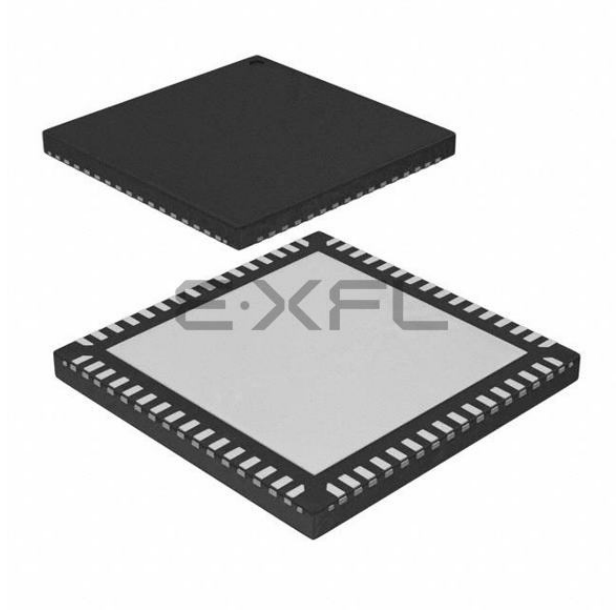


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | AVR |
| Core Size | 32-Bit Single-Core |
| Speed | 48MHz |
| Connectivity | I²C, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT |
| Number of I/O | 51 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 16K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.65V ~ 3.6V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-VFQFN Exposed Pad |
| Supplier Device Package | 64-QFN (9x9) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atuc128d3-z2ur |

## 2.2 Configuration Summary

**Table 2-1.** Configuration Summary

| Feature | ATUC128/64D3 | ATUC128/64D4 |
|---|---|---|
| Flash | 128/64 KB | 128/64KB |
| SRAM | 16 KB | 16 KB |
| Package | TQFP64, QFN64 | TQFP48, QFN48 |
| GPIO | 51 | 35 |
| FS USB Device | 1 | |
| Hi-drive pins | 4 | |
| External Interrupts | 9 | 7 |
| TWI Master/Slave | 1/1 | |
| USART | 3 | |
| Peripheral DMA Channels | 7 | |
| SPI | 1 | |
| Asynchronous Timers | 1 | |
| Timer/Counter Channels | 3 | |
| PWM channels | 7 | |
| Inter-IC Sound | 1 | |
| Frequency Meter | 1 | |
| Watchdog Timer | 1 | |
| Power Manager | 1 | |
| Oscillators | 2x Phase Locked Loop 80-240 MHz (PLL)<br>1x Crystal Oscillator 0.4-20 MHz (OSC0)<br>1x Crystal Oscillator 32 KHz (OSC32K)<br>1x RC Oscillator 120 MHz (RC120M)<br>1x RC Oscillator 115 kHz (RCSYS) | |
| 10-bit ADC channels | 8 | 6 |
| Capacitive Touch Sensor supported | 25 | 17 |
| Glue Logic Control Inputs/Outputs | 16/4 | 14/4 |
| JTAG | 1 | |
| aWire | 1 | |
| Max Frequency | 48 MHz | |

### 3.2.4 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 3-4.** Oscillator Pinout

| 48-pin Package | 64-pin Package | Pin | Oscillator Function |
|---|---|---|---|
| 30 | 39 | PA18 | XIN0 |
| 31 | 40 | PA19 | XOUT0 |
| 22 | 30 | PA11 | XIN32 |
| 23 | 31 | PA12 | XOUT32 |

### 3.2.5 Other Functions

The functions listed in Table 3-5 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.

