB[6] |
| 38 | PA16 | 16 | VDDIO | Normal I/O | ADCIFB AD[2] | TC0 CLK0 | | GLOC IN[5] | PWMA PWMA[16] | ACIFB ACREFN | EIC EXTINT[4] | CAT CSA[8] |
| 39 | PA17 | 17 | VDDIO | Normal I/O (TWI) | | TC0 A1 | USART2 CTS | TWIMS1 TWD | PWMA PWMA[17] | CAT SMP | CAT DIS | CAT CSB[8] |
| 41 | PA18 | 18 | VDDIO | Normal I/O | ADCIFB AD[4] | TC0 B1 | | GLOC IN[4] | PWMA PWMA[18] | CAT SYNC | EIC EXTINT[5] | CAT CSB[0] |

**Table 3-1.** GPIO Controller Function Multiplexing

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | PA19 | 19 | VDDIO | Normal I/O | ADCIFB AD[5] | | TC0 A2 | TWIMS1 TWALM | PWMA PWMA[19] | | CAT SYNC | CAT CSA[10] |
| 25 | PA20 | 20 | VDDIN | Normal I/O | USART2 TXD | | TC0 A1 | GLOC IN[3] | PWMA PWMA[20] | SCIF RC32OUT | | CAT CSA[12] |
| 24 | PA21 | 21 | VDDIN | Normal I/O (TWI, 5V tolerant SMBus) | USART2 RXD | TWIMS0 TWD | TC0 B1 | ADCIFB TRIGGER | PWMA PWMA[21] | PWMA PWMAOD[21] | SCIF GCLK[0] | CAT SMP |
| 9 | PA22 | 22 | VDDIO | Normal I/O | USART0 CTS | USART2 CLK | TC0 B2 | CAT SMP | PWMA PWMA[22] | ACIFB ACBN[2] | | CAT CSB[10] |
| 6 | PB00 | 32 | VDDIO | Normal I/O | USART3 TXD | ADCIFB ADP[0] | SPI NPCS[0] | TC0 A1 | PWMA PWMA[23] | ACIFB ACAP[2] | TC1 A0 | CAT CSA[9] |
| 16 | PB01 | 33 | VDDIO | High-drive I/O | USART3 RXD | ADCIFB ADP[1] | SPI SCK | TC0 B1 | PWMA PWMA[24] | | TC1 A1 | CAT CSB[9] |
| 7 | PB02 | 34 | VDDIO | Normal I/O | USART3 RTS | USART3 CLK | SPI MISO | TC0 A2 | PWMA PWMA[25] | ACIFB ACAN[2] | SCIF GCLK[1] | CAT CSB[11] |
| 8 | PB03 | 35 | VDDIO | Normal I/O | USART3 CTS | USART3 CLK | SPI MOSI | TC0 B2 | PWMA PWMA[26] | ACIFB ACBP[2] | TC1 A2 | CAT CSA[11] |
| 21 | PB04 | 36 | VDDIN | Normal I/O (TWI, 5V tolerant SMBus) | TC1 A0 | USART1 RTS | USART1 CLK | TWIMS0 TWALM | PWMA PWMA[27] | PWMA PWMAOD[27] | TWIMS1 TWCK | CAT CSA[14] |
| 20 | PB05 | 37 | VDDIN | Normal I/O (TWI, 5V tolerant SMBus) | TC1 B0 | USART1 CTS | USART1 CLK | TWIMS0 TWCK | PWMA PWMA[28] | PWMA PWMAOD[28] | SCIF GCLK[3] | CAT CSB[14] |
| 30 | PB06 | 38 | VDDIO | Normal I/O | TC1 A1 | USART3 TXD | ADCIFB AD[6] | GLOC IN[2] | PWMA PWMA[29] | ACIFB ACAN[3] | EIC EXTINT[0] | CAT CSB[13] |
| 31 | PB07 | 39 | VDDIO | Normal I/O | TC1 B1 | USART3 RXD | ADCIFB AD[7] | GLOC IN[1] | PWMA PWMA[30] | ACIFB ACAP[3] | EIC EXTINT[1] | CAT CSA[13] |
| 32 | PB08 | 40 | VDDIO | Normal I/O | TC1 A2 | USART3 RTS | ADCIFB AD[8] | GLOC IN[0] | PWMA PWMA[31] | CAT SYNC | EIC EXTINT[2] | CAT CSB[12] |
| 29 | PB09 | 41 | VDDIO | Normal I/O | TC1 B2 | USART3 CTS | USART3 CLK | | PWMA PWMA[32] | ACIFB ACBN[1] | EIC EXTINT[3] | CAT CSB[15] |
| 23 | PB10 | 42 | VDDIN | Normal I/O | TC1 CLK0 | USART1 TXD | USART3 CLK | GLOC OUT[1] | PWMA PWMA[33] | | EIC EXTINT[4] | CAT CSB[16] |
| 44 | PB11 | 43 | VDDIO | Normal I/O | TC1 CLK1 | USART1 RXD | | ADCIFB TRIGGER | PWMA PWMA[34] | CAT VDIVEN | EIC EXTINT[5] | CAT CSA[16] |
| 5 | PB12 | 44 | VDDIO | Normal I/O | TC1 CLK2 | | TWIMS1 TWALM | CAT SYNC | PWMA PWMA[35] | ACIFB ACBP[3] | SCIF GCLK[4] | CAT CSA[15] |

See Section 3.3 for a description of the various peripheral signals.

Refer to "Electrical Characteristics" on page 41 for a description of the electrical properties of the pin types used.

#### *3.2.1.1 TWI, 5V Tolerant, and SMBUS Pins*

Some Normal I/O pins offer TWI, 5V Tolerant, and SMBUS features. These features are only available when either of the TWI functions or the PWMAOD function in the PWMA are selected for these pins.

Refer to the "TWI Pin Characteristics(1)" on page 49 for a description of the electrical properties of the TWI, 5V Tolerant, and SMBUS pins.

### 3.2.2 Peripheral Functions

Each GPIO line can be assigned to one of several peripheral functions. The following table describes how the various peripheral functions are selected. The last listed function has priority in case multiple functions are enabled on the same pin.

**Table 3-2.** Peripheral Functions

| Function | Description |
|---|---|
| GPIO Controller Function multiplexing | GPIO and GPIO peripheral selection A to H |
| Nexus OCD AUX port connections | OCD trace system |
| aWire DATAOUT | aWire output in two-pin mode |
| JTAG port connections | JTAG debug port |
| Oscillators | OSC0, OSC32 |

### 3.2.3 JTAG Port Connections

If the JTAG is enabled, the JTAG will take control over a number of pins, irrespectively of the I/O Controller configuration.

**Table 3-3.** JTAG Pinout

| 48-pin | Pin Name | JTAG Pin |
|---|---|---|
| 11 | PA00 | TCK |
| 14 | PA01 | TMS |
| 13 | PA02 | TDO |
| 4 | PA03 | TDI |

### 3.2.4 Nexus OCD AUX Port Connections

If the OCD trace system is enabled, the trace system will take control over a number of pins, irrespectively of the I/O Controller configuration. Two different OCD trace pin mappings are possible, depending on the configuration of the OCD AXS register. For details, see the AVR32 UC Technical Reference Manual.

**Table 3-4.** Nexus OCD AUX Port Connections

| Pin | AXS=1 | AXS=0 |
|---|---|---|
| EVTI_N | PA05 | PB08 |
| MDO[5] | PA10 | PB00 |
| MDO[4] | PA18 | PB04 |
| MDO[3] | PA17 | PB05 |
| MDO[2] | PA16 | PB03 |

**Table 3-4.** Nexus OCD AUX Port Connections

| Pin | AXS=1 | AXS=0 |
|---|---|---|
| MDO[1] | PA15 | PB02 |
| MDO[0] | PA14 | PB09 |
| EVTO_N | PA04 | PA04 |
| MCKO | PA06 | PB01 |
| MSEO[1] | PA07 | PB11 |
| MSEO[0] | PA11 | PB12 |

### 3.2.5 Oscillator Pinout

The oscillators are not mapped to the normal GPIO functions and their muxings are controlled by registers in the System Control Interface (SCIF). Please refer to the SCIF chapter for more information about this.

**Table 3-5.** Oscillator Pinout

| 48-pin | Pin | Oscillator Function |
|---|---|---|
| 3 | PA08 | XIN0 |
| 46 | PA10 | XIN32 |
| 26 | PA13 | XIN32_2 |
| 2 | PA09 | XOUT0 |
| 47 | PA12 | XOUT32 |
| 25 | PA20 | XOUT32_2 |

### 3.2.6 Other Functions

The functions listed in Table 3-6 are not mapped to the normal GPIO functions. The aWire DATA pin will only be active after the aWire is enabled. The aWire DATAOUT pin will only be active after the aWire is enabled and the 2_PIN_MODE command has been sent. The WAKE_N pin is always enabled. Please refer to Section 6.1.4 on page 40 for constraints on the WAKE_N pin.

**Table 3-6.** Other Functions

| 48-pin | Pin | Function |
|---|---|---|
| 27 | PA11 | WAKE_N |
| 22 | RESET_N | aWire DATA |
| 11 | PA00 | aWire DATAOUT |

## 3.3 Signal Descriptions

The following table gives details on signal name classified by peripheral.

**Table 3-7.** Signal Descriptions List

| Signal Name | Function | Type | Active Level | Comments |
|---|---|---|---|---|
| **Analog Comparator Interface - ACIFB** | | | | |
| ACAN3 - ACAN0 | Negative inputs for comparators "A" | Analog | | |
| ACAP3 - ACAP0 | Positive inputs for comparators "A" | Analog | | |
| ACBN3 - ACBN0 | Negative inputs for comparators "B" | Analog | | |
| ACBP3 - ACBP0 | Positive inputs for comparators "B" | Analog | | |
| ACREFN | Common negative reference | Analog | | |
| **ADC Interface - ADCIFB** | | | | |
| AD8 - AD0 | Analog Signal | Analog | | |
| ADP1 - ADP0 | Drive Pin for resistive touch screen | Output | | |
| PRND | Pseudorandom output signal | Output | | |
| TRIGGER | External trigger | Input | | |
| **aWire - AW** | | | | |
| DATA | aWire data | I/O | | |
| DATAOUT | aWire data output for 2-pin mode | I/O | | |
| **Capacitive Touch Module - CAT** | | | | |
| CSA16 - CSA0 | Capacitive Sense A | I/O | | |
| CSB16 - CSB0 | Capacitive Sense B | I/O | | |
| DIS | Discharge current control | Analog | | |
| SMP | SMP signal | Output | | |
| SYNC | Synchronize signal | Input | | |
| VDIVEN | Voltage divider enable | Output | | |
| **External Interrupt Controller - EIC** | | | | |
| NMI | Non-Maskable Interrupt | Input | | |
| EXTINT5 - EXTINT1 | External interrupt | Input | | |
| **Glue Logic Controller - GLOC** | | | | |
| IN7 - IN0 | Inputs to lookup tables | Input | | |
| OUT1 - OUT0 | Outputs from lookup tables | Output | | |
| **JTAG module - JTAG** | | | | |
| TCK | Test Clock | Input | | |
| TDI | Test Data In | Input | | |
| TDO | Test Data Out | Output | | |

**Table 3-7.** Signal Descriptions List

| | | | | |
|---|---|---|---|---|
| TMS | Test Mode Select | Input | | |
| **Power Manager - PM** | | | | |
| RESET_N | Reset | Input | Low | |
| **Pulse Width Modulation Controller - PWMA** | | | | |
| PWMA35 - PWMA0 | PWMA channel waveforms | Output | | |
| PWMAOD35 - PWMAOD0 | PWMA channel waveforms, open drain mode | Output | | Not all channels support open drain mode |
| **System Control Interface - SCIF** | | | | |
| GCLK4 - GCLK0 | Generic Clock Output | Output | | |
| RC32OUT | RC32K output at startup | Output | | |
| XIN0 | Crystal 0 Input | Analog/ Digital | | |
| XIN32 | Crystal 32 Input (primary location) | Analog/ Digital | | |
| XIN32_2 | Crystal 32 Input (secondary location) | Analog/ Digital | | |
| XOUT0 | Crystal 0 Output | Analog | | |
| XOUT32 | Crystal 32 Output (primary location) | Analog | | |
| XOUT32_2 | Crystal 32 Output (secondary location) | Analog | | |
| **Serial Peripheral Interface - SPI** | | | | |
| MISO | Master In Slave Out | I/O | | |
| MOSI | Master Out Slave In | I/O | | |
| NPCS3 - NPCS0 | SPI Peripheral Chip Select | I/O | Low | |
| SCK | Clock | I/O | | |
| **Timer/Counter - TC0, TC1** | | | | |
| A0 | Channel 0 Line A | I/O | | |
| A1 | Channel 1 Line A | I/O | | |
| A2 | Channel 2 Line A | I/O | | |
| B0 | Channel 0 Line B | I/O | | |
| B1 | Channel 1 Line B | I/O | | |
| B2 | Channel 2 Line B | I/O | | |
| CLK0 | Channel 0 External Clock Input | Input | | |
| CLK1 | Channel 1 External Clock Input | Input | | |
| CLK2 | Channel 2 External Clock Input | Input | | |
| **Two-wire Interface - TWIMS0, TWIMS1** | | | | |
| TWALM | SMBus SMBALERT | I/O | Low | |
| TWCK | Two-wire Serial Clock | I/O | | |
| TWD | Two-wire Serial Data | I/O | | |

**Table 3-7.** Signal Descriptions List

| Universal Synchronous/Asynchronous Receiver/Transmitter - USART0, USART1, USART2, USART3 | | | | |
|---|---|---|---|---|
| CLK | Clock | I/O | | |
| CTS | Clear To Send | Input | Low | |
| RTS | Request To Send | Output | Low | |
| RXD | Receive Data | Input | | |
| TXD | Transmit Data | Output | | |

Note: 1. ADCIFB: AD3 does not exist.

**Table 3-8.** Signal Description List, Continued

| Signal Name | Function | Type | Active Level | Comments |
|---|---|---|---|---|
| **Power** | | | | |
| VDDCORE | Core Power Supply / Voltage Regulator Output | Power Input/Output | | 1.62V to 1.98V |
| VDDIO | I/O Power Supply | Power Input | | 1.62V to 3.6V. VDDIO should always be equal to or lower than VDDIN. |
| VDDANA | Analog Power Supply | Power Input | | 1.62V to 1.98V |
| ADVREFP | Analog Reference Voltage | Power Input | | 1.62V to 1.98V |
| VDDIN | Voltage Regulator Input | Power Input | | 1.62V to 3.6V (1) |
| GNDANA | Analog Ground | Ground | | |
| GND | Ground | Ground | | |
| **Auxiliary Port - AUX** | | | | |
| MCKO | Trace Data Output Clock | Output | | |
| MDO5 - MDO0 | Trace Data Output | Output | | |
| MSEO1 - MSEO0 | Trace Frame Control | Output | | |
| EVTI_N | Event In | Input | Low | |
| EVTO_N | Event Out | Output | Low | |
| **General Purpose I/O pin** | | | | |
| PA22 - PA00 | Parallel I/O Controller I/O Port 0 | I/O | | |
| PB12 - PB00 | Parallel I/O Controller I/O Port 1 | I/O | | |

1. See Section 6.1 on page 36

## 3.4 I/O Line Considerations

### 3.4.1 JTAG Pins

The JTAG is enabled if TCK is low while the RESET_N pin is released. The TCK, TMS, and TDI pins have pull-up resistors when JTAG is enabled. The TCK pin always has pull-up enabled during reset. The TDO pin is an output, driven at VDDIO, and has no pull-up resistor. The JTAG pins can be used as GPIO pins and multiplexed with peripherals when the JTAG is disabled. Please refer to Section 3.2.3 on page 11 for the JTAG port connections.

### 3.4.2 PA00

Note that PA00 is multiplexed with TCK. PA00 GPIO function must only be used as output in the application.

### 3.4.3 RESET_N Pin

The RESET_N pin is a schmitt input and integrates a permanent pull-up resistor to VDDIN. As the product integrates a power-on reset detector, the RESET_N pin can be left unconnected in case no reset from the system needs to be applied to the product.

The RESET_N pin is also used for the aWire debug protocol. When the pin is used for debugging, it must not be driven by external circuitry.

### 3.4.4 TWI Pins PA21/PB04/PB05

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with spike filtering. When used as GPIO pins or used for other peripherals, the pins have the same characteristics as other GPIO pins. Selected pins are also SMBus compliant (refer to Section 3.2 on page 9). As required by the SMBus specification, these pins provide no leakage path to ground when the AT32UC3L016/32/64 is powered down. This allows other devices on the SMBus to continue communicating even though the AT32UC3L016/32/64 is not powered.

After reset a TWI function is selected on these pins instead of the GPIO. Please refer to the GPIO Module Configuration chapter for details.

### 3.4.5 TWI Pins PA05/PA07/PA17

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with spike filtering. When used as GPIO pins or used for other peripherals, the pins have the same characteristics as other GPIO pins.

After reset a TWI function is selected on these pins instead of the GPIO. Please refer to the GPIO Module Configuration chapter for details.

### 3.4.6 GPIO Pins

All the I/O lines integrate a pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the GPIO Controller. After reset, I/O lines default as inputs with pull-up resistors disabled, except PA00. PA20 selects SCIF-RC32OUT (GPIO Function F) as default enabled after reset.

### 3.4.7 High-Drive Pins

The five pins PA02, PA06, PA08, PA09, and PB01 have high-drive output capabilities. Refer to Section 7. on page 41 for electrical characteristics.

### 3.4.8 RC32OUT Pin

#### *3.4.8.1 Clock output at startup*

After power-up, the clock generated by the 32kHz RC oscillator (RC32K) will be output on PA20, even when the device is still reset by the Power-On Reset Circuitry. This clock can be used by the system to start other devices or to clock a switching regulator to rise the power supply voltage up to an acceptable value.

The clock will be available on PA20, but will be disabled if one of the following conditions are true:

- PA20 is configured to use a GPIO function other than F (SCIF-RC32OUT)
- PA20 is configured as a General Purpose Input/Output (GPIO)
- The bit FRC32 in the Power Manager PPCR register is written to zero (refer to the Power Manager chapter)

The maximum amplitude of the clock signal will be defined by VDDIN.

Once the RC32K output on PA20 is disabled it can never be enabled again.

#### *3.4.8.2 XOUT32_2 function*

PA20 selects RC32OUT as default enabled after reset. This function is not automatically disabled when the user enables the XOUT32_2 function on PA20. This disturbs the oscillator and may result in the wrong frequency. To avoid this, RC32OUT must be disabled when XOUT32_2 is enabled.

### 3.4.9 ADC Input Pins

These pins are regular I/O pins powered from the VDDIO. However, when these pins are used for ADC inputs, the voltage applied to the pin must not exceed 1.98V. Internal circuitry ensures that the pin cannot be used as an analog input pin when the I/O drives to VDD. When the pins are not used for ADC inputs, the pins may be driven to the full I/O voltage range.

# 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 able to exploit code optimization features, both for size and speed.

In order to reduce code size to a minimum, some instructions have multiple addressing modes. As an example, instructions with immediates often have a compact format with a smaller immediate, and an extended format with a larger immediate. In this way, the compiler is able to use the format giving the smallest code size.

Another feature of the instruction set is that frequently used instructions, like add, have a compact format with two operands as well as an extended format with three operands. The larger format increases performance, allowing an addition and a data move in the same instruction in a

single cycle. Load and store instructions have several different formats in order to reduce code size and speed up execution.

The register file is organized as sixteen 32-bit registers and includes the Program Counter, the Link Register, and the Stack Pointer. In addition, register R12 is designed to hold return values from function calls and is used implicitly by some instructions.

## 4.3 The AVR32UC CPU

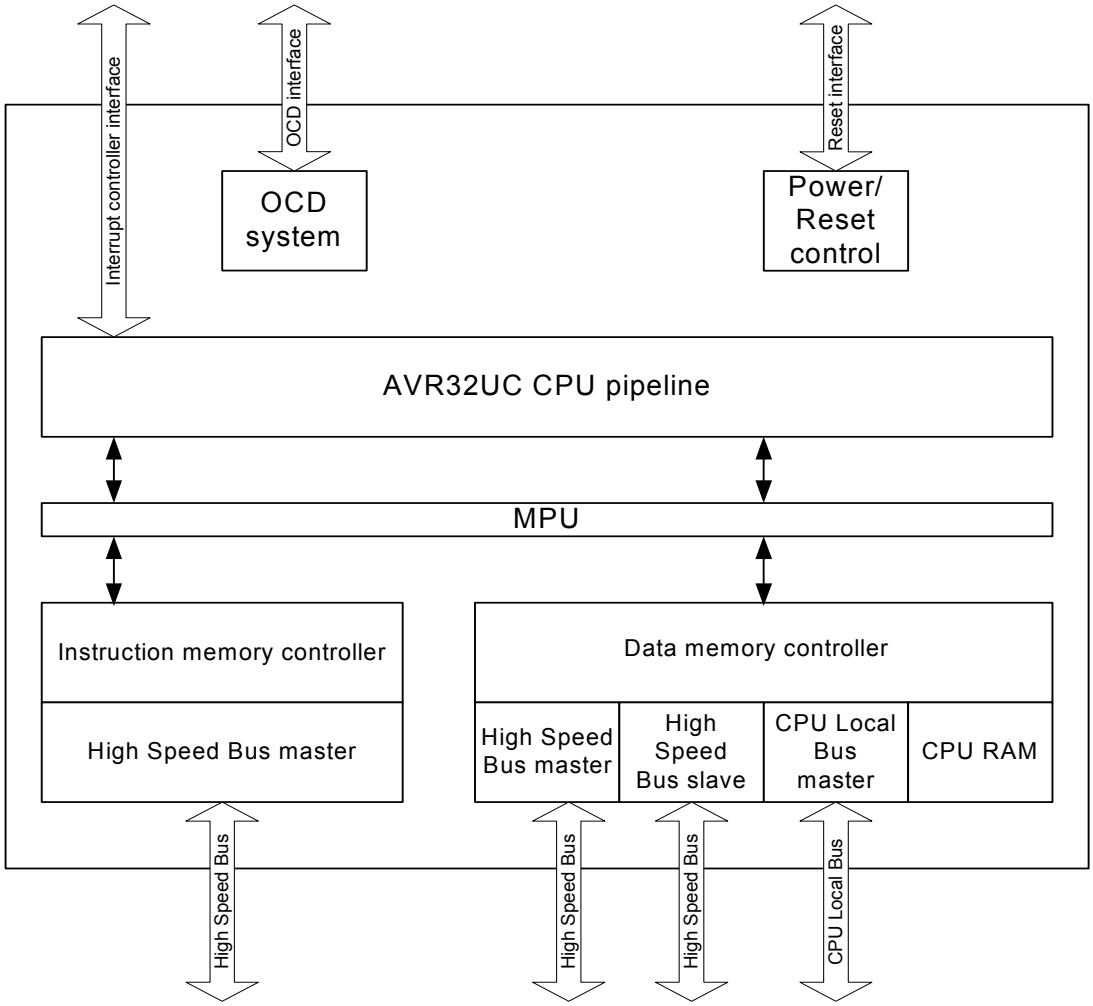
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced On-Chip Debug (OCD) system, no caches, and a Memory Protection Unit (MPU). Java acceleration hardware is not implemented.

AVR32UC provides three memory interfaces, one High Speed Bus master for instruction fetch, one High Speed Bus master for data access, and one High Speed Bus slave interface allowing other bus masters to access data RAMs internal to the CPU. Keeping data RAMs internal to the CPU allows fast access to the RAMs, reduces latency, and guarantees deterministic timing. Also, power consumption is reduced by not needing a full High Speed Bus access for memory accesses. A dedicated data RAM interface is provided for communicating with the internal data RAMs.

A local bus interface is provided for connecting the CPU to device-specific high-speed systems, such as floating-point units and I/O controller ports. This local bus has to be enabled by writing a one to the LOCEN bit in the CPUCR system register. The local bus is able to transfer data between the CPU and the local bus slave in a single clock cycle. The local bus has a dedicated memory range allocated to it, and data transfers are performed using regular load and store instructions. Details on which devices that are mapped into the local bus space is given in the CPU Local Bus section in the Memories chapter.

Figure 4-1 on page 20 displays the contents of AVR32UC.

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



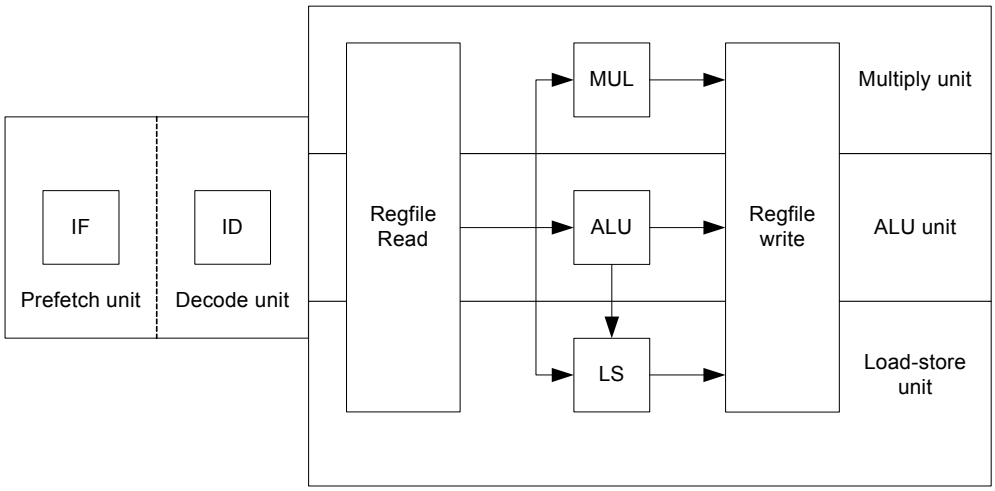

### 4.3.1 Pipeline Overview

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.

**Figure 4-2.** The AVR32UC Pipeline




### 4.3.2 AVR32A Microarchitecture Compliance

AVR32UC implements an AVR32A microarchitecture. The AVR32A microarchitecture is targeted at cost-sensitive, lower-end applications like smaller microcontrollers. This microarchitecture does not provide dedicated hardware registers for shadowing of register file registers in interrupt contexts. Additionally, it does not provide hardware registers for the return address registers and return status registers. Instead, all this information is stored on the system stack. This saves chip area at the expense of slower interrupt handling.

#### *4.3.2.1 Interrupt Handling*

Upon interrupt initiation, registers R8-R12 are automatically pushed to the system stack. These registers are pushed regardless of the priority level of the pending interrupt. The return address and status register are also automatically pushed to stack. The interrupt handler can therefore use R8-R12 freely. Upon interrupt completion, the old R8-R12 registers and status register are restored, and execution continues at the return address stored popped from stack.

The stack is also used to store the status register and return address for exceptions and *scall*. Executing the *rete* or *rets* instruction at the completion of an exception or system call will pop this status register and continue execution at the popped return address.

#### *4.3.2.2 Java Support*

AVR32UC does not provide Java hardware acceleration.