**Table 3-5.** Other Functions

| 48-Pin Package | 64-Pin Package | Pin | Function |
|---|---|---|---|
| 47 | 63 | RESET_N | aWire DATA |
| 2 | 2 | PB12 | aWire DATAOUT |

# 4. Signal Descriptions

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

**Table 4-1.** Signal Descriptions List

| Signal Name | Function | Type | Active Level | Comments |
|---|---|---|---|---|
| **aWire - AW** | | | | |
| DATA | aWire data | I/O | | |
| DATAOUT | aWire data output for 2-pin mode | I/O | | |
| **External Interrupt Controller - EIC** | | | | |
| NMI | Non-Maskable Interrupt | Input | | |
| EXTINT8 - EXTINT1 | External interrupt | Input | | |
| **JTAG module - JTAG** | | | | |
| TCK | Test Clock | Input | | |
| TDI | Test Data In | Input | | |
| TDO | Test Data Out | Output | | |
| TMS | Test Mode Select | Input | | |
| **Power Manager - PM** | | | | |
| RESET_N | Reset | Input | Low | |
| **Basic Pulse Width Modulation Controller - PWMA** | | | | |
| PWMA6 - PWMA0 | PWMA channel waveforms | Output | | |
| **System Control Interface - SCIF** | | | | |
| GCLK2 - GCLK0 | Generic clock | Output | | |
| XIN0 | Oscillator 0 XIN Pin | Analog | | |
| XOUT0 | Oscillator 0 XOUT Pin | Analog | | |
| XIN32 | 32K Oscillator XIN Pin | Analog | | |
| XOUT32 | 32K Oscillator XOUT Pin | 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 - TC** | | | | |
| A0 | Channel 0 Line A | I/O | | |
| A1 | Channel 1 Line A | I/O | | |

**Table 4-1.** Signal Descriptions List

| | | | | |
|---|---|---|---|---|
| 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 Master- TWIM** | | | | |
| TWCK | Two-wire Serial Clock | | | |
| TWD | Two-wire Serial Data | | | |
| **Two Wire Interface Slave- TWIS** | | | | |
| TWCK | Two-wire Serial Clock | | | |
| TWD | Two-wire Serial Data | | | |
| **Universal Synchronous/Asynchronous Receiver/Transmitter - USART0/1/2** | | | | |
| CLK | Clock | I/O | | |
| CTS | Clear To Send | Input | Low | |
| RTS | Request To Send | Output | Low | |
| RXD | Receive Data | Input | | |
| TXD | Transmit Data | Output | | |
| **Universal Serial Bus 2.0 Full Speed Interface - USBC** | | | | |
| DM | DM for USB FS | | | |
| DP | DP for USB FS | | | |
| VBUS | VBUS | | | |
| **IIS Controller - IISC** | | | | |
| IBCK | IIS Serial Clock | I/O | | |
| ISDI | IIS Serial Data In | Input | | |
| ISDO | IIS Serial Data Out | Output | | |
| IWS | IIS Word Select | I/O | | |
| IMCK | IIS Master Clock | Output | | |
| **Capacitive Touch Sensor - CAT** | | | | |
| CSA24 - CSA0 | Capacitive Sensor Group A | I/O | | |
| CSB24 - CSB0 | Capacitive Sensor Group B | I/O | | |
| SYNC | Synchronize signal | Input | | |
| **Glue Logic Controller - GLOC** | | | | |
| IN15 - IN0 | Inputs to lookup tables | Input | | |
| OUT3 - OUT0 | Outputs from lookup tables | Output | | |
| **ADC controller interface - ADCIFD** | | | | |

**Table 4-1.** Signal Descriptions List

| | | | | |
|---|---|---|---|---|
| EXTTRIG | ADCIFD EXTTRIG | Input | | |
| AD7 - AD0 | ADC Inputs | Analog | | |
| **Power** | | | | |
| VDDIO | Digital I/O Power Supply | Power Input | | 3.0 V to 3.6V. |
| VDDANA | Analog Power Supply | Power Input | | 3.0 V to 3.6V |
| ADVREF | Analog Reference Voltage | Power Input | | 2.6 V to 3.6 V |
| VDDCORE | Core Power Supply | Power Input | | 1.65 V to 1.95 V |
| VDDIN | Voltage Regulator Input | Power Input | | 3.0 V to 3.6V |
| VDDOUT | Voltage Regulator Output | Power Output | | 1.65 V to 1.95V |
| GNDANA | Analog Ground | Ground | | |
| GND | Ground | Ground | | |
| **General Purpose I/O pin - GPIOA, GPIOB** | | | | |
| PA31 - PA00 | General Purpose I/O Controller GPIO A | I/O | | |
| PB18 - PB00 | General Purpose I/O Controller GPIO B | I/O | | |

## 4.1 I/O Line Considerations

### 4.1.1 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. TDO pin is an output, driven at VDDIO, and has no pull-up resistor. These JTAG pins can be used as GPIO pins and muxed with peripherals when the JTAG is disabled.

### 4.1.2 RESET_N Pin

The RESET_N pin is a schmitt input and integrates a programmable pull-up resistor to VDDIO. 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 the application.

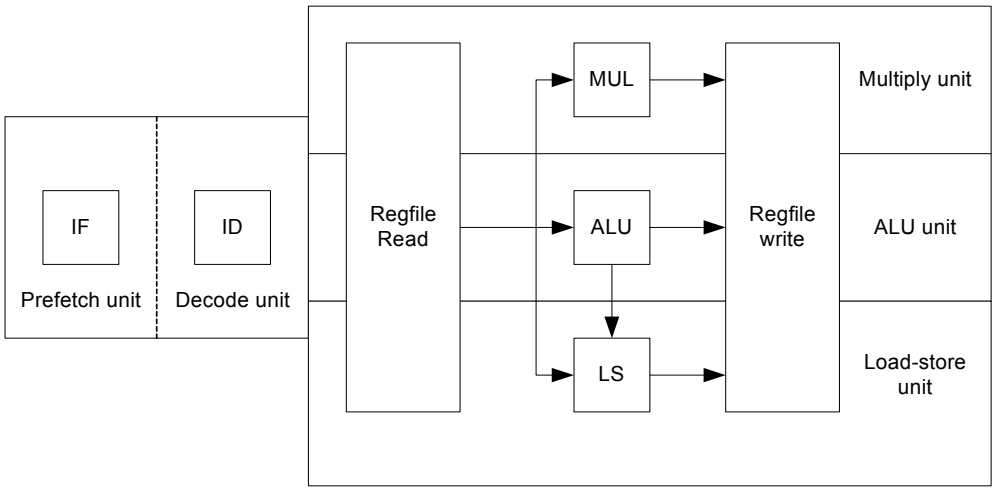
### 4.1.3 TWI Pins

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

### 4.1.4 GPIO Pins

All the I/O lines integrate a pull-up resistor. Programming of this pull-up resistor is performed

**Figure 5-2.** The AVR32UC Pipeline




### 5.3.2 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.

#### *5.3.2.1 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.

#### *5.3.2.2 Java Support*

AVR32UC does not provide Java hardware acceleration.

#### *5.3.2.3 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 5-1.** Instructions with Unaligned Reference Support

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

#### 5.3.2.4 *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

#### 5.3.2.5 *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.

**Table 5-3.** System Registers (Continued)

| Reg # | Address | Name | Function |
|---|---|---|---|
| 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 |
| 90-102 | 360-408 | Reserved | Reserved for future use |
| 103-111 | 412-444 | Reserved | Reserved for future use |
| 112-191 | 448-764 | Reserved | Reserved for future use |
| 192-255 | 768-1020 | IMPL | IMPLEMENTATION DEFINED |

## 5.5 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 5-4 on page 32. 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

contains information allowing the core to resume operation in the previous execution mode. This concludes the event handling.

### 5.5.3 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.

### 5.5.4 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.

### 5.5.5 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.

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.

If several events occur on the same instruction, they are handled in a prioritized way. The priority ordering is presented in Table 5-4 on page 32. 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.