#### *4.3.2.3 Memory Protection*

The MPU allows the user to check all memory accesses for privilege violations. If an access is attempted to an illegal memory address, the access is aborted and an exception is taken. The MPU in AVR32UC is specified in the AVR32UC Technical Reference manual.

#### *4.3.2.4 Unaligned Reference Handling*

AVR32UC does not support unaligned accesses, except for doubleword accesses. AVR32UC is able to perform word-aligned *st.d* and *ld.d*. Any other unaligned memory access will cause an

address exception. Doubleword-sized accesses with word-aligned pointers will automatically be performed as two word-sized accesses.

The following table shows the instructions with support for unaligned addresses. All other instructions require aligned addresses.

**Table 4-1.** Instructions with Unaligned Reference Support

| Instruction | Supported Alignment |
|---|---|
| ld.d | Word |
| st.d | Word |

#### 4.3.2.5 *Unimplemented Instructions*

The following instructions are unimplemented in AVR32UC, and will cause an Unimplemented Instruction Exception if executed:

- All SIMD instructions
- All coprocessor instructions if no coprocessors are present
- retj, incjosp, popjc, pushjc
- tlbr, tlbs, tlbw
- cache

#### 4.3.2.6 *CPU and Architecture Revision*

Three major revisions of the AVR32UC CPU currently exist. The device described in this datasheet uses CPU revision 3.

The Architecture Revision field in the CONFIG0 system register identifies which architecture revision is implemented in a specific device.

AVR32UC CPU revision 3 is fully backward-compatible with revisions 1 and 2, ie. code compiled for revision 1 or 2 is binary-compatible with revision 3 CPUs.

## 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

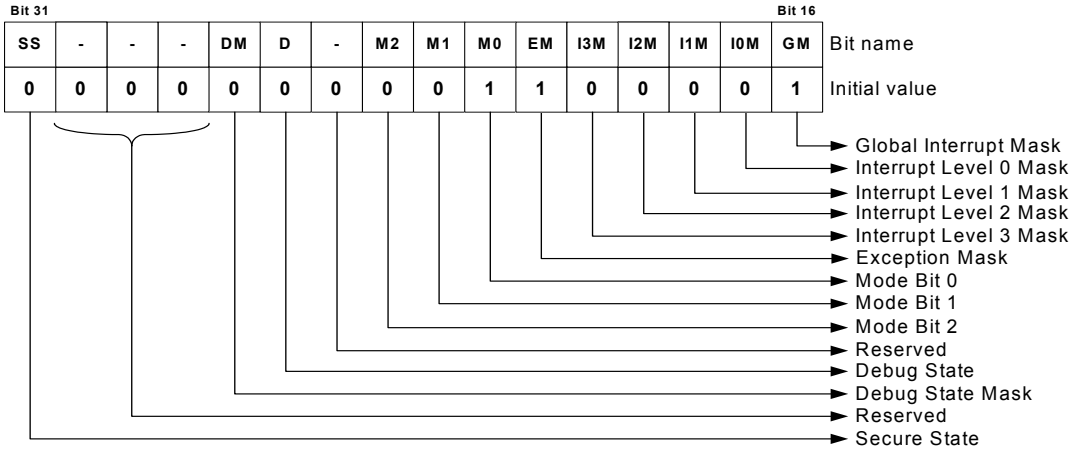
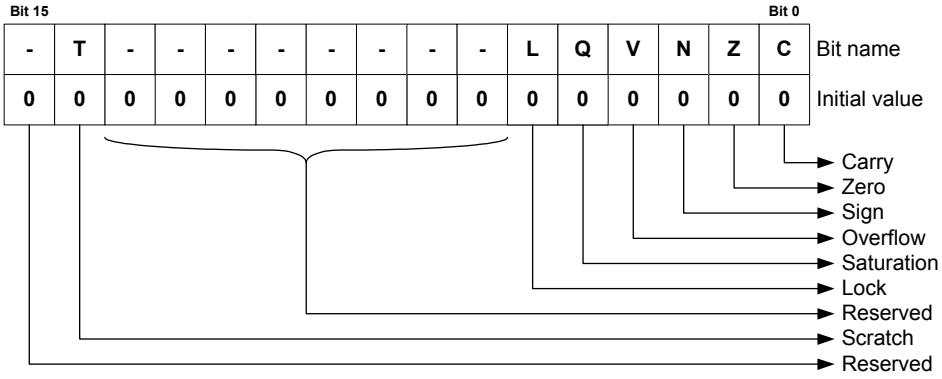


**Figure 4-5.** The Status Register Low Halfword




### 4.4.3 Processor States

#### *4.4.3.1 Normal RISC State*

The AVR32 processor supports several different execution contexts as shown in Table 4-2.

**Table 4-2.** Overview of Execution Modes, their Priorities and Privilege Levels.

| Priority | Mode | Security | Description |
|---|---|---|---|
| 1 | Non Maskable Interrupt | Privileged | Non Maskable high priority interrupt mode |
| 2 | Exception | Privileged | Execute exceptions |
| 3 | Interrupt 3 | Privileged | General purpose interrupt mode |
| 4 | Interrupt 2 | Privileged | General purpose interrupt mode |
| 5 | Interrupt 1 | Privileged | General purpose interrupt mode |
| 6 | Interrupt 0 | Privileged | General purpose interrupt mode |
| N/A | Supervisor | Privileged | Runs supervisor calls |
| N/A | Application | Unprivileged | Normal program execution mode |

Mode changes can be made under software control, or can be caused by external interrupts or exception processing. A mode can be interrupted by a higher priority mode, but never by one with lower priority. Nested exceptions can be supported with a minimal software overhead.

When running an operating system on the AVR32, user processes will typically execute in the application mode. The programs executed in this mode are restricted from executing certain instructions. Furthermore, most system registers together with the upper halfword of the status register cannot be accessed. Protected memory areas are also not available. All other operating modes are privileged and are collectively called System Modes. They have full access to all privileged and unprivileged resources. After a reset, the processor will be in supervisor mode.

#### *4.4.3.2 Debug State*

The AVR32 can be set in a debug state, which allows implementation of software monitor routines that can read out and alter system information for use during application development. This implies that all system and application registers, including the status registers and program counters, are accessible in debug state. The privileged instructions are also available.

All interrupt levels are by default disabled when debug state is entered, but they can individually be switched on by the monitor routine by clearing the respective mask bit in the status register.

Debug state can be entered as described in the *AVR32UC Technical Reference Manual*.

Debug state is exited by the *retd* instruction.

#### 4.4.3.3 *Secure State*

The AVR32 can be set in a secure state, that allows a part of the code to execute in a state with higher security levels. The rest of the code can not access resources reserved for this secure code. Secure State is used to implement FlashVault technology. Refer to the *AVR32UC Technical Reference Manual* for details.

### 4.4.4 System Registers

The system registers are placed outside of the virtual memory space, and are only accessible using the privileged *mfsr* and *mtsr* instructions. The table below lists the system registers specified in the AVR32 architecture, some of which are unused in AVR32UC. The programmer is responsible for maintaining correct sequencing of any instructions following a *mtsr* instruction. For detail on the system registers, refer to the *AVR32UC Technical Reference Manual*.

**Table 4-3.** System Registers

| Reg # | Address | Name | Function |
|---|---|---|---|
| 0 | 0 | SR | Status Register |
| 1 | 4 | EVBA | Exception Vector Base Address |
| 2 | 8 | ACBA | Application Call Base Address |
| 3 | 12 | CPUCR | CPU Control Register |
| 4 | 16 | ECR | Exception Cause Register |
| 5 | 20 | RSR_SUP | Unused in AVR32UC |
| 6 | 24 | RSR_INT0 | Unused in AVR32UC |
| 7 | 28 | RSR_INT1 | Unused in AVR32UC |
| 8 | 32 | RSR_INT2 | Unused in AVR32UC |
| 9 | 36 | RSR_INT3 | Unused in AVR32UC |
| 10 | 40 | RSR_EX | Unused in AVR32UC |
| 11 | 44 | RSR_NMI | Unused in AVR32UC |
| 12 | 48 | RSR_DBG | Return Status Register for Debug mode |
| 13 | 52 | RAR_SUP | Unused in AVR32UC |
| 14 | 56 | RAR_INT0 | Unused in AVR32UC |
| 15 | 60 | RAR_INT1 | Unused in AVR32UC |
| 16 | 64 | RAR_INT2 | Unused in AVR32UC |
| 17 | 68 | RAR_INT3 | Unused in AVR32UC |
| 18 | 72 | RAR_EX | Unused in AVR32UC |
| 19 | 76 | RAR_NMI | Unused in AVR32UC |
| 20 | 80 | RAR_DBG | Return Address Register for Debug mode |
| 21 | 84 | JECR | Unused in AVR32UC |
| 22 | 88 | JOSP | Unused in AVR32UC |
| 23 | 92 | JAVA_LV0 | Unused in AVR32UC |

**Table 4-3.** System Registers (Continued)

| Reg # | Address | Name | Function |
|---|---|---|---|
| 24 | 96 | JAVA_LV1 | Unused in AVR32UC |
| 25 | 100 | JAVA_LV2 | Unused in AVR32UC |
| 26 | 104 | JAVA_LV3 | Unused in AVR32UC |
| 27 | 108 | JAVA_LV4 | Unused in AVR32UC |
| 28 | 112 | JAVA_LV5 | Unused in AVR32UC |
| 29 | 116 | JAVA_LV6 | Unused in AVR32UC |
| 30 | 120 | JAVA_LV7 | Unused in AVR32UC |
| 31 | 124 | JTBA | Unused in AVR32UC |
| 32 | 128 | JBCR | Unused in AVR32UC |
| 33-63 | 132-252 | Reserved | Reserved for future use |
| 64 | 256 | CONFIG0 | Configuration register 0 |
| 65 | 260 | CONFIG1 | Configuration register 1 |
| 66 | 264 | COUNT | Cycle Counter register |
| 67 | 268 | COMPARE | Compare register |
| 68 | 272 | TLBEHI | Unused in AVR32UC |
| 69 | 276 | TLBELO | Unused in AVR32UC |
| 70 | 280 | PTBR | Unused in AVR32UC |
| 71 | 284 | TLBEAR | Unused in AVR32UC |
| 72 | 288 | MMUCR | Unused in AVR32UC |
| 73 | 292 | TLBARLO | Unused in AVR32UC |
| 74 | 296 | TLBARHI | Unused in AVR32UC |
| 75 | 300 | PCCNT | Unused in AVR32UC |
| 76 | 304 | PCNT0 | Unused in AVR32UC |
| 77 | 308 | PCNT1 | Unused in AVR32UC |
| 78 | 312 | PCCR | Unused in AVR32UC |
| 79 | 316 | BEAR | Bus Error Address Register |
| 80 | 320 | MPUAR0 | MPU Address Register region 0 |
| 81 | 324 | MPUAR1 | MPU Address Register region 1 |
| 82 | 328 | MPUAR2 | MPU Address Register region 2 |
| 83 | 332 | MPUAR3 | MPU Address Register region 3 |
| 84 | 336 | MPUAR4 | MPU Address Register region 4 |
| 85 | 340 | MPUAR5 | MPU Address Register region 5 |
| 86 | 344 | MPUAR6 | MPU Address Register region 6 |
| 87 | 348 | MPUAR7 | MPU Address Register region 7 |
| 88 | 352 | MPUPSR0 | MPU Privilege Select Register region 0 |
| 89 | 356 | MPUPSR1 | MPU Privilege Select Register region 1 |

**Table 4-3.** System Registers (Continued)

| Reg # | Address | Name | Function |
|---|---|---|---|
| 90 | 360 | MPUPSR2 | MPU Privilege Select Register region 2 |
| 91 | 364 | MPUPSR3 | MPU Privilege Select Register region 3 |
| 92 | 368 | MPUPSR4 | MPU Privilege Select Register region 4 |
| 93 | 372 | MPUPSR5 | MPU Privilege Select Register region 5 |
| 94 | 376 | MPUPSR6 | MPU Privilege Select Register region 6 |
| 95 | 380 | MPUPSR7 | MPU Privilege Select Register region 7 |
| 96 | 384 | MPUCRA | Unused in this version of AVR32UC |
| 97 | 388 | MPUCRB | Unused in this version of AVR32UC |
| 98 | 392 | MPUBRA | Unused in this version of AVR32UC |
| 99 | 396 | MPUBRB | Unused in this version of AVR32UC |
| 100 | 400 | MPUAPRA | MPU Access Permission Register A |
| 101 | 404 | MPUAPRB | MPU Access Permission Register B |
| 102 | 408 | MPUCR | MPU Control Register |
| 103 | 412 | SS_STATUS | Secure State Status Register |
| 104 | 416 | SS_ADRF | Secure State Address Flash Register |
| 105 | 420 | SS_ADRR | Secure State Address RAM Register |
| 106 | 424 | SS_ADR0 | Secure State Address 0 Register |
| 107 | 428 | SS_ADR1 | Secure State Address 1 Register |
| 108 | 432 | SS_SP_SYS | Secure State Stack Pointer System Register |
| 109 | 436 | SS_SP_APP | Secure State Stack Pointer Application Register |
| 110 | 440 | SS_RAR | Secure State Return Address Register |
| 111 | 444 | SS_RSR | Secure State Return Status Register |
| 112-191 | 448-764 | Reserved | Reserved for future use |
| 192-255 | 768-1020 | IMPL | IMPLEMENTATION DEFINED |

## 4.5 Exceptions and Interrupts

In the AVR32 architecture, events are used as a common term for exceptions and interrupts. AVR32UC incorporates a powerful event handling scheme. The different event sources, like Illegal Op-code and interrupt requests, have different priority levels, ensuring a well-defined behavior when multiple events are received simultaneously. Additionally, pending events of a higher priority class may preempt handling of ongoing events of a lower priority class.

When an event occurs, the execution of the instruction stream is halted, and execution is passed to an event handler at an address specified in Table 4-4 on page 31. Most of the handlers are placed sequentially in the code space starting at the address specified by EVBA, with four bytes between each handler. This gives ample space for a jump instruction to be placed there, jumping to the event routine itself. A few critical handlers have larger spacing between them, allowing the entire event routine to be placed directly at the address specified by the EVBA-relative offset generated by hardware. All interrupt sources have autovectored interrupt service routine (ISR) addresses. This allows the interrupt controller to directly specify the ISR address as an address

relative to EVBA. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes. The target address of the event handler is calculated as (EVBA | event_handler_offset), not (EVBA + event_handler_offset), so EVBA and exception code segments must be set up appropriately. The same mechanisms are used to service all different types of events, including interrupt requests, yielding a uniform event handling scheme.

An interrupt controller does the priority handling of the interrupts and provides the autovector offset to the CPU.

### 4.5.1 System Stack Issues

Event handling in AVR32UC uses the system stack pointed to by the system stack pointer, SP_SYS, for pushing and popping R8-R12, LR, status register, and return address. Since event code may be timing-critical, SP_SYS should point to memory addresses in the IRAM section, since the timing of accesses to this memory section is both fast and deterministic.

The user must also make sure that the system stack is large enough so that any event is able to push the required registers to stack. If the system stack is full, and an event occurs, the system will enter an UNDEFINED state.

### 4.5.2 Exceptions and Interrupt Requests

When an event other than *scall* or debug request is received by the core, the following actions are performed atomically:

1. The pending event will not be accepted if it is masked. The I3M, I2M, I1M, I0M, EM, and GM bits in the Status Register are used to mask different events. Not all events can be masked. A few critical events (NMI, Unrecoverable Exception, TLB Multiple Hit, and Bus Error) can not be masked. When an event is accepted, hardware automatically sets the mask bits corresponding to all sources with equal or lower priority. This inhibits acceptance of other events of the same or lower priority, except for the critical events listed above. Software may choose to clear some or all of these bits after saving the necessary state if other priority schemes are desired. It is the event source's responsability to ensure that their events are left pending until accepted by the CPU.
2. When a request is accepted, the Status Register and Program Counter of the current context is stored to the system stack. If the event is an INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also automatically stored to stack. Storing the Status Register ensures that the core is returned to the previous execution mode when the current event handling is completed. When exceptions occur, both the EM and GM bits are set, and the application may manually enable nested exceptions if desired by clearing the appropriate bit. Each exception handler has a dedicated handler address, and this address uniquely identifies the exception source.
3. The Mode bits are set to reflect the priority of the accepted event, and the correct register file bank is selected. The address of the event handler, as shown in Table 4-4 on page 31, is loaded into the Program Counter.

The execution of the event handler routine then continues from the effective address calculated.

The *rete* instruction signals the end of the event. When encountered, the Return Status Register and Return Address Register are popped from the system stack and restored to the Status Register and Program Counter. If the *rete* instruction returns from INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also popped from the system stack. The restored Status Register contains information allowing the core to resume operation in the previous execution mode. This concludes the event handling.

### 4.5.3 Supervisor Calls

The AVR32 instruction set provides a supervisor mode call instruction. The *scall* instruction is designed so that privileged routines can be called from any context. This facilitates sharing of code between different execution modes. The *scall* mechanism is designed so that a minimal execution cycle overhead is experienced when performing supervisor routine calls from time-critical event handlers.

The *scall* instruction behaves differently depending on which mode it is called from. The behaviour is detailed in the instruction set reference. In order to allow the *scall* routine to return to the correct context, a return from supervisor call instruction, *rets*, is implemented. In the AVR32UC CPU, *scall* and *rets* uses the system stack to store the return address and the status register.

### 4.5.4 Debug Requests

The AVR32 architecture defines a dedicated Debug mode. When a debug request is received by the core, Debug mode is entered. Entry into Debug mode can be masked by the DM bit in the status register. Upon entry into Debug mode, hardware sets the SR.D bit and jumps to the Debug Exception handler. By default, Debug mode executes in the exception context, but with dedicated Return Address Register and Return Status Register. These dedicated registers remove the need for storing this data to the system stack, thereby improving debuggability. The Mode bits in the Status Register can freely be manipulated in Debug mode, to observe registers in all contexts, while retaining full privileges.

Debug mode is exited by executing the *retd* instruction. This returns to the previous context.

### 4.5.5 Entry Points for Events

Several different event handler entry points exist. In AVR32UC, the reset address is 0x80000000. This places the reset address in the boot flash memory area.

TLB miss exceptions and *scall* have a dedicated space relative to EVBA where their event handler can be placed. This speeds up execution by removing the need for a jump instruction placed at the program address jumped to by the event hardware. All other exceptions have a dedicated event routine entry point located relative to EVBA. The handler routine address identifies the exception source directly.

AVR32UC uses the ITLB and DTLB protection exceptions to signal a MPU protection violation. ITLB and DTLB miss exceptions are used to signal that an access address did not map to any of the entries in the MPU. TLB multiple hit exception indicates that an access address did map to multiple TLB entries, signalling an error.

All interrupt requests have entry points located at an offset relative to EVBA. This autovector offset is specified by an interrupt controller. The programmer must make sure that none of the autovector offsets interfere with the placement of other code. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes.

Special considerations should be made when loading EVBA with a pointer. Due to security considerations, the event handlers should be located in non-writeable flash memory, or optionally in a privileged memory protection region if an MPU is present.

If several events occur on the same instruction, they are handled in a prioritized way. The priority ordering is presented in Table 4-4 on page 31. If events occur on several instructions at different locations in the pipeline, the events on the oldest instruction are always handled before any events on any younger instruction, even if the younger instruction has events of higher priority

than the oldest instruction. An instruction B is younger than an instruction A if it was sent down the pipeline later than A.

The addresses and priority of simultaneous events are shown in Table 4-4 on page 31. Some of the exceptions are unused in AVR32UC since it has no MMU, coprocessor interface, or floating-point unit.

**Table 4-4.** Priority and Handler Addresses for Events

| Priority | Handler Address | Name | Event source | Stored Return Address |
|---|---|---|---|---|
| 1 | 0x80000000 | Reset | External input | Undefined |
| 2 | Provided by OCD system | OCD Stop CPU | OCD system | First non-completed instruction |
| 3 | EVBA+0x00 | Unrecoverable exception | Internal | PC of offending instruction |
| 4 | EVBA+0x04 | TLB multiple hit | MPU | PC of offending instruction |
| 5 | EVBA+0x08 | Bus error data fetch | Data bus | First non-completed instruction |
| 6 | EVBA+0x0C | Bus error instruction fetch | Data bus | First non-completed instruction |
| 7 | EVBA+0x10 | NMI | External input | First non-completed instruction |
| 8 | Autovectored | Interrupt 3 request | External input | First non-completed instruction |
| 9 | Autovectored | Interrupt 2 request | External input | First non-completed instruction |
| 10 | Autovectored | Interrupt 1 request | External input | First non-completed instruction |
| 11 | Autovectored | Interrupt 0 request | External input | First non-completed instruction |
| 12 | EVBA+0x14 | Instruction Address | CPU | PC of offending instruction |
| 13 | EVBA+0x50 | ITLB Miss | MPU | PC of offending instruction |
| 14 | EVBA+0x18 | ITLB Protection | MPU | PC of offending instruction |
| 15 | EVBA+0x1C | Breakpoint | OCD system | First non-completed instruction |
| 16 | EVBA+0x20 | Illegal Opcode | Instruction | PC of offending instruction |
| 17 | EVBA+0x24 | Unimplemented instruction | Instruction | PC of offending instruction |
| 18 | EVBA+0x28 | Privilege violation | Instruction | PC of offending instruction |
| 19 | EVBA+0x2C | Floating-point | UNUSED | |
| 20 | EVBA+0x30 | Coprocessor absent | Instruction | PC of offending instruction |
| 21 | EVBA+0x100 | Supervisor call | Instruction | PC(Supervisor Call) +2 |
| 22 | EVBA+0x34 | Data Address (Read) | CPU | PC of offending instruction |
| 23 | EVBA+0x38 | Data Address (Write) | CPU | PC of offending instruction |
| 24 | EVBA+0x60 | DTLB Miss (Read) | MPU | PC of offending instruction |
| 25 | EVBA+0x70 | DTLB Miss (Write) | MPU | PC of offending instruction |
| 26 | EVBA+0x3C | DTLB Protection (Read) | MPU | PC of offending instruction |
| 27 | EVBA+0x40 | DTLB Protection (Write) | MPU | PC of offending instruction |
| 28 | EVBA+0x44 | DTLB Modified | UNUSED | |

# 5. Memories

## 5.1 Embedded Memories

- **Internal high-speed flash**
  - **64 Kbytes (AT32UC3L064)**
  - **32 Kbytes (AT32UC3L032)**
  - **16 Kbytes (AT32UC3L016)**
    - **0 wait state access at up to 25 mhz in worst case conditions**
    - **1 wait state access at up to 50 mhz in worst case conditions**
    - **Pipelined flash architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access**
    - **Pipelined flash architecture typically reduces the cycle penalty of 1 wait state operation to only 8% compared to 0 wait state operation**
    - **100 000 write cycles, 15-year data retention capability**
    - **Sector lock capabilities, bootloader protection, security bit**
    - **32 fuses, erased during chip erase**
    - **User page for data to be preserved during chip erase**
- **Internal high-speed SRAM, single-cycle access at full speed**
  - **16 Kbytes (AT32UC3L064, AT32UC3L032)**
  - **8 Kbytes (AT32UC3L016)**

## 5.2 Physical Memory Map

The system bus is implemented as a bus matrix. All system bus addresses are fixed, and they are never remapped in any way, not even in boot. Note that AVR32 UC CPU uses unsegmented translation, as described in the AVR32 Architecture Manual. The 32-bit physical address space is mapped as follows:

**Table 5-1.** AT32UC3L016/32/64 Physical Memory Map

| Device | Start Address | Size | | |
|---|---|---|---|---|
| | | AT32UC3L064 | AT32UC3L032 | AT32UC3L016 |
| Embedded SRAM | 0x00000000 | 16 Kbytes | 16 Kbytes | 8 Kbytes |
| Embedded Flash | 0x80000000 | 64 Kbytes | 32 Kbytes | 16 Kbytes |
| SAU Channels | 0x90000000 | 256 bytes | 256 bytes | 256 bytes |
| HSB-PB Bridge B | 0xFFFE0000 | 64 Kbytes | 64 Kbytes | 64 Kbytes |
| HSB-PB Bridge A | 0xFFFF0000 | 64 Kbytes | 64 Kbytes | 64 Kbytes |

**Table 5-2.** Flash Memory Parameters

| Part Number | Flash Size (*FLASH_PW*) | Number of pages (*FLASH_P*) | Page size (*FLASH_W*) |
|---|---|---|---|
| AT32UC3L064 | 64 Kbytes | 256 | 256 bytes |
| AT32UC3L032 | 32 Kbytes | 128 | 256 bytes |
| AT32UC3L016 | 16 Kbytes | 64 | 256 bytes |

## 5.3 Peripheral Address Map

**Table 5-3.** Peripheral Address Mapping

| Address | | Peripheral Name |
|---|---|---|
| 0xFFFE0000 | FLASHCDW | Flash Controller - FLASHCDW |
| 0xFFFE0400 | HMATRIX | HSB Matrix - HMATRIX |
| 0xFFFE0800 | SAU | Secure Access Unit - SAU |
| 0xFFFF0000 | PDCA | Peripheral DMA Controller - PDCA |
| 0xFFFF1000 | INTC | Interrupt controller - INTC |
| 0xFFFF1400 | PM | Power Manager - PM |
| 0xFFFF1800 | SCIF | System Control Interface - SCIF |
| 0xFFFF1C00 | AST | Asynchronous Timer - AST |
| 0xFFFF2000 | WDT | Watchdog Timer - WDT |
| 0xFFFF2400 | EIC | External Interrupt Controller - EIC |
| 0xFFFF2800 | FREQM | Frequency Meter - FREQM |
| 0xFFFF2C00 | GPIO | General Purpose Input/Output Controller - GPIO |
| 0xFFFF3000 | USART0 | Universal Synchronous/Asynchronous Receiver/Transmitter - USART0 |
| 0xFFFF3400 | USART1 | Universal Synchronous/Asynchronous Receiver/Transmitter - USART1 |
| 0xFFFF3800 | USART2 | Universal Synchronous/Asynchronous Receiver/Transmitter - USART2 |
| 0xFFFF3C00 | USART3 | Universal Synchronous/Asynchronous Receiver/Transmitter - USART3 |
| 0xFFFF4000 | SPI | Serial Peripheral Interface - SPI |
| 0xFFFF4400 | TWIM0 | Two-wire Master Interface - TWIM0 |

**Table 5-3.** Peripheral Address Mapping

| | | |
|---|---|---|
| **0xFFFF4800** | TWIM1 | Two-wire Master Interface - TWIM1 |
| **0xFFFF4C00** | TWIS0 | Two-wire Slave Interface - TWIS0 |
| **0xFFFF5000** | TWIS1 | Two-wire Slave Interface - TWIS1 |
| **0xFFFF5400** | PWMA | Pulse Width Modulation Controller - PWMA |
| **0xFFFF5800** | TC0 | Timer/Counter - TC0 |
| **0xFFFF5C00** | TC1 | Timer/Counter - TC1 |
| **0xFFFF6000** | ADCIFB | ADC Interface - ADCIFB |
| **0xFFFF6400** | ACIFB | Analog Comparator Interface - ACIFB |
| **0xFFFF6800** | CAT | Capacitive Touch Module - CAT |
| **0xFFFF6C00** | GLOC | Glue Logic Controller - GLOC |
| **0xFFFF7000** | AW | aWire - AW |

## 5.4 CPU Local Bus Mapping

Some of the registers in the GPIO module are mapped onto the CPU local bus, in addition to being mapped on the Peripheral Bus. These registers can therefore be reached both by accesses on the Peripheral Bus, and by accesses on the local bus.