**Table 5-4.** 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 | | | |
| 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 | | | |
| 14 | EVBA+0x18 | | | |
| 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 | | | |
| 25 | EVBA+0x70 | | | |
| 26 | EVBA+0x3C | | | |
| 27 | EVBA+0x40 | | | |
| 28 | EVBA+0x44 | | | |

## 8.7 Oscillator Characteristics

### 8.7.1 Oscillator 0 (OSC0) Characteristics

#### *8.7.1.1 Digital Clock Characteristics*

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

**Table 8-10.** 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 | % |

#### *8.7.1.2 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 8-2. 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:

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

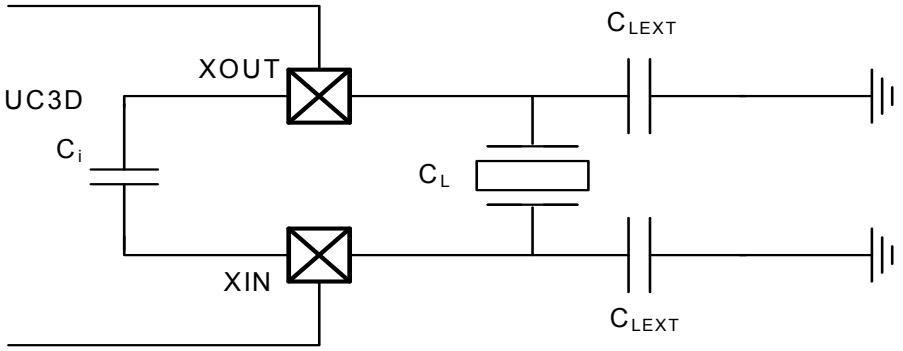
where $C_{PCB}$ is the capacitance of the PCB.

**Table 8-11.** Crystal Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $1/(t_{CPMAIN})$ | Crystal oscillator frequency | | 0.4 | | 20 | MHz |
| $C_L$ | Crystal load capacitance | | 6 | | 18 | pF |
| $C_i$ | Internal equivalent load capacitance | | | 1.7 | | pF |
| $t_{STARTUP}$ | Startup time | 400 KHz Resonator SCIF.OSCCTRL.GAIN = 0[1] | | 198 | | µs |
| | | 2 MHz Quartz SCIF.OSCCTRL.GAIN = 0[1] | | 4666 | | |
| | | 8 MHz Quartz SCIF.OSCCTRL.GAIN = 1[1] | | 975 | | |
| | | 12 MHz Quartz SCIF.OSCCTRL.GAIN = 2[1] | | 615 | | |
| | | 16 MHz Quartz SCIF.OSCCTRL.GAIN = 2[1] | | 1106 | | |
| | | 20 MHz Quartz SCIF.OSCCTRL.GAIN = 3[1] | | 1109 | | |

Notes: 1. Please refer to the SCIF chapter for details.

**Figure 8-2.** Oscillator Connection




### 8.7.2 32KHz Crystal Oscillator (OSC32K) Characteristics

#### *8.7.2.1 Digital Clock Characteristics*

The following table describes the characteristics for the oscillator when a digital clock is applied on XIN32.

**Table 8-12.** Digital Clock Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $f_{CPXIN}$ | XIN32 clock frequency | | | 32.768 | 5000 | KHz |
| $t_{CPXIN}$ | XIN32 clock duty cycle | | 40 | | 60 | % |

Figure 8-2 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 8-13.** 32 KHz Crystal Oscillator Characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $1/(t_{CP32KHz})$ | Crystal oscillator frequency | | | 32.768 | 5000 | KHz |
| $t_{ST}$ | Startup time | $R_S$ = 50kOhm, $C_L$ = 9pF | | 2 | | s |
| $C_L$ | Crystal load capacitance | | 6 | | 15 | pF |
| $C_i$ | Internal equivalent load capacitance | | | 1.4 | | pF |