Mapping these registers on the local bus allows cycle-deterministic toggling of GPIO pins since the CPU and GPIO are the only modules connected to this bus. Also, since the local bus runs at CPU speed, one write or read operation can be performed per clock cycle to the local bus-mapped GPIO registers.

The following GPIO registers are mapped on the local bus:

**Table 5-4.** Local Bus Mapped GPIO Registers

| Port | Register | Mode | Local Bus Address | Access |
|---|---|---|---|---|
| 0 | Output Driver Enable Register (ODER) | WRITE | 0x40000040 | Write-only |
| | | SET | 0x40000044 | Write-only |
| | | CLEAR | 0x40000048 | Write-only |
| | | TOGGLE | 0x4000004C | Write-only |
| | Output Value Register (OVR) | WRITE | 0x40000050 | Write-only |
| | | SET | 0x40000054 | Write-only |
| | | CLEAR | 0x40000058 | Write-only |
| | | TOGGLE | 0x4000005C | Write-only |
| | Pin Value Register (PVR) | - | 0x40000060 | Read-only |
| 1 | Output Driver Enable Register (ODER) | WRITE | 0x40000140 | Write-only |
| | | SET | 0x40000144 | Write-only |
| | | CLEAR | 0x40000148 | Write-only |
| | | TOGGLE | 0x4000014C | Write-only |
| | Output Value Register (OVR) | WRITE | 0x40000150 | Write-only |
| | | SET | 0x40000154 | Write-only |
| | | CLEAR | 0x40000158 | Write-only |
| | | TOGGLE | 0x4000015C | Write-only |
| | Pin Value Register (PVR) | - | 0x40000160 | Read-only |

# 6. Supply and Startup Considerations

## 6.1 Supply Considerations

### 6.1.1 Power Supplies

The AT32UC3L016/32/64 has several types of power supply pins:

- VDDIO: Powers I/O lines. Voltage is 1.8 to 3.3V nominal.
- VDDIN: Powers I/O lines and the internal regulator. Voltage is 1.8 to 3.3V nominal.
- VDDANA: Powers the ADC. Voltage is 1.8V nominal.
- VDDCORE: Powers the core, memories, and peripherals. Voltage is 1.8V nominal.

The ground pins GND are common to VDDCORE, VDDIO, and VDDIN. The ground pin for VDDANA is GNDANA.

When VDDCORE is not connected to VDDIN, the VDDIN voltage must be higher than 1.98V.

Refer to Section 7. on page 41 for power consumption on the various supply pins.

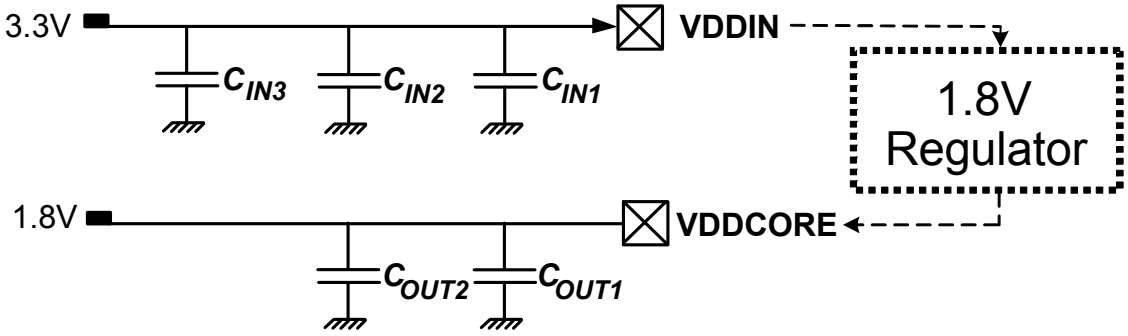
For decoupling recommendations for the different power supplies, please refer to the schematic checklist.

### 6.1.2 Voltage Regulator

The AT32UC3L016/32/64 embeds a voltage regulator that converts from 3.3V nominal to 1.8V with a load of up to 60mA. The regulator supplies the output voltage on VDDCORE. The regulator may only be used to drive internal circuitry in the device. VDDCORE should be externally connected to the 1.8V domains. See Section 6.1.3 for regulator connection figures.

Adequate output supply decoupling is mandatory for VDDCORE to reduce ripple and avoid oscillations. The best way to achieve this is to use two capacitors in parallell between VDDCORE and GND as close to the device as possible. Please refer to Section 7.8.1 on page 55 for decoupling capacitors values and regulator characteristics.

**Figure 6-1.** Supply Decoupling




### 6.1.3 Regulator Connection

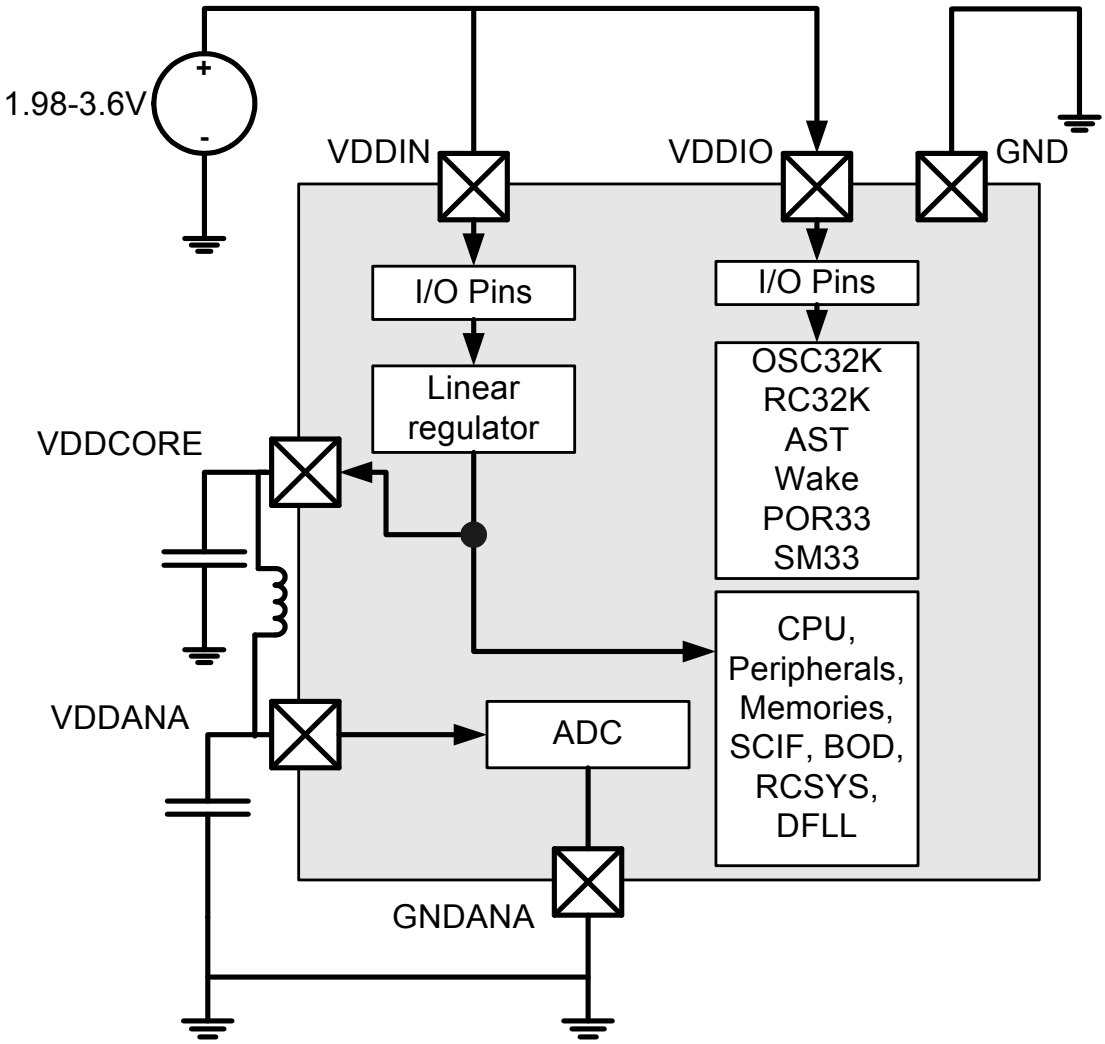
The AT32UC3L016/32/64 supports three power supply configurations:

- 3.3V single supply mode
- 1.8V single supply mode
- 3.3V supply mode, with 1.8V regulated I/O lines

### 6.1.3.1 3.3V Single Supply Mode

In 3.3V single supply mode the internal regulator is connected to the 3.3V source (VDDIN pin) and its output feeds VDDCORE. Figure 6-2 shows the power schematics to be used for 3.3V single supply mode. All I/O lines will be powered by the same power (VDDIN=VDDIO).

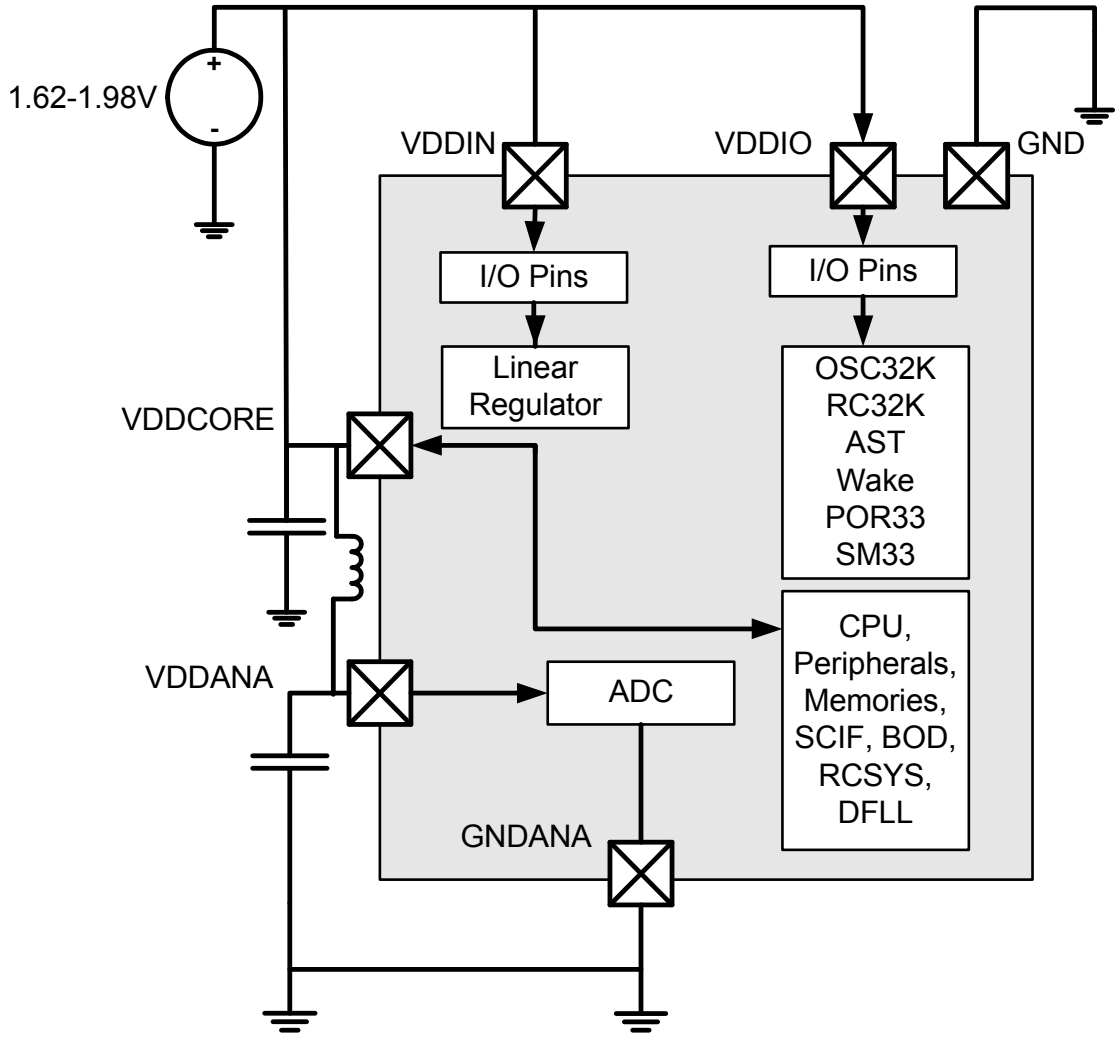
**Figure 6-2.** 3.3V Single Supply Mode



### 6.1.3.2 *1.8V Single Supply Mode*

In 1.8V single supply mode the internal regulator is not used, and VDDIO and VDDCORE are powered by a single 1.8V supply as shown in Figure 6-3. All I/O lines will be powered by the same power (VDDIN = VDDIO = VDDCORE).

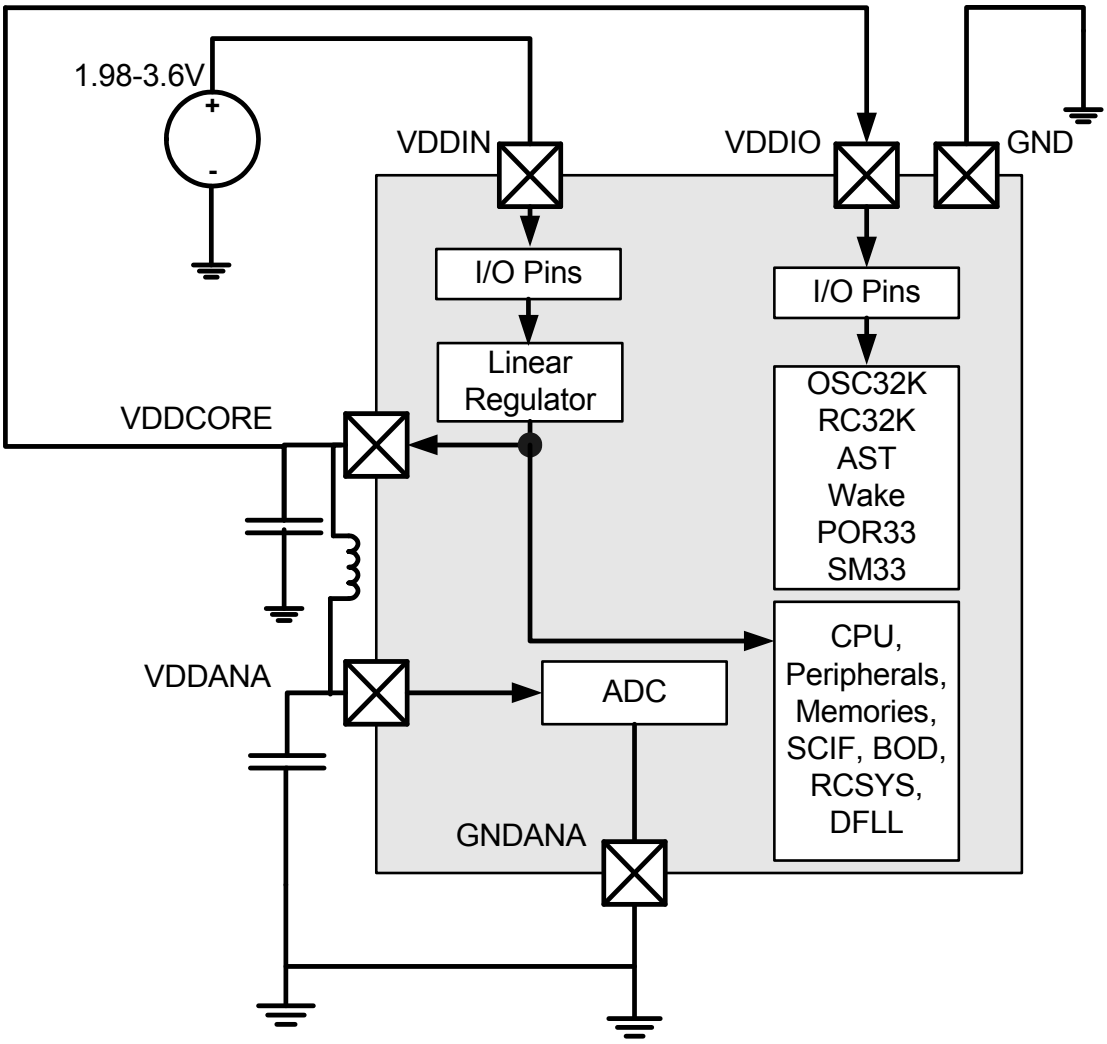
**Figure 6-3.** 1.8V Single Supply Mode.



### 6.1.3.3 *3.3V Supply Mode with 1.8V Regulated I/O Lines*

In this mode, the internal regulator is connected to the 3.3V source and its output is connected to both VDDCORE and VDDIO as shown in Figure 6-4. This configuration is required in order to use Shutdown mode.

**Figure 6-4.** 3.3V Supply Mode with 1.8V Regulated I/O Lines




In this mode, some I/O lines are powered by VDDIN while other I/O lines are powered by VDDIO. Refer to Section 3.2 on page 9 for description of power supply for each I/O line.

Refer to the Power Manager chapter for a description of what parts of the system are powered in Shutdown mode.

Important note: As the regulator has a maximum output current of 60mA, this mode can only be used in applications where the maximum I/O current is known and compatible with the core and peripheral power consumption. Typically, great care must be used to ensure that only a few I/O lines are toggling at the same time and drive very small loads.

### 6.1.4 Power-up Sequence

#### *6.1.4.1 Maximum Rise Rate*

To avoid risk of latch-up, the rise rate of the power supplies must not exceed the values described in Table 7-3 on page 42.

Recommended order for power supplies is also described in this chapter.

#### *6.1.4.2 Minimum Rise Rate*

The integrated Power-on Reset (POR33) circuitry monitoring the VDDIN powering supply requires a minimum rise rate for the VDDIN power supply.

See Table 7-3 on page 42 for the minimum rise rate value.

If the application can not ensure that the minimum rise rate condition for the VDDIN power supply is met, one of the following configurations can be used:

- A logic “0” value is applied during power-up on pin PA11 until VDDIN rises above 1.2V.
- A logic “0” value is applied during power-up on pin RESET_N until VDDIN rises above 1.2V.

## 6.2 Startup Considerations

This chapter summarizes the boot sequence of the AT32UC3L016/32/64. The behavior after power-up is controlled by the Power Manager. For specific details, refer to the Power Manager chapter.

### 6.2.1 Starting of Clocks

After power-up, the device will be held in a reset state by the Power-on Reset (POR18 and POR33) circuitry for a short time to allow the power to stabilize throughout the device. After reset, the device will use the System RC Oscillator (RCSYS) as clock source. Please refer to Table 7-17 on page 54 for the frequency for this oscillator.

On system start-up, the DFLL is disabled. All clocks to all modules are running. No clocks have a divided frequency; all parts of the system receive a clock with the same frequency as the System RC Oscillator.

When powering up the device, there may be a delay before the voltage has stabilized, depending on the rise time of the supply used. The CPU can start executing code as soon as the supply is above the POR18 and POR33 thresholds, and before the supply is stable. Before switching to a high-speed clock source, the user should use the BOD to make sure the VDDCORE is above the minimum level (1.62V).

### 6.2.2 Fetching of Initial Instructions

After reset has been released, the AVR32 UC CPU starts fetching instructions from the reset address, which is 0x80000000. This address points to the first address in the internal flash.

The code read from the internal flash is free to configure the clock system and clock sources . Please refer to the Power Manager and SCIF chapters for details.

# 7. Electrical Characteristics

## 7.1 Absolute Maximum Ratings*

**Table 7-1.** Absolute Maximum Ratings

| |
|---|
| Operating temperature ..... -40°C to +85°C |
| Storage temperature ..... -60°C to +150°C |
| Voltage on input pins (except for 5V pins) with respect to ground .....-0.3V to $V_{VDD}$[2]+0.3V |
| Voltage on 5V tolerant[1] pins with respect to ground .....-0.3V to 5.5V |
| Total DC output current on all I/O pins - VDDIO ..... 120mA |
| Total DC output current on all I/O pins - VDDIN ..... 36mA |
| Maximum operating voltage VDDCORE ..... 1.98V |
| Maximum operating voltage VDDIO, VDDIN ..... 3.6V |

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Notes: 1. 5V tolerant pins, see Section 3.2 "Peripheral Multiplexing on I/O lines" on page 9
2. $V_{VDD}$ corresponds to either $V_{VDDIN}$ or $V_{VDDIO}$, depending on the supply for the pin. Refer to Section 3.2 on page 9 for details.

## 7.2 Supply Characteristics

The following characteristics are applicable to the operating temperature range: $T_A$ = -40°C to 85°C, unless otherwise specified and are valid for a junction temperature up to $T_J$ = 100°C. Please refer to Section 6. "Supply and Startup Considerations" on page 36.

**Table 7-2.** Supply Characteristics

| Symbol | Parameter | Voltage | | |
|---|---|---|---|---|
| | | Min | Max | Unit |
| $V_{VDDIO}$ | DC supply peripheral I/Os | 1.62 | 3.6 | V |
| $V_{VDDIN}$ | DC supply peripheral I/Os, 1.8V single supply mode | 1.62 | 1.98 | V |
| | DC supply peripheral I/Os and internal regulator, 3.3V supply mode | 1.98 | 3.6 | V |
| $V_{VDDCORE}$ | DC supply core | 1.62 | 1.98 | V |
| $V_{VDDANA}$ | Analog supply voltage | 1.62 | 1.98 | V |

**Table 7-3.** Supply Rise Rates and Order[1]

| Symbol | Parameter | Rise Rate | | | |
|---|---|---|---|---|---|
| | | Min | Max | Unit | Comment |
| $V_{VDDIO}$ | DC supply peripheral I/Os | 0 | 2.5 | V/µs | |
| $V_{VDDIN}$ | DC supply peripheral I/Os and internal regulator | 0.002 | 2.5 | V/µs | Slower rise time requires external power-on reset circuit. |
| $V_{VDDCORE}$ | DC supply core | 0 | 2.5 | V/µs | Rise before or at the same time as VDDIO |
| $V_{VDDANA}$ | Analog supply voltage | 0 | 2.5 | V/µs | Rise together with VDDCORE |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

## 7.3 Maximum Clock Frequencies

These parameters are given in the following conditions:

- $V_{VDDCORE}$ = 1.62V to 1.98V
- Temperature = -40°C to 85°C

**Table 7-4.** Clock Frequencies

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $f_{CPU}$ | CPU clock frequency | | | 50 | MHz |
| $f_{PBA}$ | PBA clock frequency | | | 50 | MHz |
| $f_{PBB}$ | PBB clock frequency | | | 50 | MHz |
| $f_{GCLK0}$ | GCLK0 clock frequency | DFLLIF main reference, GCLK0 pin | | 150 | MHz |
| $f_{GCLK1}$ | GCLK1 clock frequency | DFLLIF dithering and ssg reference, GCLK1 pin | | 150 | MHz |
| $f_{GCLK2}$ | GCLK2 clock frequency | AST, GCLK2 pin | | 80 | MHz |
| $f_{GCLK3}$ | GCLK3 clock frequency | PWMA, GCLK3 pin | | 110 | MHz |
| $f_{GCLK4}$ | GCLK4 clock frequency | CAT, ACIFB, GCLK4 pin | | 110 | MHz |
| $f_{GCLK5}$ | GCLK5 clock frequency | GLOC | | 80 | MHz |

## 7.4 Power Consumption

The values in Table 7-5 are measured values of power consumption under the following conditions, except where noted:

- Operating conditions internal core supply (Figure 7-1) - this is the default configuration
  - $V_{VDDIN}$ = 3.0V
  - $V_{VDDCORE}$ = 1.62V, supplied by the internal regulator

  - Corresponds to the 3.3V supply mode with 1.8V regulated I/O lines, please refer to the Supply and Startup Considerations section for more details
    - Equivalent to the 3.3V single supply mode
    - Consumption in 1.8V single supply mode can be estimated by subtracting the regulator static current
- Operating conditions external core supply (Figure 7-2) - used only when noted
  - $V_{VDDIN} = V_{VDDCORE} = 1.8V$
  - Corresponds to the 1.8V single supply mode, please refer to the Supply and Startup Considerations section for more details
- $T_A = 25°C$
- Oscillators
  - OSC0 (crystal oscillator) stopped
  - OSC32K (32KHz crystal oscillator) running with external 32KHz crystal
  - DFLL running at 50MHz with OSC32K as reference
- Clocks
  - DFLL used as main clock source
  - CPU, HSB, and PBB clocks undivided
  - PBA clock divided by 4
  - The following peripheral clocks running
    - PM, SCIF, AST, FLASHCDW, PBA bridge
  - All other peripheral clocks stopped
- I/Os are inactive with internal pull-up
- Flash enabled in high speed mode
- POR33 disabled

**Table 7-5.** Power Consumption for Different Operating Modes

| Mode | Conditions | Measured on | Consumption Typ | Unit |
|---|---|---|---|---|
| Active[1] | -CPU running a recursive Fibonacci algorithm | Amp0 | 260 | µA/MHz |
| | -CPU running a division algorithm | | 165 | |
| Idle[1] | | | 92 | |
| Frozen[1] | | | 58 | |
| Standby[1] | | | 47 | |
| Stop | | | 37 | µA |
| DeepStop | | | 23 | |
| Static | -OSC32K and AST stopped<br>-Internal core supply | | 10 | |
| | -OSC32K running<br>-AST running at 1KHz<br>-External core supply (Figure 7-2) | | 5.3 | |
| | -OSC32K and AST stopped<br>-External core supply (Figure 7-2) | | 4.7 | |
| Shutdown | -OSC32K running<br>-AST running at 1KHz | | 600 | nA |
| | -AST and OSC32K stopped | | 9 | |

Note: 1. These numbers are valid for the measured condition only and must not be extrapolated to other frequencies.

**Figure 7-1.** Measurement Schematic, Internal Core Supply

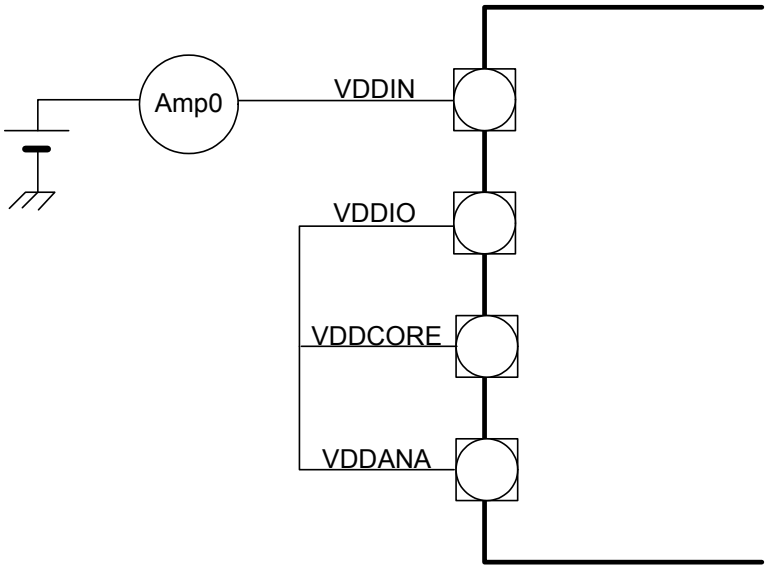
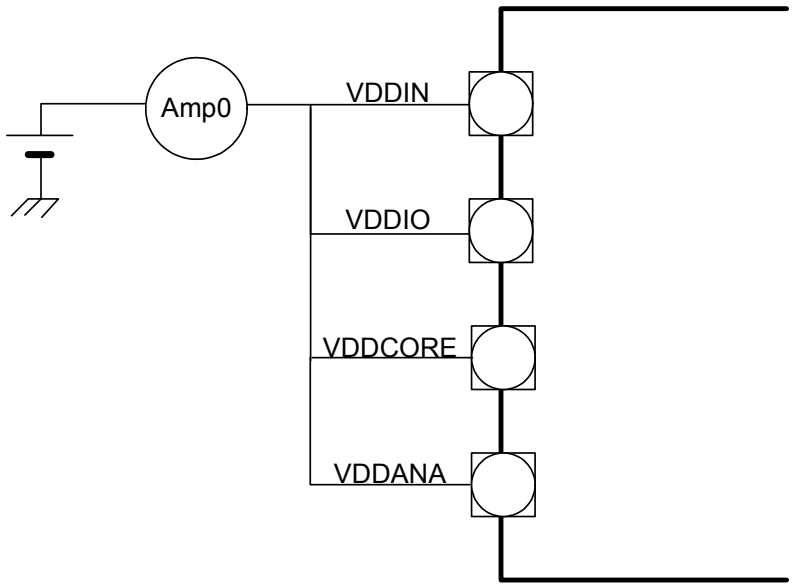


**Figure 7-2.** Measurement Schematic, External Core Supply




### 7.4.1 Peripheral Power Consumption

The values in Table 7-6 are measured values of power consumption under the following conditions.

- Operating conditions internal core supply (Figure 7-1)
  - $V_{VDDIN}$ = 3.0V
  - $V_{VDDCORE}$ = 1.62V, supplied by the internal regulator
  - Corresponds to the 3.3V supply mode with 1.8V regulated I/O lines, please refer to the Supply and Startup Considerations section for more details
- $T_A$ = 25°C
- Oscillators
  - OSC0 (crystal oscillator) stopped
  - OSC32K (32KHz crystal oscillator) running with external 32KHz crystal
  - DFLL running at 50MHz with OSC32K as reference
- Clocks
  - DFLL used as main clock source
  - CPU, HSB, and PB clocks undivided
- I/Os are inactive with internal pull-up
- Flash enabled in high speed mode
- POR33 disabled

Consumption active is the added current consumption when the module clock is turned on and the module is doing a typical set of operations.

**Table 7-6.** Typical Current Consumption by Peripheral[2]

| Peripheral | Typ Consumption Active | Unit |
|---|---|---|
| ACIFB | 14.0 | µA/MHz |
| ADCIFB[1] | 14.9 | |
| AST | 5.6 | |
| AW USART | 6.8 | |
| CAT | 12.4 | |
| EIC | 1.3 | |
| FREQM | 3.2 | |
| GLOC | 0.4 | |
| GPIO | 15.9 | |
| PWMA | 2.5 | |
| SPI | 7.6 | |
| TC | 7.2 | |
| TWIM | 5.1 | |
| TWIS | 3.2 | |
| USART | 12.3 | |
| WDT | 2.3 | |

Notes: 1. Includes the current consumption on VDDANA and ADVREFP.

2. These numbers are valid for the measured condition only and must not be extrapolated to other frequencies.

## 7.5 I/O Pin Characteristics

**Table 7-7.** Normal I/O Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $R_{PULLUP}$ | Pull-up resistance | | 75 | 100 | 145 | kOhm |
| $V_{IL}$ | Input low-level voltage | $V_{VDD}$ = 3.0V | -0.3 | | 0.3*$V_{VDD}$ | V |
| | | $V_{VDD}$ = 1.62V | -0.3 | | 0.3*$V_{VDD}$ | |
| $V_{IH}$ | Input high-level voltage | $V_{VDD}$ = 3.6V | 0.7*$V_{VDD}$ | | $V_{VDD}$ + 0.3 | V |
| | | $V_{VDD}$ = 1.98V | 0.7*$V_{VDD}$ | | $V_{VDD}$ + 0.3 | |
| $V_{OL}$ | Output low-level voltage | $V_{VDD}$ = 3.0V, $I_{OL}$ = 3mA | | | 0.4 | V |
| | | $V_{VDD}$ = 1.62V, $I_{OL}$ = 2mA | | | 0.4 | |
| $V_{OH}$ | Output high-level voltage | $V_{VDD}$ = 3.0V, $I_{OH}$ = 3mA | $V_{VDD}$ - 0.4 | | | V |
| | | $V_{VDD}$ = 1.62V, $I_{OH}$ = 2mA | $V_{VDD}$ - 0.4 | | | |
| $f_{MAX}$ | Output frequency[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 45 | MHz |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 23 | |
| $t_{RISE}$ | Rise time[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 4.7 | ns |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 11.5 | |
| $t_{FALL}$ | Fall time[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 4.8 | |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 12 | |
| $I_{LEAK}$ | Input leakage current | Pull-up resistors disabled | | | 1 | µA |
| $C_{IN}$ | Input capacitance, all normal I/O pins except PA05, PA07, PA17, PA20, PA21, PB04, PB05 | TQFP48 package | | 1.4 | | pF |
| | | QFN48 package | | 1.1 | | |
| $C_{IN}$ | Input capacitance, PA20 | TQFP48 package | | 2.7 | | |
| | | QFN48 package | | 2.4 | | |
| | | TLLGA 48 package | | 2.4 | | |
| $C_{IN}$ | Input capacitance, PA05, PA07, PA17, PA21, PB04, PB05 | TQFP48 package | | 3.8 | | |
| | | QFN48 package | | 3.5 | | |
| | | TLLGA 48 package | | 3.5 | | |

Notes: 1. $V_{VDD}$ corresponds to either $V_{VDDIN}$ or $V_{VDDIO}$, depending on the supply for the pin. Refer to Section 3.2 on page 9 for details.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Table 7-8.** High-drive I/O Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $R_{PULLUP}$ | Pull-up resistance | PA06 | 30 | 50 | 110 | kOhm |
| | | PA02, PB01, RESET | 75 | 100 | 145 | |
| | | PA08, PA09 | 10 | 20 | 45 | |

**Table 7-8.** High-drive I/O Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input low-level voltage | $V_{VDD}$ = 3.0V | -0.3 | | 0.3*$V_{VDD}$ | V |
| | | $V_{VDD}$ = 1.62V | -0.3 | | 0.3*$V_{VDD}$ | |
| $V_{IH}$ | Input high-level voltage | $V_{VDD}$ = 3.6V | 0.7*$V_{VDD}$ | | $V_{VDD}$ + 0.3 | V |
| | | $V_{VDD}$ = 1.98V | 0.7*$V_{VDD}$ | | $V_{VDD}$ + 0.3 | |
| $V_{OL}$ | Output low-level voltage | $V_{VDD}$ = 3.0V, $I_{OL}$ = 6mA | | | 0.4 | V |
| | | $V_{VDD}$ = 1.62V, $I_{OL}$ = 4mA | | | 0.4 | |
| $V_{OH}$ | Output high-level voltage | $V_{VDD}$ = 3.0V, $I_{OH}$ = 6mA | $V_{VDD}$-0.4 | | | V |
| | | $V_{VDD}$ = 1.62V, $I_{OH}$ = 4mA | $V_{VDD}$-0.4 | | | |
| $f_{MAX}$ | Output frequency, all High-drive I/O pins, except PA08 and PA09[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 45 | MHz |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 23 | |
| $t_{RISE}$ | Rise time, all High-drive I/O pins, except PA08 and PA09[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 4.7 | ns |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 11.5 | |
| $t_{FALL}$ | Fall time, all High-drive I/O pins, except PA08 and PA09[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 4.8 | |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 12 | |
| $f_{MAX}$ | Output frequency, PA08 and PA09[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 52 | MHz |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 39 | |
| $t_{RISE}$ | Rise time, PA08 and PA09[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 2.9 | ns |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 4.9 | |
| $t_{FALL}$ | Fall time, PA08 and PA09[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 2.5 | |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 4.6 | |
| $I_{LEAK}$ | Input leakage current | Pull-up resistors disabled | | | 1 | µA |
| $C_{IN}$ | Input capacitance, all High-drive I/O pins, except PA08 and PA09 | TQFP48 package | | 2,2 | | pF |
| | | QFN48 package | | 2.0 | | |
| | | TLLGA 48 package | | 2.0 | | |
| $C_{IN}$ | Input capacitance, PA08 and PA09 | TQFP48 package | | 7.0 | | |
| | | QFN48 package | | 6.7 | | |
| | | TLLGA 48 package | | 6.7 | | |

Notes: 1. $V_{VDD}$ corresponds to either $V_{VDDIN}$ or $V_{VDDIO}$, depending on the supply for the pin. Refer to Section 3.2 on page 9 for details.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Table 7-9.** High-drive I/O, 5V Tolerant, Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $R_{PULLUP}$ | Pull-up resistance | | 30 | 50 | 110 | kOhm |
| $V_{IL}$ | Input low-level voltage | $V_{VDD}$ = 3.0V | -0.3 | | 0.3*$V_{VDD}$ | V |
| | | $V_{VDD}$ = 1.62V | -0.3 | | 0.3*$V_{VDD}$ | |

**Table 7-9.** High-drive I/O, 5V Tolerant, Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{IH}$ | Input high-level voltage | $V_{VDD}$ = 3.6V | 0.7*$V_{VDD}$ | | 5.5 | V |
| | | $V_{VDD}$ = 1.98V | 0.7*$V_{VDD}$ | | 5.5 | |
| $V_{OL}$ | Output low-level voltage | $V_{VDD}$ = 3.0V, $I_{OL}$ = 6mA | | | 0.4 | V |
| | | $V_{VDD}$ = 1.62V, $I_{OL}$ = 4mA | | | 0.4 | |
| $V_{OH}$ | Output high-level voltage | $V_{VDD}$ = 3.0V, $I_{OH}$ = 6mA | $V_{VDD}$-0.4 | | | V |
| | | $V_{VDD}$ = 1.62V, $I_{OH}$ = 4mA | $V_{VDD}$-0.4 | | | |
| $f_{MAX}$ | Output frequency[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 87 | MHz |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 58 | |
| $t_{RISE}$ | Rise time[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 2.3 | ns |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 4.3 | |
| $t_{FALL}$ | Fall time[2] | $V_{VDD}$ = 3.0V, load = 10pF | | | 1.9 | |
| | | $V_{VDD}$ = 3.0V, load = 30pF | | | 3.7 | |
| $I_{LEAK}$ | Input leakage current | 5.5V, pull-up resistors disabled | | | 10 | µA |
| $C_{IN}$ | Input capacitance | TQFP48 package | | 4.5 | | pF |
| | | QFN48 package | | 4.2 | | |
| | | TLLGA48 package | | 4.2 | | |

Notes: 1. $V_{VDD}$ corresponds to either $V_{VDDIN}$ or $V_{VDDIO}$, depending on the supply for the pin. Refer to Section 3.2 on page 9 for details.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Table 7-10.** TWI Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $R_{PULLUP}$ | Pull-up resistance | | 25 | 35 | 60 | kOhm |
| $V_{IL}$ | Input low-level voltage | $V_{VDD}$ = 3.0V | -0.3 | | 0.3*$V_{VDD}$ | V |
| | | $V_{VDD}$ = 1.62V | -0.3 | | 0.3*$V_{VDD}$ | |
| $V_{IH}$ | Input high-level voltage | $V_{VDD}$ = 3.6V | 0.7*$V_{VDD}$ | | $V_{VDD}$ + 0.3 | V |
| | | $V_{VDD}$ = 1.98V | 0.7*$V_{VDD}$ | | $V_{VDD}$ + 0.3 | |
| | Input high-level voltage, 5V tolerant SMBUS compliant pins | $V_{VDD}$ = 3.6V | 0.7*$V_{VDD}$ | | 5.5 | V |
| | | $V_{VDD}$ = 1.98V | 0.7*$V_{VDD}$ | | 5.5 | |
| $V_{OL}$ | Output low-level voltage | $I_{OL}$ = 3mA | | | 0.4 | V |
| $I_{LEAK}$ | Input leakage current | Pull-up resistors disabled | | | 1 | µA |
| $I_{IL}$ | Input low leakage | | | | 1 | |
| $I_{IH}$ | Input high leakage | | | | 1 | |
| $C_{IN}$ | Input capacitance | TQFP48 package | | 3.8 | | pF |
| | | QFN48 package | | 3.5 | | |
| | | TLLGA48 package | | 3.5 | | |

**Table 7-10.** TWI Pin Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $t_{FALL}$ | Fall time | Cbus = 400pF, $V_{VDD}$ > 2.0V | | 250 | | ns |
| | | Cbus = 400pF, $V_{VDD}$ > 1.62V | | 470 | | |
| $f_{MAX}$ | Max frequency | Cbus = 400pF, $V_{VDD}$ > 2.0V | 400 | | | kHz |

Note: 1. $V_{VDD}$ corresponds to either $V_{VDDIN}$ or $V_{VDDIO}$, depending on the supply for the pin. Refer to Section 3.2 on page 9 for details.

## 7.6 Oscillator Characteristics

### 7.6.1 Oscillator 0 (OSC0) Characteristics

#### *7.6.1.1 Digital Clock Characteristics*

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN.

**Table 7-11.** Digital Clock Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $f_{CPXIN}$ | XIN clock frequency | | | | 50 | MHz |
| $t_{CPXIN}$ | XIN clock duty cycle | | 40 | | 60 | % |
| $t_{STARTUP}$ | Startup time | | | 0 | | cycles |
| $C_{IN}$ | XIN input capacitance | TQFP48 package | | 7.0 | | pF |
| | | QFN48 package | | 6.7 | | |
| | | TLLGA48 package | | 6.7 | | |

#### *7.6.1.2 Crystal Oscillator Characteristics*

The following table describes the characteristics for the oscillator when a crystal is connected between XIN and XOUT as shown in Figure 7-3. The user must choose a crystal oscillator where the crystal load capacitance $C_L$ is within the range given in the table. The exact value of $C_L$ can be found in the crystal datasheet. The capacitance of the external capacitors ($C_{LEXT}$) can then be computed as follows:

$$C_{LEXT} = 2(C_L - C_i) - C_{PCB}$$

where $C_{PCB}$ is the capacitance of the PCB and $C_i$ is the internal equivalent load capacitance.

**Table 7-12.** Crystal Oscillator Characteristics

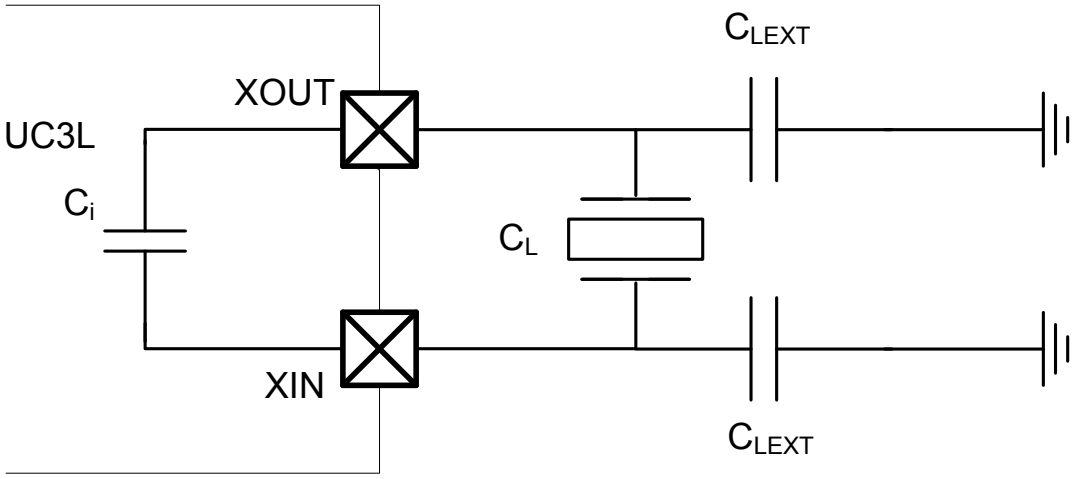
| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Crystal oscillator frequency | | 0.45 | 10 | 16 | MHz |
| $C_L$ | Crystal load capacitance | | 6 | | 18 | pF |
| $C_i$ | Internal equivalent load capacitance | | | 2 | | |

**Table 7-12.** Crystal Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $t_{STARTUP}$ | Startup time | SCIF.OSCCTRL.GAIN = 2[1] | | 30 000[2] | | cycles |
| $I_{OSC}$ | Current consumption | Active mode, f = 0.45MHz, SCIF.OSCCTRL.GAIN = 0 | | 30 | | µA |
| | | Active mode, f = 10MHz, SCIF.OSCCTRL.GAIN = 2 | | 170 | | |

Notes: 1. Please refer to the SCIF chapter for details.
2. Nominal crystal cycles.
3. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Figure 7-3.** Oscillator Connection




### 7.6.2 32KHz Crystal Oscillator (OSC32K) Characteristics

Figure 7-3 and the equation above also applies to the 32KHz oscillator connection. The user must choose a crystal oscillator where the crystal load capacitance $C_L$ is within the range given in the table. The exact value of $C_L$ can then be found in the crystal datasheet.

**Table 7-13.** 32 KHz Crystal Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Crystal oscillator frequency | | | 32 768 | | Hz |
| $t_{STARTUP}$ | Startup time | $R_S$ = 60kOhm, $C_L$ = 9pF | | 30 000[1] | | cycles |
| $C_L$ | Crystal load capacitance[2] | | 6 | | 12.5 | pF |
| $C_i$ | Internal equivalent load capacitance | | | 2 | | |
| $I_{OSC32}$ | Current consumption | | | 0.9 | | µA |
| $R_S$ | Equivalent series resistance[2] | 32 768Hz | 35 | | 85 | kOhm |

Notes: 1. Nominal crystal cycles.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

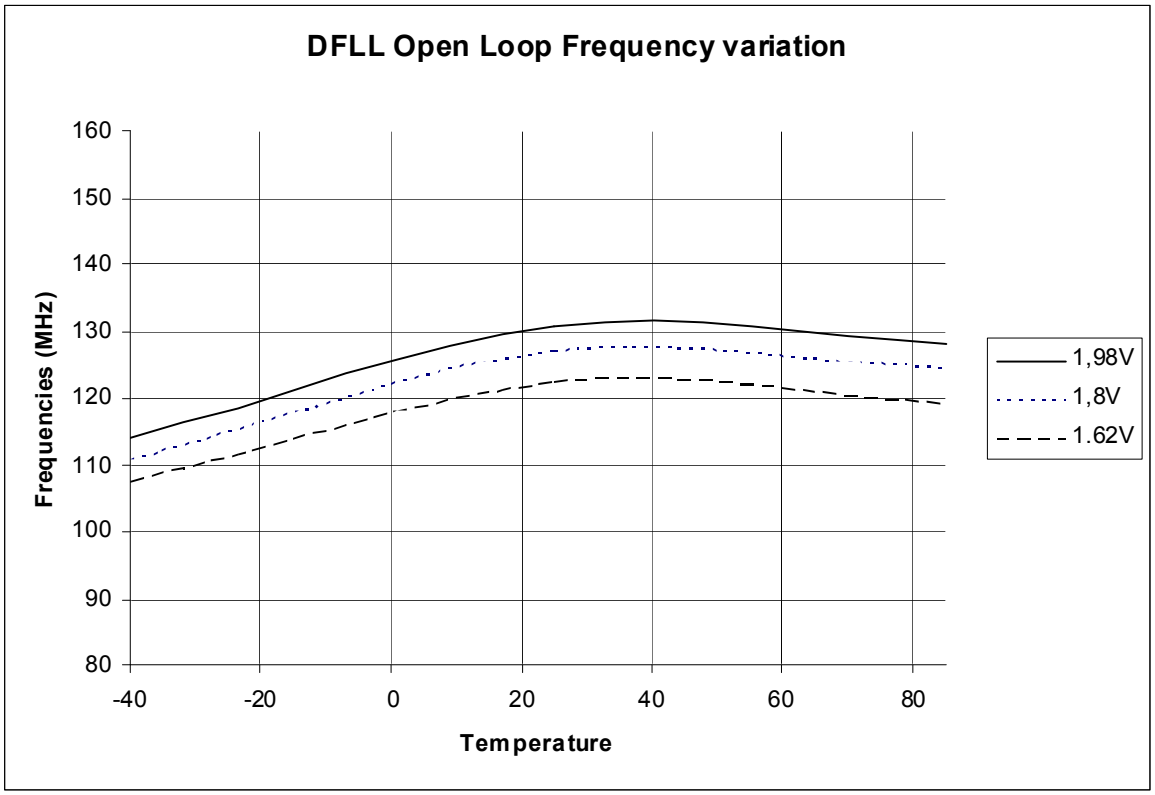
### 7.6.3 Digital Frequency Locked Loop (DFLL) Characteristics

**Table 7-14.** Digital Frequency Locked Loop Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Output frequency(2) | | 40 | | 150 | MHz |
| $f_{REF}$ | Reference frequency(2) | | 8 | | 150 | kHz |
| | FINE resolution | FINE > 100, all COARSE values | | 0.25 | | % |
| | Frequency drift over voltage and temperature | | | See Figure 7-4 | | |
| | Accuracy(2) | Fine lock, $f_{REF}$ = 32kHz, SSG disabled | | 0.1 | 0.5 | % |
| | | Accurate lock, $f_{REF}$ = 32kHz, dither clk RCSYS/2, SSG disabled | | 0.06 | 0.5 | |
| | | Fine lock, $f_{REF}$ = 8-150kHz, SSG disabled | | 0.2 | 1 | |
| | | Accurate lock, $f_{REF}$ = 8-150kHz, dither clk RCSYS/2, SSG disabled | | 0.1 | 1 | |
| $I_{DFLL}$ | Power consumption | | | 22 | | µA/MHz |
| $t_{STARTUP}$ | Startup time(2) | Within 90% of final values | | | 100 | µs |
| $t_{LOCK}$ | Lock time | $f_{REF}$ = 32kHz, fine lock, SSG disabled | | 600 | | |
| | | $f_{REF}$ = 32kHz, accurate lock, dithering clock = RCSYS/2, SSG disabled | | 1100 | | |

Notes: 1. Spread Spectrum Generator (SSG) is disabled by writing a zero to the EN bit in the SCIF.DFLL0SSG register.
2. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Figure 7-4.** DFLL Open Loop Frequency Variation[1]




Note: 1. The plot shows a typical behaviour for coarse = 99 and fine = 255 in open loop mode.

### 7.6.4 120MHz RC Oscillator (RC120M) Characteristics

**Table 7-15.** Internal 120MHz RC Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Output frequency[1] | | 88 | 120 | 152 | MHz |
| $I_{RC120M}$ | Current consumption | | | 1.85 | | mA |
| $t_{STARTUP}$ | Startup time | $V_{VDDCORE}$ = 1.8V | | 3 | | µs |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

### 7.6.5 32kHz RC Oscillator (RC32K) Characteristics

**Table 7-16.** 32kHz RC Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Output frequency[1] | | 20 | 32 | 44 | kHz |
| $I_{RC32K}$ | Current consumption | | | 0.6 | | µA |
| $t_{STARTUP}$ | Startup time | | | 100 | | µs |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

### 7.6.6 System RC Oscillator (RCSYS) Characteristics

**Table 7-17.** System RC Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OUT}$ | Output frequency | Calibrated at 85°C | 111.6 | 115 | 118.4 | kHz |

## 7.7 Flash Characteristics

Table 7-18 gives the device maximum operating frequency depending on the number of flash wait states and the flash read mode. The FSW bit in the FLASHCDW FSR register controls the number of wait states used when accessing the flash memory.

**Table 7-18.** Maximum Operating Frequency

| Flash Wait States | Read Mode | Maximum Operating Frequency |
|---|---|---|
| 1 | High speed read mode | 50MHz |
| 0 | | 25MHz |
| 1 | Normal read mode | 30MHz |
| 0 | | 15MHz |

**Table 7-19.** Flash Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $t_{FPP}$ | Page programming time | $f_{CLK\_HSB}$ = 50MHz | | 5 | | ms |
| $t_{FPE}$ | Page erase time | | | 5 | | |
| $t_{FFP}$ | Fuse programming time | | | 1 | | |
| $t_{FEA}$ | Full chip erase time (EA) | | | 5 | | |
| $t_{FCE}$ | JTAG chip erase time (CHIP_ERASE) | $f_{CLK\_HSB}$ = 115kHz | | 170 | | |

**Table 7-20.** Flash Endurance and Data Retention

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $N_{FARRAY}$ | Array endurance (write/page) | | 100k | | | cycles |
| $N_{FFUSE}$ | General Purpose fuses endurance (write/bit) | | 10k | | | |
| $t_{RET}$ | Data retention | | 15 | | | years |

## 7.8 Analog Characteristics

### 7.8.1 Voltage Regulator Characteristics

**Table 7-21.** VREG Electrical Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{VDDIN}$ | Input voltage range | | 1.98 | 3.3 | 3.6 | V |
| $V_{VDDCORE}$ | Output voltage, calibrated value | $V_{VDDIN}$ >= 1.98V | | 1.8 | | |
| | Output voltage accuracy | $I_{OUT}$ = 0.1mA to 60mA, $V_{VDDIN}$ > 2.2V | | 2 | | % |
| | | $I_{OUT}$ = 0.1mA to 60mA, $V_{VDDIN}$ = 1.98V to 2.2V | | 4 | | |
| $I_{OUT}$ | DC output current[1] | Normal mode | | | 60 | mA |
| | | Low power mode | | | 1 | |
| $I_{VREG}$ | Static current of internal regulator | Normal mode | | 20 | | µA |
| | | Low power mode | | 6 | | |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Table 7-22.** Decoupling Requirements

| Symbol | Parameter | Condition | Typ | Techno. | Units |
|---|---|---|---|---|---|
| $C_{IN1}$ | Input regulator capacitor 1 | | 33 | | nF |
| $C_{IN2}$ | Input regulator capacitor 2 | | 100 | | |
| $C_{IN3}$ | Input regulator capacitor 3 | | 10 | | µF |
| $C_{OUT1}$ | Output regulator capacitor 1 | | 100 | | nF |
| $C_{OUT2}$ | Output regulator capacitor 2 | | 2.2 | Tantalum 0.5<ESR<10Ohm | µF |

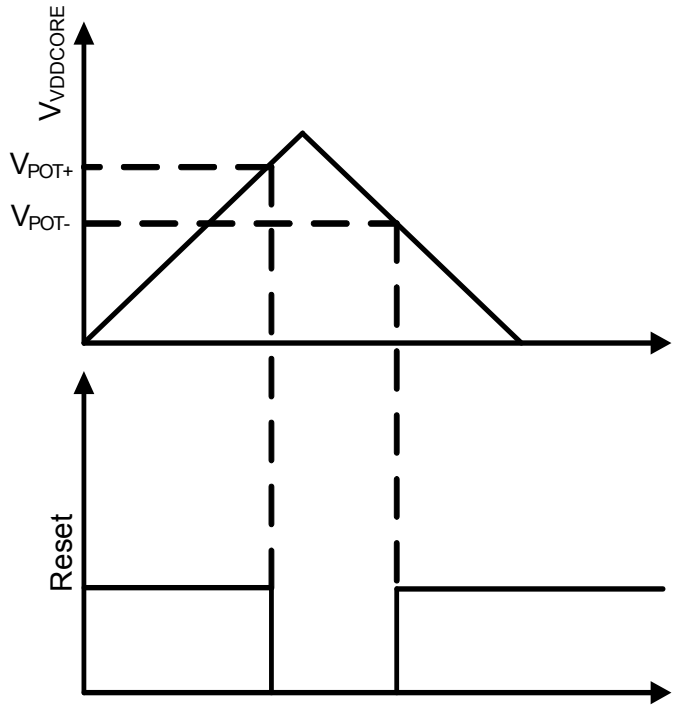
Note: 1. Refer to Section 6.1.2 on page 36.