### 8.9.4 Reset Sequence

**Table 8-29.** Electrical Characteristics

| Symbol | Parameter | Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $V_{DDRR}$ | VDDCORE rise rate to ensure power-on-reset | | 2.5 | | | V/ms |
| $V_{DDFR}$ | VDDCORE fall rate to ensure power-on-reset | | 0.01 | | 400 | V/ms |
| $V_{POR+}$ | Rising threshold voltage: voltage up to which device is kept under reset by POR on rising VDDCORE | Rising VDDCORE: $V_{RESTART}$ -> $V_{POR+}$ | 1.4 | 1.55 | 1.65 | V |
| $V_{POR-}$ | Falling threshold voltage: voltage when POR resets device on falling VDDCORE | Falling VDDCORE: 1.8V -> $V_{POR+}$ | 1.2 | 1.3 | 1.4 | V |
| $V_{RESTART}$ | On falling VDDCORE, voltage must go down to this value before supply can rise again to ensure reset signal is released at $V_{POR+}$ | Falling VDDCORE: 1.8V -> $V_{RESTART}$ | -0.1 | | 0.5 | V |
| $T_{POR}$ | Minimum time with VDDCORE < $V_{POR-}$ | Falling VDDCORE: 1.8V -> 1.1V | | 15 | | µs |
| $T_{RST}$ | Time for reset signal to be propagated to system | | | 200 | 400 | µs |
| $T_{SSU1}$ | Time for Cold System Startup: Time for CPU to fetch its first instruction (RCosc not calibrated) | | 480 | | 960 | µs |
| $T_{SSU2}$ | Time for Hot System Startup: Time for CPU to fetch its first instruction (RCosc calibrated) | | | 420 | | µs |