### 7.8.2 Power-on Reset 18 Characteristics

**Table 7-23.** POR18 Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{POT+}$ | Voltage threshold on $V_{VDDCORE}$ rising | | | 1.45 | 1.58 | V |
| $V_{POT-}$ | Voltage threshold on $V_{VDDCORE}$ falling | | 1.2 | 1.32 | | |
| $t_{DET}$ | Detection time | Time with VDDCORE < $V_{POT-}$ necessary to generate a reset signal | | 460 | | µs |

**Figure 7-5.** POR18 Operating Principles

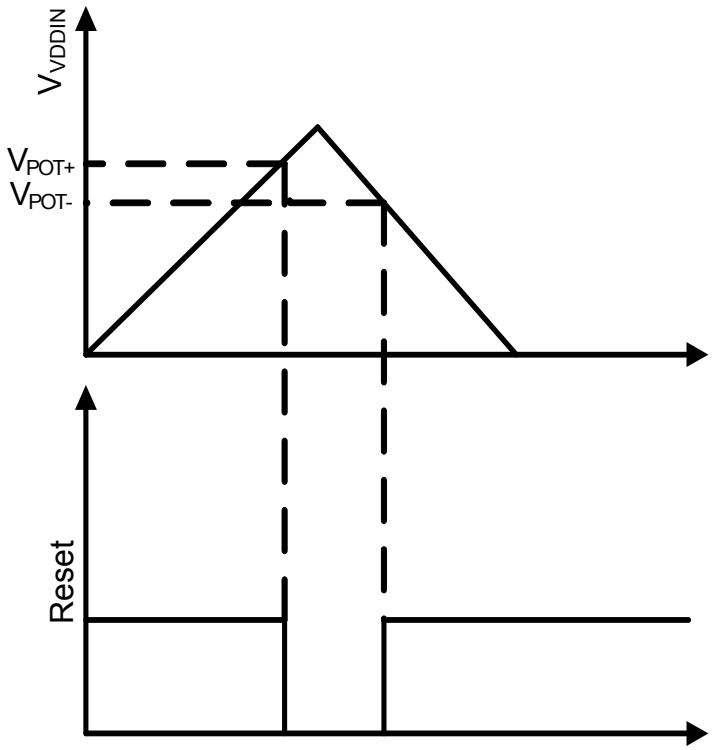


### 7.8.3 Power-on Reset 33 Characteristics

**Table 7-24.** POR33 Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{POT+}$ | Voltage threshold on $V_{VDDIN}$ rising | | | 1.49 | 1.58 | V |
| $V_{POT-}$ | Voltage threshold on $V_{VDDIN}$ falling | | 1.3 | 1.45 | | |
| $t_{DET}$ | Detection time | Time with VDDIN < $V_{POT-}$ necessary to generate a reset signal | | 460 | | µs |
| $I_{POR33}$ | Current consumption | After $t_{RESET}$ | | 15 | | µA |
| $t_{STARTUP}$ | Startup time | | | 400 | | µs |

**Figure 7-6.** POR33 Operating Principles




### 7.8.4 Brown Out Detector Characteristics

The values in Table 7-25 describe the values of the BODLEVEL in the flash General Purpose Fuse register.

**Table 7-25.** BODLEVEL Values

| BODLEVEL Value | Min | Typ | Max | Units |
|---|---|---|---|---|
| 011111 binary (31) 0x1F | | 1.56 | | V |
| 100111 binary (39) 0x27 | | 1.65 | | |

**Table 7-26.** BOD Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{HYST}$ | BOD hysteresis | T = 25°C | | 10 | | mV |
| $t_{DET}$ | Detection time | Time with VDDCORE < BODLEVEL necessary to generate a reset signal | | 1 | | µs |
| $I_{BOD}$ | Current consumption | | | 16 | | µA |
| $t_{STARTUP}$ | Startup time | | | 5 | | µs |

### 7.8.5 Supply Monitor 33 Characteristics

**Table 7-27.** SM33 Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{TH}$ | Voltage threshold | Calibrated[1], T = 25°C | 1.675 | 1.75 | 1.825 | V |
| | Step size, between adjacent values in SCIF.SM33.CALIB | | | 11 | | mV |
| $V_{HYST}$ | Hysteresis | | | 30 | | |
| $t_{DET}$ | Detection time | Time with VDDIN < $V_{TH}$ necessary to generate a reset signal | | 280 | | µs |
| $I_{SM33}$ | Current consumption | Normal mode | | 15 | | µA |
| $t_{STARTUP}$ | Startup time | Normal mode | | 140 | | µs |

Note: 1. Calibration value can be read from the SCIF.SM33.CALIB field. This field is updated by the flash fuses after a reset. Refer to SCIF chapter for details.

### 7.8.6 Analog to Digital Converter Characteristics

**Table 7-28.** ADC Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $f_{ADC}$ | ADC clock frequency | 10-bit resolution mode | | | 6 | MHz |
| | | 8-bit resolution mode | | | 6 | |
| $t_{STARTUP}$ | Startup time | Return from Idle Mode | | 15 | | µs |
| $t_{CONV}$ | Conversion time (latency) | $f_{ADC}$ = 6MHz | 11 | | 26 | cycles |
| | Throughput rate | $V_{VDD}$ > 3.0V, $f_{ADC}$ = 6MHz, 10-bit resolution mode, low impedance source | | | 460 | kSPS |
| | | $V_{VDD}$ > 3.0V, $f_{ADC}$ = 6MHz, 8-bit resolution mode, low impedance source | | | 460 | |
| $V_{ADVREFP}$ | Reference voltage range | $V_{ADVREFP}$ = $V_{VDDANA}$ | 1.62 | | 1.98 | V |
| $I_{ADC}$ | Current consumption on $V_{VDDANA}$ | ADC Clock = 6MHz | | 300 | | µA |
| $I_{ADVREFP}$ | Current consumption on ADVREFP pin | $f_{ADC}$ = 6MHz | | 250 | | |

Note: These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

#### *7.8.6.1 Inputs and Sample and Hold Aquisition Time*

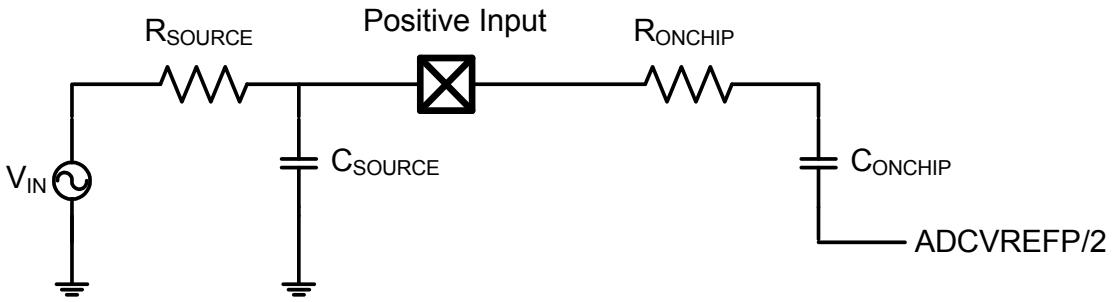
**Table 7-29.** Analog Inputs

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{ADn}$ | Input Voltage Range | 10-bit mode | 0 | | $V_{ADVREFP}$ | V |
| | | 8-bit mode | | | | |
| $C_{ONCHIP}$ | Internal Capacitance[1] | | | | 21.5 | pF |
| $R_{ONCHIP}$ | Internal Resistance[1] | $V_{VDDIO}$ = 3.0V to 3.6V, $V_{VDDCORE}$ = 1.8V | | | 2.55 | kOhm |
| | | $V_{VDDIO}$ = $V_{VDDCORE}$ = 1.62V to 1.98V | | | 55.3 | |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

An analog voltage input must be able to charge the sample and hold (S/H) capacitor in the ADC in order to achieve maximum accuracy. Seen externally the ADC input consists of a resistor ($R_{ONCHIP}$) and a capacitor ($C_{ONCHIP}$). In addition the resistance ($R_{SOURCE}$) and capacitance ($C_{SOURCE}$) of the PCB and source must be taken into account when calculating the sample and hold time. Figure 7-7 shows the ADC input channel equivalent circuit.

**Figure 7-7.** ADC Input




The minimum sample and hold time (in ns) can be found using this formula:

$$t_{SAMPLEHOLD} \geq (R_{ONCHIP} + R_{OFFCHIP}) \times (C_{ONCHIP} + C_{OFFCHIP}) \times \ln(2^{n+1})$$

Where n is the number of bits in the conversion. $t_{SAMPLEHOLD}$ is defined by the SHTIM field in the ADCIFB ACR register. Please refer to the ADCIFB chapter for more information.

### *7.8.6.2 Applicable Conditions and Derating Data*

**Table 7-30.** Transfer Characteristics 10-bit Resolution Mode

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Resolution | | | 10 | | Bit |
| Integral non-linearity | ADC clock frequency = 6MHz | | +/-2 | | LSB |
| Differential non-linearity | | -0.9 | | 1 | |
| Offset error | | | +/-4 | | |
| Gain error | | | +/-4 | | |

**Table 7-31.** Transfer Characteristics 8-bit Resolution Mode

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Resolution | | | 8 | | Bit |
| Integral non-linearity | ADC clock frequency = 6MHz | | +/-0.5 | | LSB |
| Differential non-linearity | | -0.23 | | 0.25 | |
| Offset error | | | +/-1 | | |
| Gain error | | | +/-1 | | |

### 7.8.7 Temperature Sensor Characteristics

**Table 7-32.** Temperature Sensor Characteristics[1]

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Gradient | | | 1 | | mV/°C |
| $I_{TS}$ | Current consumption | | | 0.5 | | µA |
| $t_{STARTUP}$ | Startup time | | | 0 | | µs |

Note: 1. The Temperature Sensor is not calibrated. The accuracy of the Temperature Sensor is governed by the ADC accuracy.

### 7.8.8 Analog Comparator Characteristics

**Table 7-33.** Analog Comparator Characteristics

| Symbol | Parameter | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | Positive input voltage range | | -0.2 | | $V_{VDDIO}$ + 0.3 | V |
| | Negative input voltage range | | -0.2 | | $V_{VDDIO}$ - 0.6 | |
| | Statistical offset | $V_{ACREFN}$ = 1.0V, $f_{AC}$ = 12MHz, filter length = 2, hysteresis = 0[1] | | 20 | | mV |
| $f_{AC}$ | Clock frequency for GCLK4 | | | | 12 | MHz |
| | Throughput rate[3] | $f_{AC}$ = 12MHz | | | 12 000 000 | Comparisons per second |
| | Propagation delay | Delay from input change to Interrupt Status Register Changes | | $\left(\left\lfloor \frac{1}{t_{CLKACIFB} \times f_{AC}} \right\rfloor + 3\right) \times t_{CLKACIFB}$ | | ns |
| $I_{AC}$ | Current consumption | All channels, VDDIO = 3.3V, $f_A$ = 3MHz | | 420 | | µA |
| $t_{STARTUP}$ | Startup time | | | 3 | | cycles |
| | Input current per pin | | | 0.2 | | µA/MHz[2] |

Notes: 1. AC.CONFn.FLEN and AC.CONFn.HYS fields, refer to the Analog Comparator Interface chapter.
2. Referring to $f_{AC}$.
3. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

### 7.8.9 Capacitive Touch Characteristics

#### *7.8.9.1 Discharge Current Source*

**Table 7-34.** DICS Characteristics

| Symbol | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| $R_{REF}$ | Internal resistor | | 120 | | kOhm |
| k | Trim step size | | 0.7 | | % |

#### *7.8.9.2 Strong Pull-up Pull-down*

**Table 7-35.** Strong Pull-up Pull-down

| Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|
| Pull-down resistor | | 1 | | kOhm |
| Pull-up resistor | | 1 | | |

## 7.9 Timing Characteristics

### 7.9.1 Startup, Reset, and Wake-up Timing

The startup, reset, and wake-up timings are calculated using the following formula:

$$t = t_{CONST} + N_{CPU} \times t_{CPU}$$

Where $t_{CONST}$ and $N_{CPU}$ are found in Table 7-36. $t_{CPU}$ is the period of the CPU clock. If another clock source than RCSYS is selected as CPU clock the startup time of the oscillator, $t_{OSCSTART}$, must added to the wake-up time in the stop, deepstop, and static sleep modes. Please refer to the source for the CPU clock in the "Oscillator Characteristics" on page 50 for more details about oscillator startup times.

**Table 7-36.** Maximum Reset and Wake-up Timing[1]

| Parameter | | Measuring | Max $t_{CONST}$ (in µs) | Max $N_{CPU}$ |
|---|---|---|---|---|
| Startup time from power-up, using regulator | | Time from VDDIN crossing the $V_{POT+}$ threshold of POR33 to the first instruction entering the decode stage of CPU. VDDCORE is supplied by the internal regulator. | 2210 | 0 |
| Startup time from power-up, no regulator | | Time from VDDIN crossing the $V_{POT+}$ threshold of POR33 to the first instruction entering the decode stage of CPU. VDDCORE is connected to VDDIN. | 1810 | 0 |
| Startup time from reset release | | Time from releasing a reset source (except POR18, POR33, and SM33) to the first instruction entering the decode stage of CPU. | 170 | 0 |
| Wake-up | Idle | From wake-up event to the first instruction of an interrupt routine entering the decode stage of the CPU. | 0 | 19 |
| | Frozen | | 0 | 110 |
| | Standby | | 0 | 110 |
| | Stop | | 27 + $t_{OSCSTART}$ | 116 |
| | Deepstop | | 27 + $t_{OSCSTART}$ | 116 |
| | Static | | 97 + $t_{OSCSTART}$ | 116 |
| Wake-up from shutdown | | From wake-up event to the first instruction entering the decode stage of the CPU. | 1180 | 0 |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

### 7.9.2 RESET_N Timing

**Table 7-37.** RESET_N Waveform Parameters[1]

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| $t_{RESET}$ | RESET_N minimum pulse length | | 10 | | ns |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

### 7.9.3 USART in SPI Mode Timing

#### *7.9.3.1 Master mode*

**Figure 7-8.** USART in SPI Master Mode With (CPOL= CPHA= 0) or (CPOL= CPHA= 1)

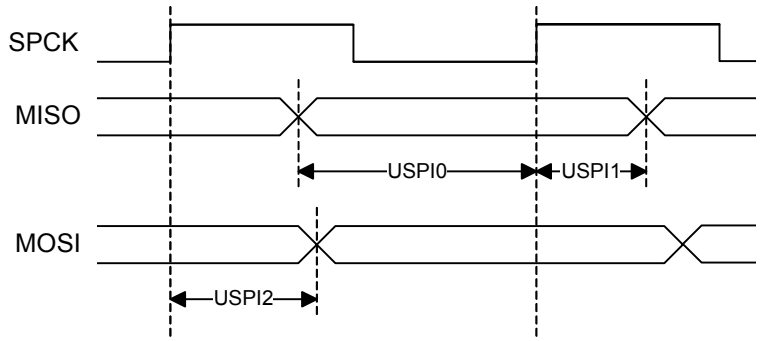



**Figure 7-9.** USART in SPI Master Mode With (CPOL= 0 and CPHA= 1) or (CPOL= 1 and CPHA= 0)

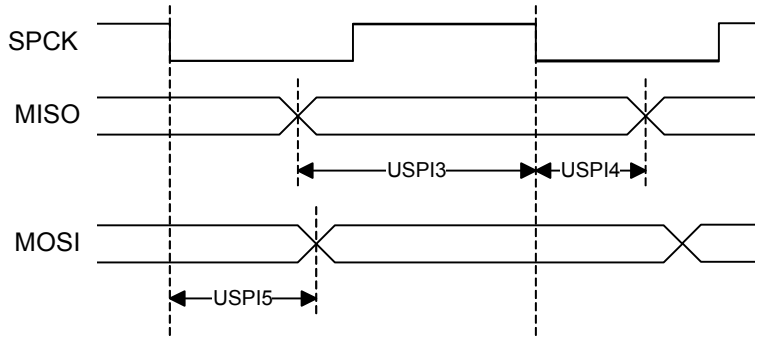



**Table 7-38.** USART in SPI Mode Timing, Master Mode[1]

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| USPI0 | MISO setup time before SPCK rises | $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 40pF | 30.0+ $t_{SAMPLE}$[2] | | ns |
| USPI1 | MISO hold time after SPCK rises | | 0 | | |
| USPI2 | SPCK rising to MOSI delay | | | 8.5 | |
| USPI3 | MISO setup time before SPCK falls | | 25.5 + $t_{SAMPLE}$[2] | | |
| USPI4 | MISO hold time after SPCK falls | | 0 | | |
| USPI5 | SPCK falling to MOSI delay | | | 13.6 | |

Notes: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

2. Where: $t_{SAMPLE} = t_{SPCK} - \left( \left\lfloor \frac{t_{SPCK}}{2 \times t_{CLKUSART}} \right\rfloor \frac{1}{2} \right) \times t_{CLKUSART}$

**Maximum SPI Frequency, Master Output**

The maximum SPI master output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn}, \frac{f_{CLKSPI} \times 2}{9})$$

Where $SPIn$ is the MOSI delay, USPI2 or USPI5 depending on CPOL and NCPHA. $f_{PINMAX}$ is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins. $f_{CLKSPI}$ is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

**Maximum SPI Frequency, Master Input**

The maximum SPI master input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{1}{SPIn + t_{VALID}}, \frac{f_{CLKSPI} \times 2}{9})$$

Where $SPIn$ is the MISO setup and hold time, USPI0 + USPI1 or USPI3 + USPI4 depending on CPOL and NCPHA. $T_{VALID}$ is the SPI slave response time. Please refer to the SPI slave datasheet for $T_{VALID}$. $f_{CLKSPI}$ is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

*7.9.3.2 Slave mode*

**Figure 7-10.** USART in SPI Slave Mode With (CPOL= 0 and CPHA= 1) or (CPOL= 1 and CPHA= 0)

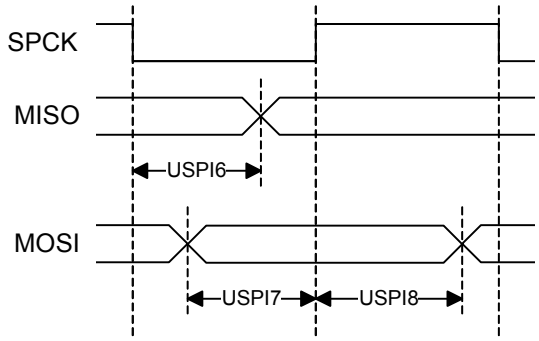



**Figure 7-11.** USART in SPI Slave Mode With (CPOL= CPHA= 0) or (CPOL= CPHA= 1)

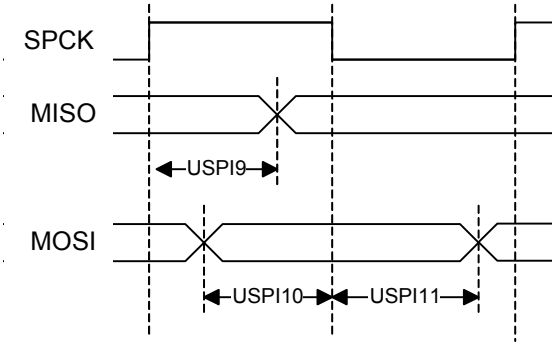


**Figure 7-12.** USART in SPI Slave Mode NPCS Timing

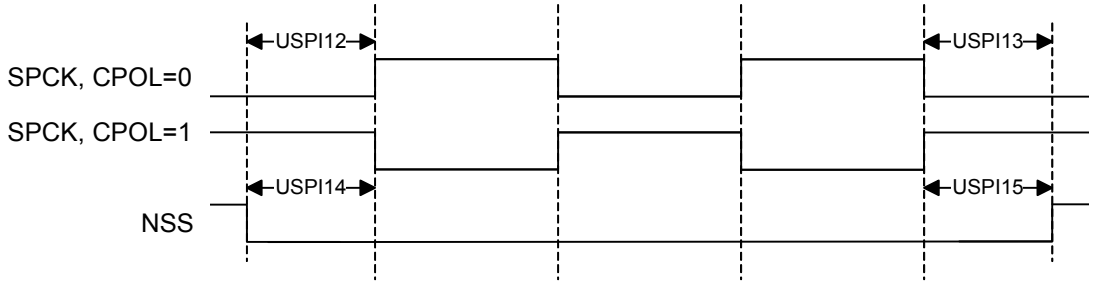



**Table 7-39.** USART in SPI mode Timing, Slave Mode[1]

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| USPI6 | SPCK falling to MISO delay | $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 40pF | | 27.6 | ns |
| USPI7 | MOSI setup time before SPCK rises | | $t_{SAMPLE}$[2] + $t_{CLK\_USART}$ | | |
| USPI8 | MOSI hold time after SPCK rises | | 0 | | |
| USPI9 | SPCK rising to MISO delay | | | 27.2 | |
| USPI10 | MOSI setup time before SPCK falls | | $t_{SAMPLE}$[2] + $t_{CLK\_USART}$ | | |
| USPI11 | MOSI hold time after SPCK falls | | 0 | | |
| USPI12 | NSS setup time before SPCK rises | | 25.0 | | |
| USPI13 | NSS hold time after SPCK falls | | 0 | | |
| USPI14 | NSS setup time before SPCK falls | | 25.0 | | |
| USPI15 | NSS hold time after SPCK rises | | 0 | | |

Notes: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

2. Where: $t_{SAMPLE} = t_{SPCK} - \left( \left\lfloor \frac{t_{SPCK}}{2 \times t_{CLKUSART}} \right\rfloor + \frac{1}{2} \right) \times t_{CLKUSART}$

### Maximum SPI Frequency, Slave Input Mode

The maximum SPI slave input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{f_{CLKSPI} \times 2}{9}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI setup and hold time, USPI7 + USPI8 or USPI10 + USPI11 depending on CPOL and NCPHA. $f_{CLKSPI}$ is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

### Maximum SPI Frequency, Slave Output Mode

The maximum SPI slave output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{f_{CLKSPI} \times 2}{9}, f_{PINMAX}, \frac{1}{SPIn + t_{SETUP}})$$

Where $SPIn$ is the MISO delay, USPI6 or USPI9 depending on CPOL and NCPHA. $T_{SETUP}$ is the SPI master setup time. Please refer to the SPI master datasheet for $T_{SETUP}$ . $f_{CLKSPI}$ is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock. $f_{PINMAX}$ is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

### 7.9.4 SPI Timing

#### *7.9.4.1 Master mode*

**Figure 7-13.** SPI Master Mode With (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)

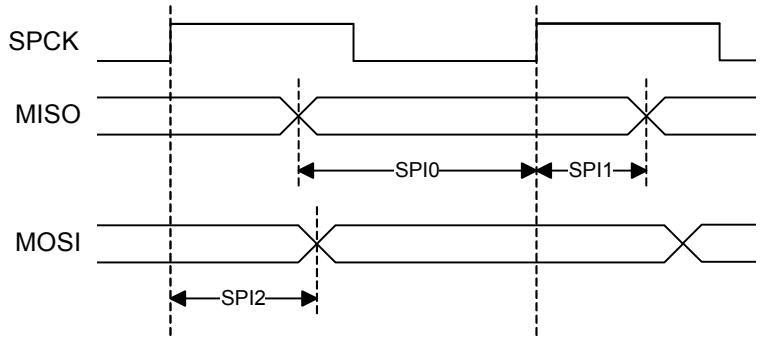



**Figure 7-14.** SPI Master Mode With (CPOL= 0 and NCPHA= 1) or (CPOL= 1 and NCPHA= 0)

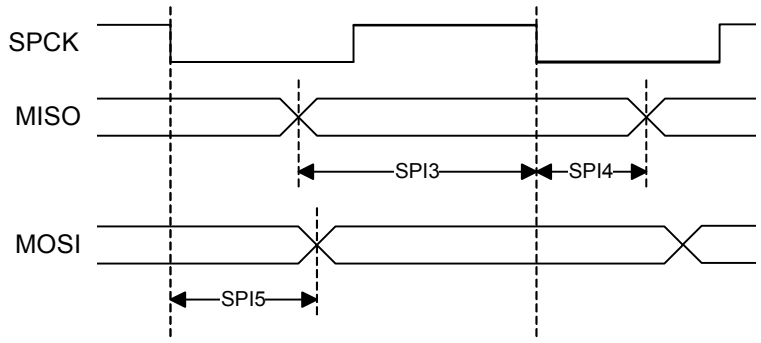


**Table 7-40.** SPI Timing, Master Mode[1]

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| SPI0 | MISO setup time before SPCK rises | $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 40pF | 28.4 + ($t_{CLK\_SPI}$)/2 | | ns |
| SPI1 | MISO hold time after SPCK rises | | 0 | | |
| SPI2 | SPCK rising to MOSI delay | | | 7.1 | |
| SPI3 | MISO setup time before SPCK falls | | 22.8 + ($t_{CLK\_SPI}$)/2 | | |
| SPI4 | MISO hold time after SPCK falls | | 0 | | |
| SPI5 | SPCK falling to MOSI delay | | | 11.0 | |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Maximum SPI Frequency, Master Output**

The maximum SPI master output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI delay, SPI2 or SPI5 depending on CPOL and NCPHA. $f_{PINMAX}$ is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

**Maximum SPI Frequency, Master Input**

The maximum SPI master input frequency is given by the following formula:

$$f_{SPCKMAX} = \frac{1}{SPIn + t_{VALID}}$$

Where $SPIn$ is the MISO setup and hold time, SPI0 + SPI1 or SPI3 + SPI4 depending on CPOL and NCPHA. $t_{VALID}$ is the SPI slave response time. Please refer to the SPI slave datasheet for $t_{VALID}$ .

### *7.9.4.2 Slave mode*

**Figure 7-15.** SPI Slave Mode With (CPOL= 0 and NCPHA= 1) or (CPOL= 1 and NCPHA= 0)

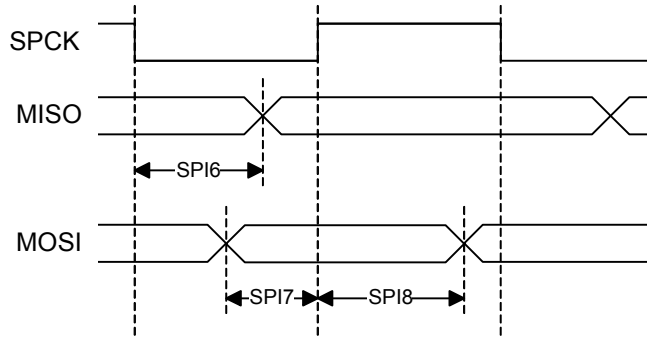


**Figure 7-16.** SPI Slave Mode With (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)

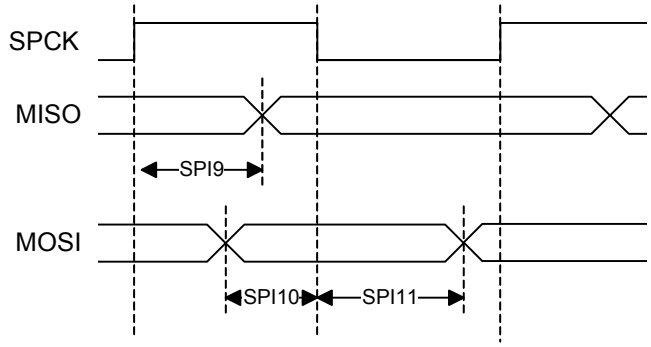



**Figure 7-17.** SPI Slave Mode NPCS Timing

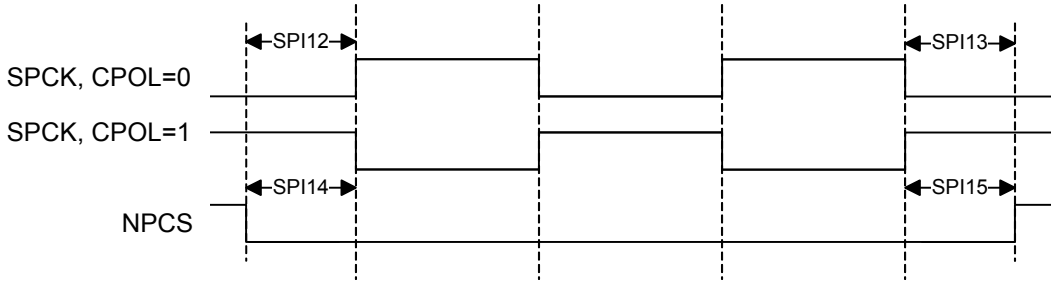



**Table 7-41.** SPI Timing, Slave Mode[1]

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| SPI6 | SPCK falling to MISO delay | $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 40pF | | 30.8 | ns |
| SPI7 | MOSI setup time before SPCK rises | | 0 | | |
| SPI8 | MOSI hold time after SPCK rises | | 4.1 | | |
| SPI9 | SPCK rising to MISO delay | | | 29.9 | |
| SPI10 | MOSI setup time before SPCK falls | | 0 | | |
| SPI11 | MOSI hold time after SPCK falls | | 3.5 | | |
| SPI12 | NPCS setup time before SPCK rises | | 1.9 | | |
| SPI13 | NPCS hold time after SPCK falls | | 0.2 | | |
| SPI14 | NPCS setup time before SPCK falls | | 2.2 | | |
| SPI15 | NPCS hold time after SPCK rises | | 0 | | |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

**Maximum SPI Frequency, Slave Input Mode**

The maximum SPI slave input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{CLKSPI}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI setup and hold time, SPI7 + SPI8 or SPI10 + SPI11 depending on CPOL and NCPHA. $f_{CLKSPI}$ is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

**Maximum SPI Frequency, Slave Output Mode**

The maximum SPI slave output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn + t_{SETUP}})$$

Where $SPIn$ is the MISO delay, SPI6 or SPI9 depending on CPOL and NCPHA. $t_{SETUP}$ is the SPI master setup time. Please refer to the SPI master datasheet for $t_{SETUP}$ . $f_{PINMAX}$ is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

### 7.9.5 TWIM/TWIS Timing

Figure 7-42 shows the TWI-bus timing requirements and the compliance of the device with them. Some of these requirements ($t_r$ and $t_f$) are met by the device without requiring user intervention. Compliance with the other requirements ($t_{HD-STA}$, $t_{SU-STA}$, $t_{SU-STO}$, $t_{HD-DAT}$, $t_{SU-DAT-TWI}$, $t_{LOW-TWI}$, $t_{HIGH}$, and $f_{TWCK}$) requires user intervention through appropriate programming of the relevant TWIM and TWIS user interface registers. Please refer to the TWIM and TWIS sections for more information.

**Table 7-42.** TWI-Bus Timing Requirements

| Symbol | Parameter | Mode | Minimum | | Maximum | | Unit |
|---|---|---|---|---|---|---|---|
| | | | Requirement | Device | Requirement | Device | |
| $t_r$ | TWCK and TWD rise time | Standard(1) | - | | 1000 | | ns |
| | | Fast(1) | $20 + 0.1C_b$ | | 300 | | |
| $t_f$ | TWCK and TWD fall time | Standard | - | | 300 | | ns |
| | | Fast | $20 + 0.1C_b$ | | 300 | | |
| $t_{HD-STA}$ | (Repeated) START hold time | Standard | 4 | $t_{clkpb}$ | - | | µs |
| | | Fast | 0.6 | | | | |
| $t_{SU-STA}$ | (Repeated) START set-up time | Standard | 4.7 | $t_{clkpb}$ | - | | µs |
| | | Fast | 0.6 | | | | |
| $t_{SU-STO}$ | STOP set-up time | Standard | 4.0 | $4t_{clkpb}$ | - | | µs |
| | | Fast | 0.6 | | | | |
| $t_{HD-DAT}$ | Data hold time | Standard | 0.3(2) | $2t_{clkpb}$ | 3.45() | $15t_{prescaled} + t_{clkpb}$ | µs |
| | | Fast | | | 0.9() | | |

**Table 7-42.** TWI-Bus Timing Requirements

| Symbol | Parameter | Mode | Minimum | | Maximum | | Unit |
|---|---|---|---|---|---|---|---|
| | | | Requirement | Device | Requirement | Device | |
| $t_{SU-DAT-TWI}$ | Data set-up time | Standard | 250 | $2t_{clkpb}$ | - | | ns |
| | | Fast | 100 | | | | |
| $t_{SU-DAT}$ | | - | - | $t_{clkpb}$ | - | | - |
| $t_{LOW-TWI}$ | TWCK LOW period | Standard | 4.7 | $4t_{clkpb}$ | - | | µs |
| | | Fast | 1.3 | | | | |
| $t_{LOW}$ | | - | - | $t_{clkpb}$ | - | | - |
| $t_{HIGH}$ | TWCK HIGH period | Standard | 4.0 | $8t_{clkpb}$ | - | | µs |
| | | Fast | 0.6 | | | | |
| $f_{TWCK}$ | TWCK frequency | Standard | - | | 100 | $\frac{1}{12t_{clkpb}}$ | kHz |
| | | Fast | | | 400 | | |

Notes: 1. Standard mode: $f_{TWCK} \leq 100$ kHz ; fast mode: $f_{TWCK} > 100$ kHz .
2. A device must internally provide a hold time of at least 300 ns for TWD with reference to the falling edge of TWCK.

Notations:

$C_b$ = total capacitance of one bus line in pF

$t_{clkpb}$ = period of TWI peripheral bus clock

$t_{prescaled}$ = period of TWI internal prescaled clock (see chapters on TWIM and TWIS)

The maximum $t_{HD;DAT}$ has only to be met if the device does not stretch the LOW period ($t_{LOW-TWI}$) of TWCK.

### 7.9.6 JTAG Timing

**Figure 7-18.** JTAG Interface Signals

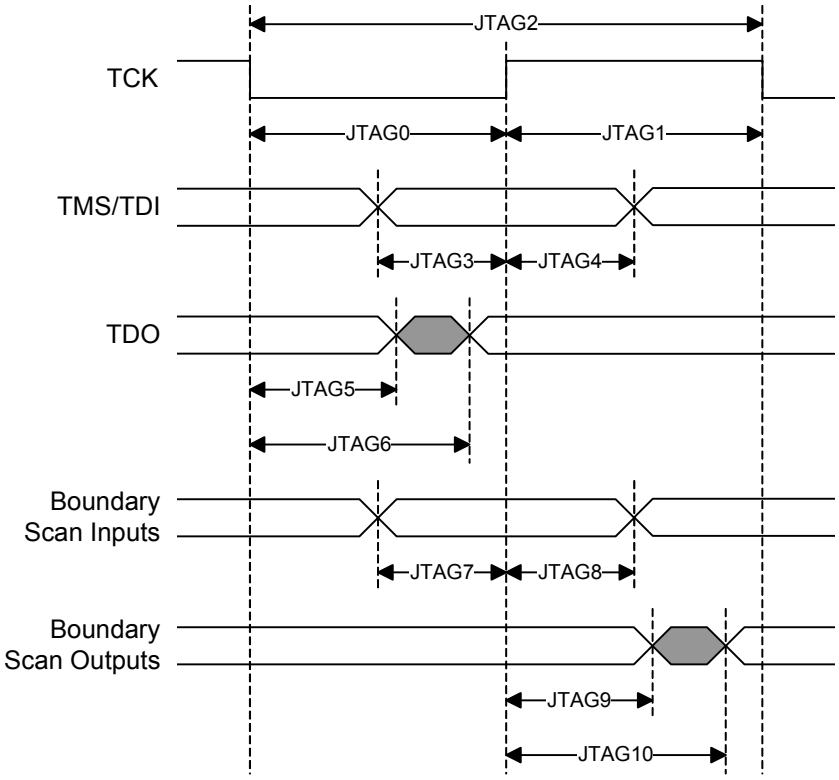



**Table 7-43.** JTAG Timings[1]

| Symbol | Parameter | Conditions | Min | Max | Units |
|---|---|---|---|---|---|
| JTAG0 | TCK Low Half-period | $V_{VDDIO}$ from 3.0V to 3.6V, maximum external capacitor = 40pF | 23.2 | | ns |
| JTAG1 | TCK High Half-period | | 8.8 | | |
| JTAG2 | TCK Period | | 32.0 | | |
| JTAG3 | TDI, TMS Setup before TCK High | | 3.9 | | |
| JTAG4 | TDI, TMS Hold after TCK High | | 0.6 | | |
| JTAG5 | TDO Hold Time | | 4.5 | | |
| JTAG6 | TCK Low to TDO Valid | | | 23.2 | |
| JTAG7 | Boundary Scan Inputs Setup Time | | 0 | | |
| JTAG8 | Boundary Scan Inputs Hold Time | | 5.0 | | |
| JTAG9 | Boundary Scan Outputs Hold Time | | 8.7 | | |
| JTAG10 | TCK to Boundary Scan Outputs Valid | | | 17.7 | |

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

# 8. Mechanical Characteristics

## 8.1 Thermal Considerations

### 8.1.1 Thermal Data

Table 8-1 summarizes the thermal resistance data depending on the package.

**Table 8-1.** Thermal Resistance Data

| Symbol | Parameter | Condition | Package | Typ | Unit |
|---|---|---|---|---|---|
| $\theta_{JA}$ | Junction-to-ambient thermal resistance | Still Air | TQFP48 | 63.2 | °C/W |
| $\theta_{JC}$ | Junction-to-case thermal resistance | | TQFP48 | 21.8 | |
| $\theta_{JA}$ | Junction-to-ambient thermal resistance | Still Air | QFN48 | 28.3 | °C/W |
| $\theta_{JC}$ | Junction-to-case thermal resistance | | QFN48 | 2.5 | |
| $\theta_{JA}$ | Junction-to-ambient thermal resistance | Still Air | TLLGA48 | 25.4 | °C/W |
| $\theta_{JC}$ | Junction-to-case thermal resistance | | TLLGA48 | 12.7 | |

### 8.1.2 Junction Temperature

The average chip-junction temperature, $T_J$, in °C can be obtained from the following:

1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

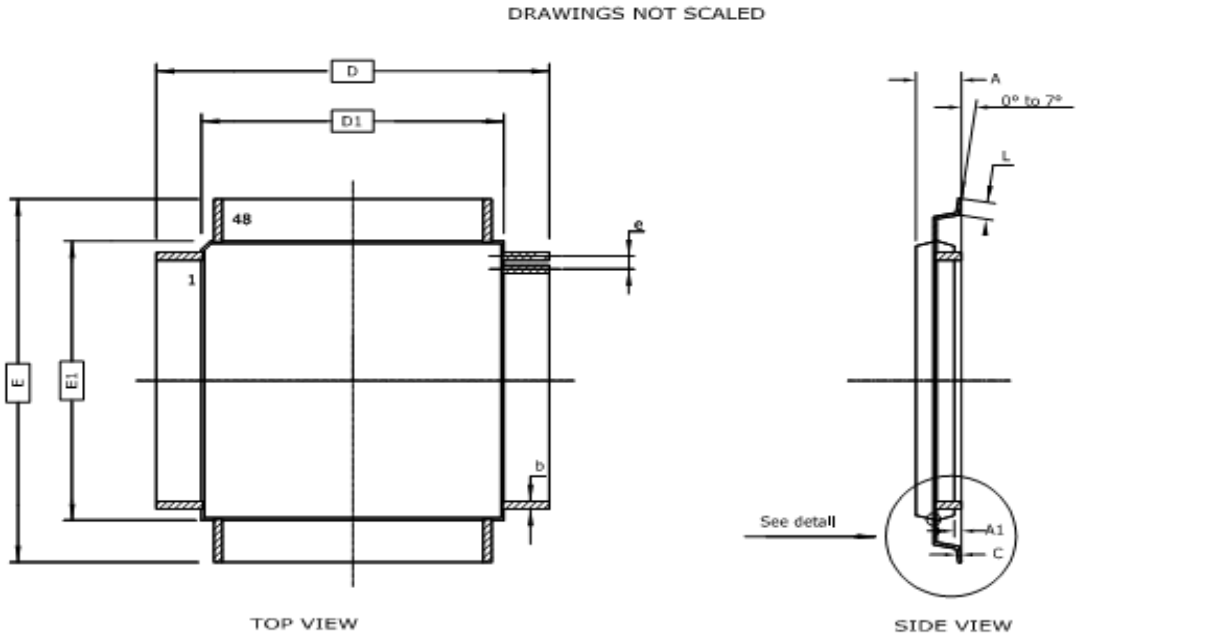
where:

- $\theta_{JA}$ = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 8-1.
- $\theta_{JC}$ = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 8-1.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- $P_D$ = device power consumption (W) estimated from data provided in the Section 7.4 on page 42.
- $T_A$ = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature $T_J$ in °C.

## 8.2 Package Drawings

**Figure 8-1.** TQFP-48 Package Drawing




COMMON DIMENSIONS

(Unit of Measure = mm)

| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | --- | --- | 1.20 | |
| A1 | 0.05 | --- | 0.15 | |
| A2 | 0.95 | --- | 1.05 | |
| C | 0.09 | --- | 0.20 | |
| D/E | | 9.00 BSC | | |
| D1/E1 | | 7.00 BSC | | |
| L | 0.45 | --- | 0.75 | |
| b | 0.17 | --- | 0.27 | |
| e | | 0.50 BSC | | |

Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MS-026, Variation ABC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10mm maximum.

10/04/2011

**Table 8-2.** Device and Package Maximum Weight

| | |
|---|---|
| 140 | mg |

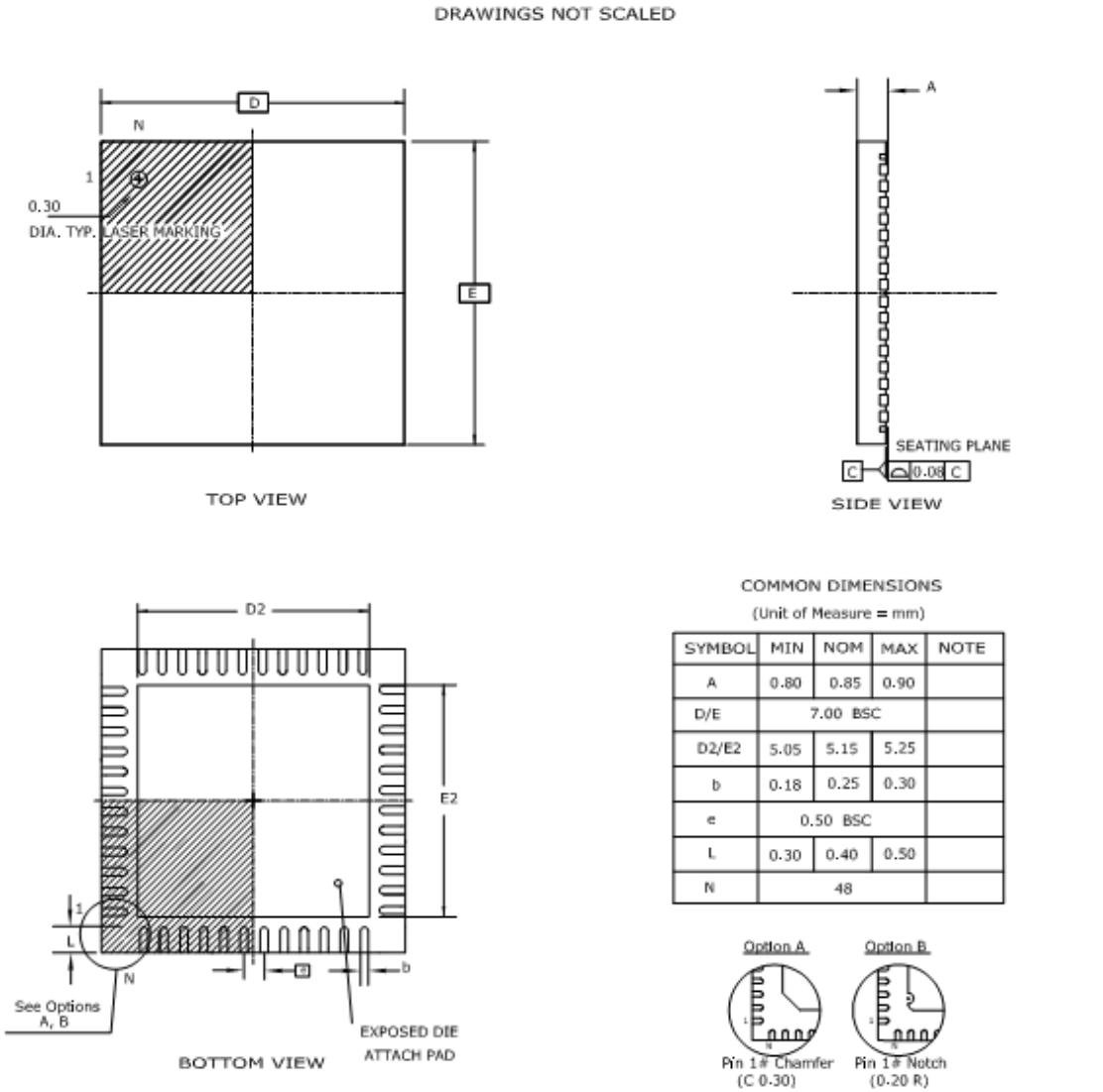
**Table 8-3.** Package Characteristics

| | |
|---|---|
| Moisture Sensitivity Level | MSL3 |

**Table 8-4.** Package Reference

| | |
|---|---|
| JEDEC Drawing Reference | MS-026 |
| JESD97 Classification | E3 |

**Figure 8-2.** QFN-48 Package Drawing




| SYMBOL | MIN | NOM | MAX | NOTE |
|---|---|---|---|---|
| A | 0.80 | 0.85 | 0.90 | |
| D/E | | 7.00 BSC | | |
| D2/E2 | 5.05 | 5.15 | 5.25 | |
| b | 0.18 | 0.25 | 0.30 | |
| e | | 0.50 BSC | | |
| L | 0.30 | 0.40 | 0.50 | |
| N | | 48 | | |

Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation VKKD-4, for proper dimensions, tolerances, datums, etc.
2. Dimension b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip.
If the terminal has the optical radius on the other end of the terminal, the dimension should not be measured in that radius area.

07/27/2011

Note: The exposed pad is not connected to anything internally, but should be soldered to ground to increase board level reliability.

**Table 8-5.** Device and Package Maximum Weight

| | |
|---|---|
| 140 | mg |

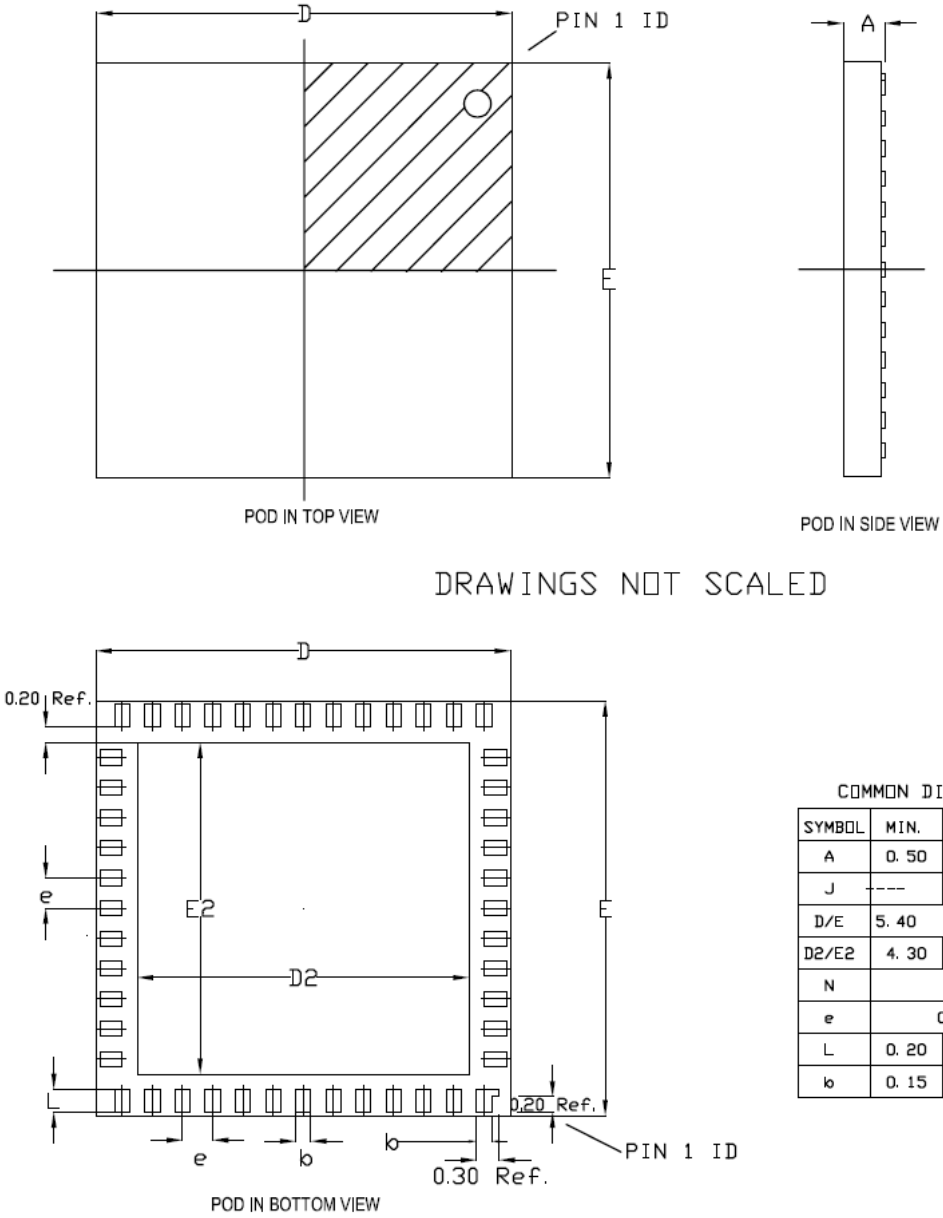
**Table 8-6.** Package Characteristics

| | |
|---|---|
| Moisture Sensitivity Level | MSL3 |

**Table 8-7.** Package Reference

| | |
|---|---|
| JEDEC Drawing Reference | M0-220 |
| JESD97 Classification | E3 |

**Figure 8-3.** TLLGA-48 Package Drawing




COMMON DIMENSIONS IN MM

| SYMBOL | MIN. | NOM. | MAX. | NOTES |
|---|---|---|---|---|
| A | 0.50 | 0.55 | 0.60 | |
| J | ---- | ---- | ---- | |
| D/E | 5.40 | 5.50 | 5.60 | |
| D2/E2 | 4.30 | 4.40 | 4.50 | |
| N | 48 | | | |
| e | 0.40 BSC | | | |
| L | 0.20 | 0.30 | 0.40 | |
| b | 0.15 | 0.20 | 0.25 | |

**Table 8-8.** Device and Package Maximum Weight

| | |
|---|---|
| 39.3 | mg |

**Table 8-9.** Package Characteristics

| | |
|---|---|
| Moisture Sensitivity Level | MSL3 |

**Table 8-10.** Package Reference

| | |
|---|---|
| JEDEC Drawing Reference | N/A |
| JESD97 Classification | E4 |

## 8.3 Soldering Profile

Table 8-11 gives the recommended soldering profile from J-STD-20.

**Table 8-11.** Soldering Profile

| Profile Feature | Green Package |
|---|---|
| Average Ramp-up Rate (217°C to Peak) | 3°C/s max |
| Preheat Temperature 175°C ±25°C | 150-200°C |
| Time Maintained Above 217°C | 60-120s |
| Time within 5°C of Actual Peak Temperature | 30s |
| Peak Temperature Range | 260°C |
| Ramp-down Rate | 6°C/s max |
| Time 25°C to Peak Temperature | 8 minutes max |

A maximum of three reflow passes is allowed per component.

# 9. Ordering Information

**Table 9-1.** Ordering Information

| Device | Ordering Code | Carrier Type | Package | Package Type | Temperature Operating Range |
|---|---|---|---|---|---|
| AT32UC3L064 | AT32UC3L064-AUTES | ES | TQFP 48 | JESD97 Classification E3 | Industrial (-40°C to 85°C) |
| | AT32UC3L064-AUT | Tray | | | |
| | AT32UC3L064-AUR | Tape & Reel | | | |
| | AT32UC3L064-ZAUES | ES | QFN 48 | | |
| | AT32UC3L064-ZAUT | Tray | | | |
| | AT32UC3L064-ZAUR | Tape & Reel | | | |
| | AT32UC3L064-D3HES | ES | TLLGA 48 | JESD97 Classification E4 | |
| | AT32UC3L064-D3HT | Tray | | | |
| | AT32UC3L064-D3HR | Tape & Reel | | | |
| AT32UC3L032 | AT32UC3L032-AUT | Tray | TQFP 48 | JESD97 Classification E3 | |
| | AT32UC3L032-AUR | Tape & Reel | | | |
| | AT32UC3L032-ZAUT | Tray | QFN 48 | | |
| | AT32UC3L032-ZAUR | Tape & Reel | | | |
| | AT32UC3L032-D3HT | Tray | TLLGA 48 | JESD97 Classification E4 | |
| | AT32UC3L032-D3HR | Tape & Reel | | | |
| AT32UC3L016 | AT32UC3L016-AUT | Tray | TQFP 48 | JESD97 Classification E3 | |
| | AT32UC3L016-AUR | Tape & Reel | | | |
| | AT32UC3L016-ZAUT | Tray | QFN 48 | | |
| | AT32UC3L016-ZAUR | Tape & Reel | | | |
| | AT32UC3L016-D3HT | Tray | TLLGA 48 | JESD97 Classification E4 | |
| | AT32UC3L016-D3HR | Tape & Reel | | | |

# 10. Errata

## 10.1 Rev. E

### 10.1.1 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.

### 10.1.2 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).

### 10.1.3 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**

If the CPU is in idle or frozen sleep mode and a module is in a state that triggers sleep walking, all PB clocks will be masked except the PB clock to the sleepwalking module.
**Fix/Workaround**
Mask all clock requests in the PM.PPCR register before going into idle or frozen mode.

### 10.1.4 SCIF

1. **PCLKSR.OSC32RDY bit might not be cleared after disabling OSC32K**
In some cases the OSC32RDY bit in the PCLKSR register will not be cleared when OSC32K is disabled.
**Fix/Workaround**
When re-enabling the OSC32K, read the PCLKSR.OSC32RDY bit. If this bit is:
0: Follow normal procedures.
1: Ignore the PCLKSR.OSC32RDY and ISR.OSC32RDY bit. Use the Frequency Meter (FREQM) to determine if the OSC32K clock is ready. The OSC32K clock is ready when the FREQM measures a non-zero frequency.

2. **The RC32K output on PA20 is not always permanently disabled**
The RC32K output on PA20 may sometimes re-appear.
**Fix/Workaround**
Before using RC32K for other purposes, the following procedure has to be followed in order to properly disable it:
- Run the CPU on RCSYS
- Disable the output to PA20 by writing a zero to PM.PPCR.RC32OUT
- Enable RC32K by writing a one to SCIF.RC32KCR.EN, and wait for this bit to be read as one
- Disable RC32K by writing a zero to SCIF.RC32KCR.EN, and wait for this bit to be read as zero.

### 10.1.5 AST

1. **Reset may set status bits in the AST**
If a reset occurs and the AST is enabled, the SR.ALARM0, SR.PER0, and SR.OVF bits may be set.
**Fix/Workaround**
If the part is reset and the AST is used, clear all bits in the Status Register before entering sleep mode.

2. **AST wake signal is released one AST clock cycle after the BUSY bit is cleared**
After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
**Fix/Workaround**
Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

### 10.1.6 WDT

1. **Clearing the Watchdog Timer (WDT) counter in second half of timeout period will issue a Watchdog reset**
If the WDT counter is cleared in the second half of the timeout period, the WDT will immediately issue a Watchdog reset.
**Fix/Workaround**

Use twice as long timeout period as needed and clear the WDT counter within the first half of the timeout period. If the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT counter is not cleared at all, the time before the reset will be twice as long as needed.

2. **WDT Control Register does not have synchronization feedback**
   When writing to the Timeout Prescale Select (PSEL), Time Ban Prescale Select (TBAN), Enable (EN), or WDT Mode (MODE) fieldss of the WDT Control Register (CTRL), a synchronizer is started to propagate the values to the WDT clcok domain. This synchronization takes a finite amount of time, but only the status of the synchronization of the EN bit is reflected back to the user. Writing to the synchronized fields during synchronization can lead to undefined behavior.
   **Fix/Workaround**
   -When writing to the affected fields, the user must ensure a wait corresponding to 2 clock cycles of both the WDT peripheral bus clock and the selected WDT clock source.
   -When doing writes that changes the EN bit, the EN bit can be read back until it reflects the written value.

### 10.1.7 GPIO

1. **Clearing GPIO interrupt may fail**
   Writing a one to the GPIO.IFRC register to clear an interrupt will be ignored if interrupt is enabled for the corresponding port.
   **Fix/Workaround**
   Disable the interrupt, clear it by writing a one to GPIO.IFRC, then enable the interrupt.

### 10.1.8 SPI

1. **SPI data transfer hangs with CSR0.CSAAT==1 and MR.MODFDIS==0**
   When CSR0.CSAAT==1 and mode fault detection is enabled (MR.MODFDIS==0), the SPI module will not start a data transfer.
   **Fix/Workaround**
   Disable mode fault detection by writing a one to MR.MODFDIS.

2. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.
   **Fix/Workaround**
   Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

3. **SPI disable does not work in SLAVE mode**
   SPI disable does not work in SLAVE mode.
   **Fix/Workaround**
   Read the last received data, then perform a software reset by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

4. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
   When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.

**Fix/Workaround**
When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

5. **SPI mode fault detection enable causes incorrect behavior**
When mode fault detection is enabled (MR.MODFDIS==0), the SPI module may not operate properly.
**Fix/Workaround**
Always disable mode fault detection before using the SPI by writing a one to MR.MODFDIS.

### 10.1.9 TWI

1. **TWIM SR.IDLE goes high immediately when NAK is received**
When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
**Fix/Workaround**
If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

2. **TWIM TWALM polarity is wrong**
The TWALM signal in the TWIM is active high instead of active low.
**Fix/Workaround**
Use an external inverter to invert the signal going into the TWIM. When using both TWIM and TWIS on the same pins, the TWALM cannot be used.

3. **TWIS may not wake the device from sleep mode**
If the CPU is put to a sleep mode (except Idle and Frozen) directly after a TWI Start condition, the CPU may not wake upon a TWIS address match. The request is NACKed.
**Fix/Workaround**
When using the TWI address match to wake the device from sleep, do not switch to sleep modes deeper than Frozen. Another solution is to enable asynchronous EIC wake on the TWIS clock (TWCK) or TWIS data (TWD) pins, in order to wake the system up on bus events.

### 10.1.10 PWMA

1. **BUSY bit is never cleared after writes to the Control Register (CR)**
When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR when the PWMA is disabled (CR.EN == 0), the BUSY bit in the Status Register (SR.BUSY) will be set, but never cleared.
**Fix/Workaround**
When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR, make sure the PWMA is enabled, or simultaneously enable the PWMA by writing a one to CR.EN.

2. **Incoming peripheral events are discarded during duty cycle register update**
Incoming peripheral events to all applied channels will be discarded if a duty cycle update is received from the user interface in the same PWMA clock period.
**Fix/Workaround**

Ensure that duty cycle writes from the user interface are not performed in a PWMA period when an incoming peripheral event is expected.

### 10.1.11 ADCIFB

1. **Using STARTUPTIME larger than 0x1F will freeze the ADC**
   Writing a value larger than 0x1F to the Startup Time field in the ADC Configuration Register (ACR.STARTUP) will freeze the ADC, and the Busy Status bit in the Status Register (SR.BUSY) will never be cleared.
   **Fix/Workaround**
   Do not write values larger than 0x1F to ACR.STARTUP.

### 10.1.12 CAT

1. **CAT asynchronous wake will be delayed by one AST event period**
   If the CAT detects a condition the should asynchronously wake the device in static mode, the asynchronous wake will not occur until the next AST event. For example, if the AST is generating events to the CAT every 50ms, and the CAT detects a touch at t=9200ms, the asynchronous wake will occur at t=9250ms.
   **Fix/Workaround**
   None.

2. **CAT QMatrix sense capacitors discharged prematurely**
   At the end of a QMatrix burst charging sequence that uses different burst count values for different Y lines, the Y lines may be incorrectly grounded for up to n-1 periods of the peripheral bus clock, where n is the ratio of the PB clock frequency to the GCLK_CAT frequency. This results in premature loss of charge from the sense capacitors and thus increased variability of the acquired count values.
   **Fix/Workaround**
   Enable the 1kOhm drive resistors on all implemented QMatrix Y lines (CSA 1, 3, 5, 7, 9, 11, 13, and/or 15) by writing ones to the corresponding odd bits of the CSARES register.

### 10.1.13 aWire

1. **aWire CPU clock speed robustness**
   The aWire memory speed request command counter warps at clock speeds below approximately 5kHz.
   **Fix/Workaround**
   None.

2. **The aWire debug interface is reset after leaving Shutdown mode**
   If the aWire debug mode is used as debug interface and the program enters Shutdown mode, the aWire interface will be reset when the part receives a wakeup either from the WAKE_N pin or the AST.
   **Fix/Workaround**
   None.

### 10.1.14 CHIP

1. **Increased Power Consumption in VDDIO in sleep modes**
   If OSC0 is enabled in crystal mode when entering a sleep mode where the OSC0 is disabled, this will lead to an increased power consumption in VDDIO.
   **Fix/Workaround**

Solution 1: Disable OSC0 by writing a zero to the Oscillator Enable bit in the System Control Interface (SCIF) Oscillator Control Register (SCIF.OSC0CTRL.OSCEN) before going to a sleep mode where OSC0 is disabled.
Solution 2: Pull down or up XIN0 or XOUT0 with 1 MOhm resistor.

#### 10.1.15 I/O Pins

1. **PA17 has low ESD tolerance**
   PA17 only tolerates 500 V ESD pulses (Human Body Model).
   **Fix/Workaround**
   Care must be taken during manufacturing and PCB design.

### 10.2 Rev. D

#### 10.2.1 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.

#### 10.2.2 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).

#### 10.2.3 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. **External reset in Shutdown mode**
   If an external reset is asserted while the device is in Shutdown mode, the Power Manager will register this as a Power-on reset (POR), and not as a SLEEP reset, in the Reset Cause register (RCAUSE)
   **Fix/Workaround**
   None.

3. **Disabling POR33 may generate spurious resets**
   Depending on operating conditions, POR33 may generate a spurious reset in one of the following cases:
   - When POR33 is disabled from the user interface
   - When SM33 supply monitor is enabled
   - When entering Shutdown mode while debugging the chip using JTAG or aWire interface

   In the listed cases, writing a one to the bit VREGCR.POR33MASK in the System Control Interface (SCIF) to mask the POR33 reset will be ineffective
   **Fix/Workaround**
   - Do not disable POR33 using the user interface
   - Do not use the SM33 supply monitor
   - Do not enter Shutdown mode if a debugger is connected to the chip

4. **Instability when exiting sleep walking**
   If all the following operating conditions are true, exiting sleep walking might lead to instability:
   - The OSC0 is enabled in external clock mode (OSCCTRL0.OSCEN == 1 and OSCCTRL0.MODE == 0)
   - A sleep mode where the OSC0 is automatically disabled is entered
   - The device enters sleep walking

   **Fix/Workaround**
   Do not run OSC0 in external clock mode if sleepwalking is expected to be used.

5. **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.

6. **Sleepwalking in idle and frozen sleep mode will mask all other PB clocks**
   If the CPU is in idle or frozen sleep mode and a module is in a state that triggers sleep walking, all PB clocks will be masked except the PB clock to the sleepwalking module.
   **Fix/Workaround**
   Mask all clock requests in the PM.PPCR register before going into idle or frozen mode.

### 10.2.4 SCIF

1. **PCLKSR.OSC32RDY bit might not be cleared after disabling OSC32K**
   In some cases the OSC32RDY bit in the PCLKSR register will not be cleared when OSC32K is disabled.
   **Fix/Workaround**

When re-enabling the OSC32K, read the PCLKSR.OSC32RDY bit. If this bit is:
0: Follow normal procedures.
1: Ignore the PCLKSR.OSC32RDY and ISR.OSC32RDY bit. Use the Frequency Meter (FREQM) to determine if the OSC32K clock is ready. The OSC32K clock is ready when the FREQM measures a non-zero frequency.

2. **The RC32K output on PA20 is not always permanently disabled**
The RC32K output on PA20 may sometimes re-appear.
**Fix/Workaround**
Before using RC32K for other purposes, the following procedure has to be followed in order to properly disable it:
- Run the CPU on RCSYS
- Disable the output to PA20 by writing a zero to PM.PPCR.RC32OUT
- Enable RC32K by writing a one to SCIF.RC32KCR.EN, and wait for this bit to be read as one
- Disable RC32K by writing a zero to SCIF.RC32KCR.EN, and wait for this bit to be read as zero.

### 10.2.5 AST

1. **Reset may set status bits in the AST**
If a reset occurs and the AST is enabled, the SR.ALARM0, SR.PER0, and SR.OVF bits may be set.
**Fix/Workaround**
If the part is reset and the AST is used, clear all bits in the Status Register before entering sleep mode.

2. **AST wake signal is released one AST clock cycle after the BUSY bit is cleared**
After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
**Fix/Workaround**
Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

### 10.2.6 WDT

1. **Clearing the Watchdog Timer (WDT) counter in second half of timeout period will issue a Watchdog reset**
If the WDT counter is cleared in the second half of the timeout period, the WDT will immediately issue a Watchdog reset.
**Fix/Workaround**
Use twice as long timeout period as needed and clear the WDT counter within the first half of the timeout period. If the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT counter is not cleared at all, the time before the reset will be twice as long as needed.

2. **WDT Control Register does not have synchronization feedback**
When writing to the Timeout Prescale Select (PSEL), Time Ban Prescale Select (TBAN), Enable (EN), or WDT Mode (MODE) fieldss of the WDT Control Register (CTRL), a synchronizer is started to propagate the values to the WDT clcok domain. This synchronization takes a finite amount of time, but only the status of the synchronization of the EN bit is reflected back to the user. Writing to the synchronized fields during synchronization can lead to undefined behavior.

**Fix/Workaround**
-When writing to the affected fields, the user must ensure a wait corresponding to 2 clock cycles of both the WDT peripheral bus clock and the selected WDT clock source.
-When doing writes that changes the EN bit, the EN bit can be read back until it reflects the written value.

### 10.2.7 GPIO

1. **Clearing GPIO interrupt may fail**
   Writing a one to the GPIO.IFRC register to clear an interrupt will be ignored if interrupt is enabled for the corresponding port.
   **Fix/Workaround**
   Disable the interrupt, clear it by writing a one to GPIO.IFRC, then enable the interrupt.

### 10.2.8 SPI

1. **SPI data transfer hangs with CSR0.CSAAT==1 and MR.MODFDIS==0**
   When CSR0.CSAAT==1 and mode fault detection is enabled (MR.MODFDIS==0), the SPI module will not start a data transfer.
   **Fix/Workaround**
   Disable mode fault detection by writing a one to MR.MODFDIS.

2. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.
   **Fix/Workaround**
   Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

3. **SPI disable does not work in SLAVE mode**
   SPI disable does not work in SLAVE mode.
   **Fix/Workaround**
   Read the last received data, then perform a software reset by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

4. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
   When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
   **Fix/Workaround**
   When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

5. **SPI mode fault detection enable causes incorrect behavior**
   When mode fault detection is enabled (MR.MODFDIS==0), the SPI module may not operate properly.
   **Fix/Workaround**
   Always disable mode fault detection before using the SPI by writing a one to MR.MODFDIS.

### 10.2.9 TWI

1. **TWIM SR.IDLE goes high immediately when NAK is received**
   When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
   **Fix/Workaround**
   If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

2. **TWIM TWALM polarity is wrong**
   The TWALM signal in the TWIM is active high instead of active low.
   **Fix/Workaround**
   Use an external inverter to invert the signal going into the TWIM. When using both TWIM and TWIS on the same pins, the TWALM cannot be used.

3. **TWIS may not wake the device from sleep mode**
   If the CPU is put to a sleep mode (except Idle and Frozen) directly after a TWI Start condition, the CPU may not wake upon a TWIS address match. The request is NACKed.
   **Fix/Workaround**
   When using the TWI address match to wake the device from sleep, do not switch to sleep modes deeper than Frozen. Another solution is to enable asynchronous EIC wake on the TWIS clock (TWCK) or TWIS data (TWD) pins, in order to wake the system up on bus events.

### 10.2.10 PWMA

1. **BUSY bit is never cleared after writes to the Control Register (CR)**
   When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR when the PWMA is disabled (CR.EN == 0), the BUSY bit in the Status Register (SR.BUSY) will be set, but never cleared.
   **Fix/Workaround**
   When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR, make sure the PWMA is enabled, or simultaneously enable the PWMA by writing a one to CR.EN.

2. **Incoming peripheral events are discarded during duty cycle register update**
   Incoming peripheral events to all applied channels will be discarded if a duty cycle update is received from the user interface in the same PWMA clock period.
   **Fix/Workaround**
   Ensure that duty cycle writes from the user interface are not performed in a PWMA period when an incoming peripheral event is expected.

### 10.2.11 ADCIFB

1. **Using STARTUPTIME larger than 0x1F will freeze the ADC**
   Writing a value larger than 0x1F to the Startup Time field in the ADC Configuration Register (ACR.STARTUP) will freeze the ADC, and the Busy Status bit in the Status Register (SR.BUSY) will never be cleared.
   **Fix/Workaround**
   Do not write values larger than 0x1F to ACR.STARTUP.

### 10.2.12 CAT

1. **CAT asynchronous wake will be delayed by one AST event period**
   If the CAT detects a condition the should asynchronously wake the device in static mode, the asynchronous wake will not occur until the next AST event. For example, if the AST is generating events to the CAT every 50ms, and the CAT detects a touch at t=9200ms, the asynchronous wake will occur at t=9250ms.
   **Fix/Workaround**
   None.

2. **CAT QMatrix sense capacitors discharged prematurely**
   At the end of a QMatrix burst charging sequence that uses different burst count values for different Y lines, the Y lines may be incorrectly grounded for up to n-1 periods of the peripheral bus clock, where n is the ratio of the PB clock frequency to the GCLK_CAT frequency. This results in premature loss of charge from the sense capacitors and thus increased variability of the acquired count values.
   **Fix/Workaround**
   Enable the 1kOhm drive resistors on all implemented QMatrix Y lines (CSA 1, 3, 5, 7, 9, 11, 13, and/or 15) by writing ones to the corresponding odd bits of the CSARES register.

### 10.2.13 aWire

1. **aWire CPU clock speed robustness**
   The aWire memory speed request command counter warps at clock speeds below approximately 5kHz.
   **Fix/Workaround**
   None.

2. **The aWire debug interface is reset after leaving Shutdown mode**
   If the aWire debug mode is used as debug interface and the program enters Shutdown mode, the aWire interface will be reset when the part receives a wakeup either from the WAKE_N pin or the AST.
   **Fix/Workaround**
   None.

### 10.2.14 CHIP

1. **In 3.3V Single Supply Mode, the Analog Comparator inputs affects the device's ability to start**
   When using the 3.3V Single Supply Mode the state of the Analog Comparator input pins can affect the device's ability to release POR reset. This is due to an interaction between the Analog Comparator input pins and the POR circuitry. The issue is not present in the 1.8V Supply Mode or the 3.3V Supply Mode with 1.8V Regulated I/O Lines.
   **Fix/Workaround**
   ACREFN (pin PA16) must be connected to GND until the POR reset is released and the Analog Comparator inputs should not be driven higher than 1.0V until the POR reset is released.

2. **Increased Power Consumption in VDDIO in sleep modes**
   If OSC0 is enabled in crystal mode when entering a sleep mode where the OSC0 is disabled, this will lead to an increased power consumption in VDDIO.
   **Fix/Workaround**

Solution 1: Disable OSC0 by writing a zero to the Oscillator Enable bit in the System Control Interface (SCIF) Oscillator Control Register (SCIF.OSC0CTRL.OSCEN) before going to a sleep mode where OSC0 is disabled.
Solution 2: Pull down or up XIN0 or XOUT0 with 1 MOhm resistor.

#### 10.2.15 I/O Pins

1. **PA17 has low ESD tolerance**
   PA17 only tolerates 500 V ESD pulses (Human Body Model).
   **Fix/Workaround**
   Care must be taken during manufacturing and PCB design.

## 10.3 Rev. C

Not sampled.

## 10.4 Rev. B

#### 10.4.1 Processor and Architecture

1. **RETS behaves incorrectly when MPU is enabled**
   RETS behaves incorrectly when MPU is enabled and MPU is configured so that system stack is not readable in unprivileged mode.
   **Fix/Workaround**
   Make system stack readable in unprivileged mode, or return from supervisor mode using rete instead of rets. This requires:
   1. Changing the mode bits from 001 to 110 before issuing the instruction. Updating the mode bits to the desired value must be done using a single mtsr instruction so it is done atomically. Even if this step is generally described as not safe in the UC technical reference manual, it is safe in this very specific case.
   2. Execute the RETE instruction.

2. **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.

3. **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.

#### 10.4.2 PDCA

1. **PCONTROL.CHxRES is non-functional**
   PCONTROL.CHxRES is non-functional. Counters are reset at power-on, and cannot be reset by software.
   **Fix/Workaround**
   Software needs to keep history of performance counters.

2. **Transfer error will stall a transmit peripheral handshake interface**
   If a transfer error is encountered on a channel transmitting to a peripheral, the peripheral handshake of the active channel will stall and the PDCA will not do any more transfers on the affected peripheral handshake interface.
   **Fix/Workaround**
   Disable and then enable the peripheral after the transfer error.

3. **VERSION register reads 0x120**
   The VERSION register reads 0x120 instead of 0x122.
   **Fix/Workaround**
   None.

### 10.4.3 FLASHCDW

1. **Fuse Programming**
   Programming fuses does not work.
   **Fix/Workaround**
   Do not program fuses. All fuses will be erased during chip erase command.

2. **Chip Erase**
   When performing a chip erase, the device may report that it is protected (IR=0x11) and that the erase failed, even if it was successful.
   **Fix/Workaround**
   Perform a reset before any further read and programming.

3. Wait 500 ns before reading from the flash after switching read mode
   After switching between normal read mode and high-speed read mode, the application must wait at least 500ns before attempting any access to the flash.
   **Fix/Workaround**
   Solution 1: Make sure that the appropriate instructions are executed from RAM, and that a waiting-loop is executed from RAM waiting 500ns or more before executing from flash.
   Solution 2. Execute from flash with a clock with period longer than 500ns. This guarantees that no new read access is attempted before the flash has had time to settle in the new read mode.

4. **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).

5. **VERSION register reads 0x100**
   The VERSION register reads 0x100 instead of 0x102.
   **Fix/Workaround**
   None.

### 10.4.4 SAU

1. **The SR.IDLE bit reads as zero**
   The IDLE bit in the Status Register (SR.IDLE) reads as zero.

**Fix/Workaround**
None.

2. **Open Mode is not functional**
The Open Mode is not functional.
**Fix/Workaround**
None.

3. **VERSION register reads 0x100**
The VERSION register reads 0x100 instead of 0x110.
**Fix/Workaround**
None.

### 10.4.5 HMATRIX

1. **In the PRAS and PRBS registers, the MxPR fields are only two bits**
In the PRAS and PRBS registers, the MxPR fields are only two bits wide, instead of four bits. The unused bits are undefined when reading the registers.
**Fix/Workaround**
Mask undefined bits when reading PRAS and PRBS.

### 10.4.6 Power Manager

1. **CONFIG register reads 0x4F**
The CONFIG register reads 0x4F instead of 0x43.
**Fix/Workaround**
None.

2. **It is not possible to mask the request clock requests**
It is not possible to mask the request clock requests using PPCR.
**Fix/Workaround**
None.

3. **Static mode cannot be entered if the WDT is using OSC32**
If the WDT is using OSC32 as clock source and the user tries to enter Static mode, the Deepstop mode will be entered instead.
**Fix/Workaround**
None.

4. **Clock Failure Detector (CFD) does not work**
Clock Failure Detector (CFD) does not work.
**Fix/Workaround**
None.

5. **WCAUSE register should not be used**
The WCAUSE register should not be used.
**Fix/Workaround**
None.

6. **PB writes via debugger in sleep modes are blocked during sleepwalking**
During sleepwalking, PB writes performed by a debugger will be discarded by all PB modules except the module that is requesting the clock.
**Fix/Workaround**

None.

7. **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.

8. **Disabling POR33 may generate spurious resets**
   Depending on operating conditions, POR33 may generate a spurious reset in one of the following cases:
   - When POR33 is disabled from the user interface
   - When SM33 supply monitor is enabled
   - When entering Shutdown mode while debugging the chip using JTAG or aWire interface

   In the listed cases, writing a one to the bit VREGCR.POR33MASK in the System Control Interface (SCIF) to mask the POR33 reset will be ineffective
   **Fix/Workaround**
   - Do not disable POR33 using the user interface
   - Do not use the SM33 supply monitor
   - Do not enter Shutdown mode if a debugger is connected to the chip

9. **Instability when exiting sleep walking**
   If all the following operating conditions are true, exiting sleep walking might lead to instability:
   - The OSC0 is enabled in external clock mode (OSCCTRL0.OSCEN == 1 and OSCCTRL0.MODE == 0)
   - A sleep mode where the OSC0 is automatically disabled is entered
   - The device enters sleep walking

   **Fix/Workaround**
   Do not run OSC0 in external clock mode if sleepwalking is expected to be used.

10. **Sleepwalking in idle and frozen sleep mode will mask all other PB clocks**
    If the CPU is in idle or frozen sleep mode and a module is in a state that triggers sleep walking, all PB clocks will be masked except the PB clock to the sleepwalking module.
    **Fix/Workaround**
    Mask all clock requests in the PM.PPCR register before going into idle or frozen mode.

11. **VERSION register reads 0x400**
    The VERSION register reads 0x400 instead of 0x411.
    **Fix/Workaround**
    None.

### 10.4.7 SCIF

1. **The DFLL should be slowed down before disabling it**
   The frequency of the DFLL should be set to minimum before disabling it.
   **Fix/Workaround**
   Before disabling the DFLL the value of the COARSE register should be zero.

2. **Writing to ICR masks new interrupts received in the same clock cycle**

Writing to ICR masks any new SCIF interrupt received in the same clock cycle, regardless of write value.

**Fix/Workaround**

For every interrupt except BODDET, SM33DET, and VREGOK the PCLKSR register can be read to detect new interrupts. BODDET, SM33DET and VREGOK interrupts will not be generated if they occur whilst writing to the ICR register.

3. **FINE value for DFLL is not correct when dithering is disabled**

   In open loop mode, the FINE value used by the DFLL DAC is offset by two compared to the value written to the DFLL0CONF.FINE field. The value used by the DFLL DAC is DFLL0CONF.FINE-0x002. If DFLL0CONF.FINE is written to 0x000, 0x001 or 0x002 the value used by the DFLL DAC will be 0x1FE, 0x1FF, or 0x000 respectively.

   **Fix/Workaround**

   Write the desired value added by two to the DFLL0CONF.FINE field.

4. **BODVERSION register reads 0x100**

   The BODVERSION register reads 0x100 instead of 0x101

   **Fix/Workaround**

   None.

5. **VREGCR.DEEPMODEDISABLE bit is not readable**

   VREGCR.DEEPMODEDISABLE bit is not readable.

   **Fix/Workaround**

   None.

6. **DFLL step size should be seven or lower when below 30MHz**

   If max step size is above seven, the DFLL might not lock at the correct frequency if the target frequency is below 30MHz.

   **Fix/Workaround**

   If the target frequency is below 30MHz, use a max step size (DFLL0MAXSTEP.MAXSTEP) of seven or lower.

7. **Generic clock sources are kept running in sleep modes**

   If a clock is used as a source for a generic clock when going to a sleep mode where clock sources are stopped, the source of the generic clock will be kept running. Please refer to Power Manager chapter for details about sleep modes.

   **Fix/Workaround**

   Disable generic clocks before going to sleep modes where clock sources are stopped to save power.

8. **DFLL clock is unstable with a fast reference clock**

   The DFLL clock can be unstable when a fast clock is used as a reference clock in closed loop mode.

   **Fix/Workaround**

   Use the 32KHz crystal oscillator clock, or a clock with a similar frequency, as DFLLIF reference clock.

9. **DFLLIF indicates coarse lock too early**

   The DFLLIF might indicate coarse lock too early, the DFLL will lose coarse lock and regain it later.

   **Fix/Workaround**

   Use max step size (DFLL0MAXSTEP.MAXSTEP) of 4 or higher.

10. **DFLLIF dithering does not work**

The DFLLIF dithering does not work.
**Fix/Workaround**
None.

11. **DFLLIF might lose fine lock when dithering is disabled**
When dithering is disabled and fine lock has been acquired, the DFLL might lose the fine lock resulting in up to 20% over-/undershoot.
**Fix/Workaround**
Solution 1: When the DFLL is used as main clock source, the target frequency of the DFLL should be 20% below the maximum operating frequency of the CPU. Don't use the DFLL as clock source for frequency sensitive applications.
Solution 2: Do not use the DFLL in closed loop mode.

12. **GCLK5 is non-functional**
GCLK5 is non-functional.
**Fix/Workaround**
None.

13. **BRIFA is non-functional**
BRIFA is non-functional.
**Fix/Workaround**
None.

14. **SCIF VERSION register reads 0x100**
SCIFVERSION register reads 0x100 instead of 0x102.
**Fix/Workaround**
None.

15. **BODVERSION register reads 0x100**
BODVERSION register reads 0x100 instead of 0x101.
**Fix/Workaround**
None.

16. **DFLLVERSION register reads 0x200**
DFLLVERSION register reads 0x200 instead of 0x201.
**Fix/Workaround**
None.

17. **RCCRVERSION register reads 0x100**
RCCRVERSION register reads 0x100 instead of 0x101.
**Fix/Workaround**
None.

18. **OSC32VERSION register reads 0x100**
OSC32VERSION register reads 0x100 instead of 0x101.
**Fix/Workaround**
None.

19. **VREGVERSION register reads 0x100**
VREGVERSION register reads 0x100 instead of 0x101.
**Fix/Workaround**
None.

20. **RC120MVERSION register reads 0x100**

RC120MVERSION register reads 0x100 instead of 0x101.
**Fix/Workaround**
None.

### 10.4.8 AST

1. **AST wake signal is released one AST clock cycle after the BUSY bit is cleared**
After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
**Fix/Workaround**
Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

### 10.4.9 WDT

1. **Clearing the WDT in window mode**
In window mode, if the WDT is cleared $2^{TBAN}$ CLK_WDT cycles after entering the window, the counter will be cleared, but will not exit the window. If this occurs, the SR.WINDOW bit will not be cleared after clearing the WDT.
**Fix/Workaround**
Check SR.WINDOW immediately after clearing the WDT. If set then clear the WDT once more.

2. **Clearing the Watchdog Timer (WDT) counter in second half of timeout period will issue a Watchdog reset**
If the WDT counter is cleared in the second half of the timeout period, the WDT will immediately issue a Watchdog reset.
**Fix/Workaround**
Use twice as long timeout period as needed and clear the WDT counter within the first half of the timeout period. If the WDT counter is cleared after the first half of the timeout period, you will get a Watchdog reset immediately. If the WDT counter is not cleared at all, the time before the reset will be twice as long as needed.

3. **VERSION register reads 0x400**
The VERSION register reads 0x400 instead of 0x402.
**Fix/Workaround**
None.

4. **WDT Control Register does not have synchronization feedback**
When writing to the Timeout Prescale Select (PSEL), Time Ban Prescale Select (TBAN), Enable (EN), or WDT Mode (MODE) fieldss of the WDT Control Register (CTRL), a synchronizer is started to propagate the values to the WDT clcok domain. This synchronization takes a finite amount of time, but only the status of the synchronization of the EN bit is reflected back to the user. Writing to the synchronized fields during synchronization can lead to undefined behavior.
**Fix/Workaround**
-When writing to the affected fields, the user must ensure a wait corresponding to 2 clock cycles of both the WDT peripheral bus clock and the selected WDT clock source.
-When doing writes that changes the EN bit, the EN bit can be read back until it reflects the written value.

### 10.4.10 FREQM

1. **Measured clock (CLK_MSR) sources 15-17 are shifted**
   CLKSEL = 14 selects the RC120M AW clock, CLKSEL = 15 selects the RC120M clock, and CLKSEL = 16 selects the RC32K clock as source for the measured clock (CLK_MSR).
   **Fix/Workaround**
   None.

2. **GCLK5 can not be used as source for the CLK_MSR**
   The frequency for GCLK5 can not be measured by the FREQM.
   **Fix/Workaround**
   None.

### 10.4.11 GPIO

1. **GPIO interrupt can not be cleared when interrupts are disabled**
   The GPIO interrupt can not be cleared unless the interrupt is enabled for the pin.
   **Fix/Workaround**
   Enable interrupt for the corresponding pin, then clear the interrupt.

2. **VERSION register reads 0x210**
   The VERSION register reads 0x210 instead of 0x211.
   **Fix/Workaround**
   None.

### 10.4.12 USART

1. **The RTS output does not function correctly in hardware handshaking mode**
   The RTS signal is not generated properly when the USART receives data in hardware handshaking mode. When the Peripheral DMA receive buffer becomes full, the RTS output should go high, but it will stay low.
   **Fix/Workaround**
   Do not use the hardware handshaking mode of the USART. If it is necessary to drive the RTS output high when the Peripheral DMA receive buffer becomes full, use the normal mode of the USART. Configure the Peripheral DMA Controller to signal an interrupt when the receive buffer is full. In the interrupt handler code, write a one to the RTSDIS bit in the USART Control Register (CR). This will drive the RTS output high. After the next DMA transfer is started and a receive buffer is available, write a one to the RTSEN bit in the USART CR so that RTS will be driven low.

### 10.4.13 SPI

1. **SPI data transfer hangs with CSR0.CSAAT==1 and MR.MODFDIS==0**
   When CSR0.CSAAT==1 and mode fault detection is enabled (MR.MODFDIS==0), the SPI module will not start a data transfer.
   **Fix/Workaround**
   Disable mode fault detection by writing a one to MR.MODFDIS.

2. **Disabling SPI has no effect on the SR.TDRE bit**
   Disabling SPI has no effect on the SR.TDRE bit whereas the write data command is filtered when SPI is disabled. Writing to TDR when SPI is disabled will not clear SR.TDRE. If SPI is disabled during a PDCA transfer, the PDCA will continue to write data to TDR until its buffer is empty, and this data will be lost.
   **Fix/Workaround**

Disable the PDCA, add two NOPs, and disable the SPI. To continue the transfer, enable the SPI and PDCA.

3. **SPI disable does not work in SLAVE mode**
   SPI disable does not work in SLAVE mode.
   **Fix/Workaround**
   Read the last received data, then perform a software reset by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

4. **SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0**
   When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
   **Fix/Workaround**
   When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

5. **SPI mode fault detection enable causes incorrect behavior**
   When mode fault detection is enabled (MR.MODFDIS==0), the SPI module may not operate properly.
   **Fix/Workaround**
   Always disable mode fault detection before using the SPI by writing a one to MR.MODFDIS.

### 10.4.14 TWI

1. **TWI pins are not SMBus compliant**
   The TWI pins draw current when they are supplied with 3.3V and the part is left unpowered.
   **Fix/Workaround**
   None.

2. **PA21, PB04, and PB05 are not 5V tolerant**
   Pins PA21, PB04, and PB05 are only 3.3V tolerant.
   **Fix/Workaround**
   None.

3. **PB04 SMBALERT function should not be used**
   The SMBALERT function from TWIMS0 should not be selected on pin PB04.
   **Fix/Workaround**
   None.

4. **TWIM STOP bit in IMR always reads as zero**
   The STOP bit in IMR always reads as zero.
   **Fix/Workaround**
   None.

5. **Disabled TWIM drives TWD and TWCK low**
   When the TWIM is disabled, it drives the TWD and TWCK signals with logic level zero. This can lead to communication problems with other devices on the TWI bus.
   **Fix/Workaround**
   Enable the TWIM first and then enable the TWD and TWCK peripheral pins in the GPIO controller. If it is necessary to disable the TWIM, first disable the TWD and TWCK peripheral pins in the GPIO controller and then disable the TWIM.

6. **TWIM SR.IDLE goes high immediately when NAK is received**
   When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
   **Fix/Workaround**
   If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

7. **TWIM TWALM polarity is wrong**
   The TWALM signal in the TWIM is active high instead of active low.
   **Fix/Workaround**
   Use an external inverter to invert the signal going into the TWIM. When using both TWIM and TWIS on the same pins, the TWALM cannot be used.

8. **TWIS CR.STREN does not work in deep sleep modes**
   When the device is in Stop, DeepStop, or Static mode, address reception will not wake device if both CR.SOAM and CR.STREN are one.
   **Fix/Workaround**
   Do not write both CR.STREN and CR.SOAM to one if the device needs to wake from deep sleep modes.

9. **TWI0.TWCK on PB05 is non-functional**
   TWI0.TWCK on PB05 is non-functional.
   **Fix/Workaround**
   Use TWI0.TWCK on other pins.

10. **TWIM Version Register reads zero**
    TWIM Version Register (VR) reads zero instead of 0x101
    **Fix/Workaround**
    None.

11. **TWIS Version Register reads zero**
    TWIS Version Register (VR) reads zero instead of 0x112
    **Fix/Workaround**
    None.

### 10.4.15 PWMA

1. **PARAMETER register reads 0x2424**
   The PARAMETER register reads 0x2424 instead of 0x24.
   **Fix/Workaround**
   None.

2. **Writing to the duty cycle registers when the timebase counter overflows can give an undefined result**
   The duty cycle registers will be corrupted if written when the timebase counter overflows. If the duty cycle registers are written exactly when the timebase counter overflows at TOP, the duty cycle registers may become corrupted.
   **Fix/Workaround**

Write to the duty cycle registers only directly after the Timebase Overflow bit in the status register is set.

3. **Open Drain mode does not work**
   The open drain mode does not work.
   **Fix/Workaround**
   None.

4. **BUSY bit is never cleared after writes to the Control Register (CR)**
   When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR when the PWMA is disabled (CR.EN == 0), the BUSY bit in the Status Register (SR.BUSY) will be set, but never cleared.
   **Fix/Workaround**
   When writing a non-zero value to CR.TOP, CR.SPREAD, or CR.TCLR, make sure the PWMA is enabled, or simultaneously enable the PWMA by writing a one to CR.EN.

5. **Incoming peripheral events are discarded during duty cycle register update**
   Incoming peripheral events to all applied channels will be discarded if a duty cycle update is received from the user interface in the same PWMA clock period.
   **Fix/Workaround**
   Ensure that duty cycle writes from the user interface are not performed in a PWMA period when an incoming peripheral event is expected.

6. **VERSION register reads 0x100**
   The VERSION register reads 0x100 instead of 0x101.
   **Fix/Workaround**
   None.

### 10.4.16 TC

1. **When the main clock is RCSYS, TIMER_CLOCK5 is equal to CLK_PBA**
   When the main clock is generated from RCSYS, TIMER_CLOCK5 is equal to CLK_PBA and not CLK_PBA/128.
   **Fix/Workaround**
   None.

### 10.4.17 ADCIFB

1. **Pendetect in sleep modes without CLK_ADCIFB will not wake the system**
   The pendetect will not wake the system from a sleep mode if the clock for the ADCIFB (CLK_ADCIFB) is turned off.
   **Fix/Workaround**
   Use a sleep mode where CLK_ADCIFB is not turned off to wake the part using pendetect.

2. **8-bit mode is not working**
   Do not use the ADCIFB 8-bit mode.
   **Fix/Workaround**
   Use the 10-bit mode and shift right by two bits.

3. **Using STARTUPTIME larger than 0x1F will freeze the ADC**
   Writing a value larger than 0x1F to the Startup Time field in the ADC Configuration Register (ACR.STARTUP) will freeze the ADC, and the Busy Status bit in the Status Register (SR.BUSY) will never be cleared.
   **Fix/Workaround**

Do not write values larger than 0x1F to ACR.STARTUP.

4. **ADC channels six to eight are non-functional**
   ADC channels six to eight are non-functional.
   **Fix/Workaround**
   None.

5. **VERSION register reads 0x100**
   The VERSION register reads 0x100 instead of 0x101.
   **Fix/Workaround**
   None.

### 10.4.18 ACIFB

1. **Generic clock sources in sleep modes.**
   The ACIFB should not use RC32K or CLK_1K as generic clock source if the chip uses sleep modes.
   **Fix/Workaround**
   None.

2. **Negative offset**
   The static offset of the analog comparator is approximately -50mV
   **Fix/Workaround**
   None.

3. **CONFW.WEVSRC and CONFW.WEVEN are not correctly described in the user interface**
   CONFW.WEVSRC is only two bits instead of three bits wide. Only values 0, 1, and 2 can be written to this register. CONFW.WEVEN is in bit position 10 instead of 11.
   **Fix/Workaround**
   Only write values 0, 1, and 2 to CONFW.WEVSRC. When reading CONFW.WEVSRC, disregard the third bit. Read/write bit 10 to access CONFW.WEVEN.

4. **VERSION register reads 0x200**
   The VERSION register reads 0x200 instead of 0x212.
   **Fix/Workaround**
   None.

### 10.4.19 CAT

1. **Switch off discharge current when reaching 0V**
   The discharge current will switch off when reaching MGCFG1.MAX, not when reaching 0V.
   **Fix/Workaround**
   None.

2. **CAT external capacitors are not clamped to ground when CAT is idle**
   The CAT module does not clamp the external capacitors to ground when it is idle. The capacitors are left floating, so they could accumulate small amounts of charge.
   **Fix/Workaround**
   None.

3. **DISHIFT field is stuck at zero**

The DISHIFT field in the MGCFG1, TGACFG1, TGBCFG1, and ATCFG1 registers is stuck at zero and cannot be written to a different value. Capacitor discharge time will only be determined by the DILEN field.
**Fix/Workaround**
None.

4. **MGCFG2.CONSEN field is stuck at zero**
The CONSEN field in the MGCFG2 register is stuck at zero and cannot be written to a different value. The CAT consensus filter does not function properly, so termination of QMatrix data acquisition is controlled only by the MAX field in MGCFG1.
**Fix/Workaround**
None.

5. **MGCFG2.ACCTRL bit is stuck at zero**
The ACCTRL bit in the MGCFG2 register is stuck at zero and cannot be written to one. The analog comparators will be constantly enabled.
**Fix/Workaround**
None.

6. **CAT asynchronous wake will be delayed by one AST event period**
If the CAT detects a condition the should asynchronously wake the device in static mode, the asynchronous wake will not occur until the next AST event. For example, if the AST is generating events to the CAT every 50ms, and the CAT detects a touch at t=9200ms, the asynchronous wake will occur at t=9250ms.
**Fix/Workaround**
None.

7. **VERSION register reads 0x100**
The VERSION register reads 0x100 instead of 0x200.
**Fix/Workaround**
None.

### 10.4.20 GLOC

1. **GLOC is non-functional**
Glue Logic Controller (GLOC) is non-functional.
**Fix/Workaround**
None.

### 10.4.21 aWire

1. **SAB multiaccess reads are not working**
Reading more than one word, halfword, or byte in one command is not working correctly.
**Fix/Workaround**
Split the access into several single word, halfword, or byte accesses.

2. **If a reset happens during the last SAB write, the aWire will stall**
If a reset happens during the last word, halfword or byte write the aWire will wait forever for an acknowledge from the SAB.
**Fix/Workaround**
Reset the aWire by keeping the RESET_N line low for 100ms.

3. **aWire enable does not work in Static mode**
aWire enable does not work in Static mode.

**Fix/Workaround**
None.

4. **aWire CPU clock speed robustness**
The aWire memory speed request command counter warps at clock speeds below approximately 5kHz.
**Fix/Workaround**
None.

5. **The aWire debug interface is reset after leaving Shutdown mode**
If the aWire debug mode is used as debug interface and the program enters Shutdown mode, the aWire interface will be reset when the part receives a wakeup either from the WAKE_N pin or the AST.
**Fix/Workaround**
None.

6. **aWire PB mapping and PB clock mask number**
The aWire PB has a different PB address and PB clock mask number.
**Fix/Workaround**
Use aWire PB address 0xFFFF6C00 and PB clock (PBAMASK) 24.

7. **VERSION register reads 0x200**
The VERSION register reads 0x200 instead of 0x210.
**Fix/Workaround**
None.

### 10.4.22 Chip

1. **WAKE_N pin can only wake up the chip from Shutdown mode**
It is not possible to wake up the chip from any other sleep mode than Shutdown using the WAKE_N pin. If the WAKE_N pin is asserted during a sleep mode other than Shutdown, nothing will happen.
**Fix/Workaround**
Use an EIC pin to wake up from sleep modes higher than Shutdown.

2. **Power consumption in static mode is too high**
Power consumption in static mode is too high when PA21 is high.
**Fix/Workaround**
Ensure PA21 is low.

3. **Shutdown mode is not functional**
Do not enter Shutdown mode.
**Fix/Workaround**
None.

4. **VDDIN current consumption increase above 1.8V**
When VDDIN increases above 1.8V, current on VDDIN increases with up to 40uA.
**Fix/Workaround**
None.

5. **Increased Power Consumption in VDDIO in sleep modes**
If OSC0 is enabled in crystal mode when entering a sleep mode where the OSC0 is disabled, this will lead to an increased power consumption in VDDIO.
**Fix/Workaround**

Solution 1: Disable OSC0 by writing a zero to the Oscillator Enable bit in the System Control Interface (SCIF) Oscillator Control Register (SCIF.OSC0CTRL.OSCEN) before going to a sleep mode where OSC0 is disabled.
Solution 2: Pull down or up XIN0 and XOUT0 with 1 MOhm resistor.

### 10.4.23 I/O Pins

1. **PB10 is not 3.3 V tolerant.**
   PB10 should be grounded on the PCB and left unused.
   **Fix/Workaround**
   None.

2. **Analog multiplexing consumes extra power**
   Current consumption on VDDIO increases when the voltage on analog inputs is close to VDDIO/2.
   **Fix/Workaround**
   None.

3. **PA02, PB01, PB04, PB05, and RESET_N have half of the pull-up strength**
   Pins PA02, PB01, PB04, PB05, and RESET_N have half of the specified pull-up strength.
   **Fix/Workaround**
   None.

4. **JTAG is enabled at power up**
   The JTAG function on pins PA00, PA01, PA02, and PA03, are enabled after startup. Normal I/O module functionality is not possible on these pins.
   **Fix/Workaround**
   Add a 10 kOhm pullup on the reset line.

5. **MCKO and MDO[3] are swapped in the AUX1 mapping**
   When using the OCD AUX1 mapping of trace signals MDO[3] is located on pin PB05 and MCKO is located on PB01.
   **Fix/Workaround**
   Swap pins PB01 and PB05 if using OCD AUX1.

# 11. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

## 11.1 Rev. H - 12/2011

1. Mechanical Characteristics: Updated the note related to the QFN48 Package Drawing.Updated thermal reistance data for TLLGA48 package. Updated package drawings for TQFP48 and QFN48 packages. Soldering profile updated (Time maintained above 217°C.)
2. Memories: Local bus address map corrected: The address offset for port 1 registers is 0x100, not 0x200.
3. SCIF DFLL: Removed “not” from “DFLLnSTEP.CSTEP and DFLLnSTEP.FSTEP should not be lower than 50% of the maximum value of DFLLnCONF.COARSE and DFLLnCONF.FINE”.
4. SCIF VREG POR descriptions updated. POR33 bits added in VREGCR.
5. SCIF VREG: Removed reference to flash fuses for CALIB field, user is recommended not writing to SELVDD field. Removed references to Electrical Characteristics.
6. SCIF: Fuses text removed from some submodules (SM33, VREG).
7. SCIF VREG: Flash recalibration is always done at POR.
8. SCIF SM33: Enabling SM33 will disable the POR33 detector.
9, Erratum regarding OSC32 disabling is not valid for RevB.
10. Flash Controller: Serial number address updated.
11. Block diagram and ADCIFB: Removed PRND signal from block diagram and ADCIFB block diagram.
12. USART: Added CTSIC bit description.
13. Power Manager: Updated Clock division and Clock ready flag sections.
14. ADCIFB: Added DMA section in Product Dependencies.
15. Electrcal Characteristics: Updated SPI timing data.
16. Electrical Characteristics, I/O Pin Characteristics: Added Input capacitance for TLLGA48 package.
17. Errata: Removed erratum regarding SPI RDR.PCS field, as the PCS field has been removed (refer to Section 11.7 on page 107).

## 11.2 Rev. G - 06/2011

1. FLASHCDW: FSR register is a Read-only register. Added info about QPRUP.
2. PM: Clarified POR33 masking requirements before shutdown. Added more info about wakeup sources. Added AWEN description. PPCR register reset value corrected.
3. SAU: SR.IDLE definition and reset value corrected.

4. DFLL: Open- and closed-loop operation clarified.
5. OSC32: Added note about OSC32RDY bit not always cleared when disabling OSC32.
6. USART: Major cleanup.

## 11.3 Rev. F- 11/2010

1. Features: Removed superfluous R mark.
2. Package and Pinout, GPIO function multiplexing:TWIMS0-TWCK on PA20 removed. ADCIFB-AD[3] on PA17 removed, number of ADC channels are 8, not 9. These were removed from rev. C, but reappeared in rev. E.

## 11.4 Rev. E- 10/2010

1. Overview: Added missing signals in block diagram.
2. Package and pinout: Added note about TWI, SMBUS and 5V tolerant pads in peripheral multiplexing. Added CAT DIS signal to signal descriptions. Removed TBD on ADVREFP minimum voltage.
3. Memories: Added SAU slave address to physical memory map.
4. Supply and startup considerations: VDDIN is using GND as ground pin. Clarified references to PORs in startup considerations.
5. FLASHCDW: Added serial number location to module configuration section.
6. PM: Added more info about the WAKE_N pin. Added info about CLK_PM, Updated the selection main clock source section.
7. SCIF: Major chapter update.
8. AST: Updated digital tuner formula and conditions.
9. GPIO: Updated GPER reset value and added more registers with non-zero reset value.
10. CAT: Added info about VDIVEN and discharge current formula.
11. ADCIFB: Fixed Sample and Hold time formula.
12. GLOC: Added info about pullup control and renamed LUTCR register to CR.
13. TC: Added features and version register.
14. SAU: Added OPEN bit to config register. Added description of unlock fields.
15. TWIS: SCR is Write-only. Improved explanation of slave transmitter mode. Updated data transfer diagrams.
16. Electrical Characteristics: Added more values. Added notes on simulated and characterized values. Added pin capacitance, rise, and fall times. Added timing characteristics. Removed all TBDs. Added ADC analog input characteristics. Symbol cleanup.
17. Errata: Updated errata list.

## 11.5 Rev. D - 06/2010

1. Ordering Information: Ordering code for TQFP ES changed from AT32UC3L064-AUES to AT32UC3L064-AUTES. TLLGA48 Tray option added.

## 11.6 Rev. C - 06/2010

1. Features and Description: Added QTouch library support.
2. USART: Description of unimplemented features removed.
3. Electrical Characteristics: Power Consumption numbers updated. Flash timing numbers added.

## 11.7 Rev. B - 05/2010

1. Package and Pinout: Added pinout figure for TLLGA48 package.
2. Package and Pinout, GPIO function multiplexing:TWIMS0-TWCK on PA20 removed. ADCIFB-AD[3] on PA17 removed, number of ADC channels are 8, not 9.
3. I/O Lines Considerations: Added: Following pins have high-drive capability: PA02, PA06, PA08, PA09, and PB01.
   Some TWI0 pins are SMBUS compliant (PA21, PB04, PB05).
4. HMATRIX Masters: PDCA is master 4, not master 3. SAU is master 3, not master 4.
5. SAU: IDLE bit added in the Status Register.
6. PDCA: Number of PDCA performance monitors is device dependent.
7. Peripheral Event System: Chapter updated.
8. PM: Bits in RCAUSE registers removed and renamed (JTAGHARD and AWIREHARD renamed to JTAG and AWIRE respectively, JTAG and AWIRE removed. BOD33 bit removed).
9. PM: RCAUSE.BOD33 bit removed. SM33 reset will be detected as a POR reset.
10. PM: WDT can be used as wake-up source if WDT is clocked from 32KHz oscillator.
11. PM: Entering Shutdown mode description updated.
12. SCIF: DFLL output frequency is 40-150MHz, not 20-150MHz or 30-150MHz.
13. SCIF: Temperature sensor is connected to ADC channel 9, not 7.
14. SCIF: Updated the oscillator connection figure for OSC0
15. GPIO: Removed unimplemented features (pull-down, buskeeper, drive strength, slew rate, Schmidt trigger, open drain).
16. SPI: RDR.PCS field removed (RDR[19:16]).
17. TWIS: Figures updated.
18. ADCIFB: The sample and hold time and the startup time formulas have been corrected (ADC Configuration Register).

19. ADCIFB: Updated ADC signal names.
20. ACIFB: CONFW.WEVSRC is bit 8-10, CONFW.EWEVEN is bit 11. CONF.EVENP and CONF.EVENN bits are swapped.
21. CAT: Matrix size is 16 by 8, not 18 by 8.
22. Electrical Characteristics: General update.
23. Mechanical Characteristics: Added numbers for package drawings.
24. Mechanical Characteristics: In the TQFP-48 package drawing the Lead Coplanarity is 0.102mm, not 0.080mm.
25. Ordering Information: Ordering code for TLLGA-48 package updated.

## 11.8 Rev. A – 06/2009

1. Initial revision.

# Table of Contents

***Atmel Corporation***
2325 Orchard Parkway
San Jose, CA 95131
USA
**Tel**: (+1)(408) 441-0311
**Fax**: (+1)(408) 487-2600
www.atmel.com

***Atmel Asia Limited***
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
**Tel**: (+852) 2245-6100
**Fax**: (+852) 2722-1369

***Atmel Munich GmbH***
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
**Tel**: (+49) 89-31970-0
**Fax**: (+49) 89-3194621

***Atmel Japan***
16F, Shin Osaki Kangyo Bldg.
1-6-4 Osaka Shinagawa-ku
Tokyo 104-0032
JAPAN
**Tel**: (+81) 3-6417-0300
**Fax**: (+81) 3-6417-0370