**Figure 8-3.** MCU Cold Start-Up RESET_N tied to VDDIN

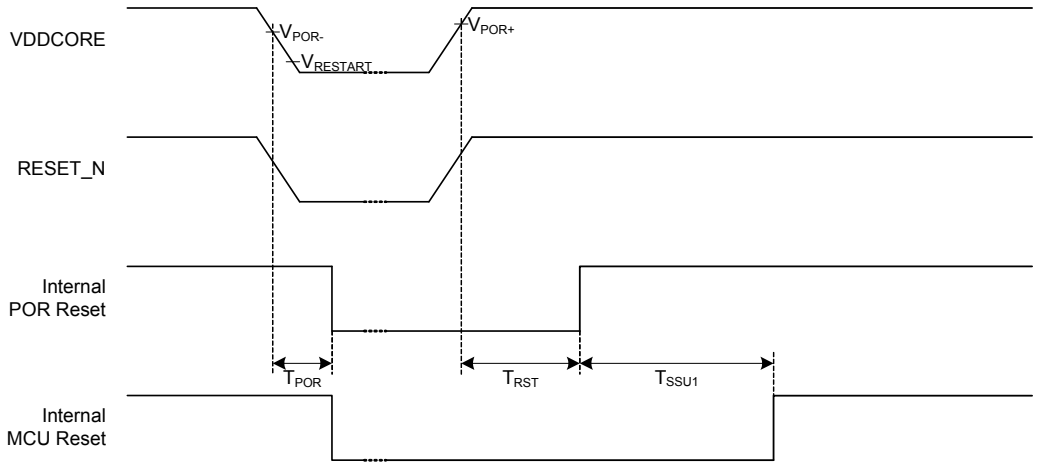


**Figure 8-4.** MCU Cold Start-Up RESET_N Externally Driven

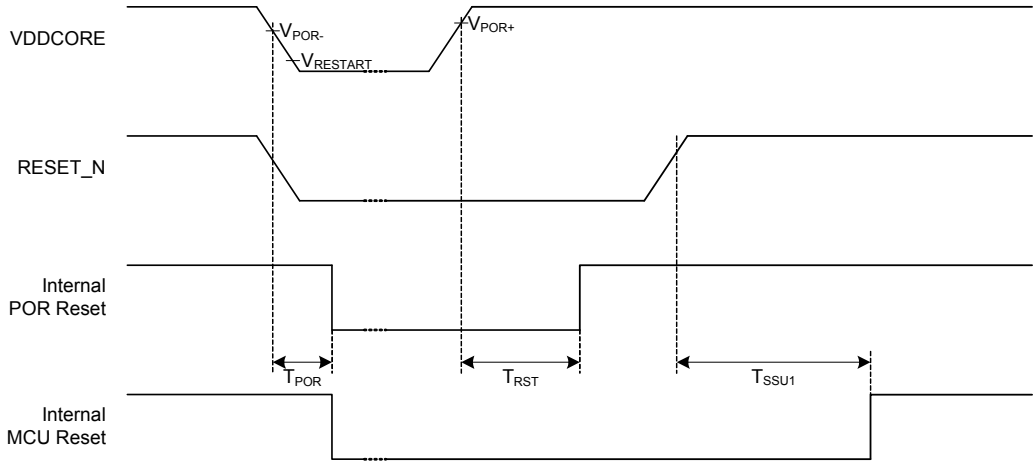
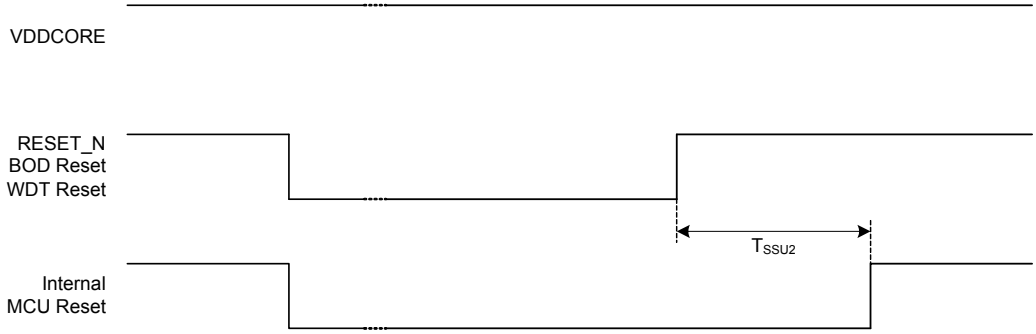



**Figure 8-5.** MCU Hot Start-Up




In dual supply configuration, the power up sequence must be carefully managed to ensure a safe startup of the device in all conditions.

The power up sequence must ensure that the internal logic is safely powered when the internal reset (Power On Reset) is released and that the internal Flash logic is safely powered when the CPU fetch the first instructions.

Therefore VDDCORE rise rate (VDDRR) must be equal or superior to 2.5V/ms and VDDIO must reach VDDIO mini value before 500 us (< TRST + TSSU1) after VDDCORE has reached $V_{POR+}$ min value.

## 9.3 Soldering Profile

Table 9-14 gives the recommended soldering profile from J-STD-20.

**Table 9-14.** 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°C min, 200°C max |
| Temperature Maintained Above 217°C | 60-150 s |
| Time within 5·C of Actual Peak Temperature | 30 s |
| 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.

# 10. Ordering Information

**Table 10-1.** Ordering Information

| Device | Ordering Code | Carrier Type | Package | Package Type | Temperature Operating Range |
|---|---|---|---|---|---|
| ATUC128D3 | ATUC128D3-A2UT | Tray | TQFP 64 | JESD97 Classification E3 | Industrial (-40·C to 85·C) |
| | ATUC128D3-A2UR | Tape & Reel | TQFP 64 | | |
| | ATUC128D3-Z2UT | Tray | QFN 64 | | |
| | ATUC128D3-Z2UR | Tape & Reel | QFN 64 | | |
| ATUC128D4 | ATUC128D4-AUT | Tray | TQFP 48 | | |
| | ATUC128D4-AUR | Tape & Reel | TQFP 48 | | |
| | ATUC128D4-Z1UT | Tray | QFN 48 | | |
| | ATUC128D4-Z1UR | Tape & Reel | QFN 48 | | |
| ATUC64D3 | ATUC64D3-A2UT | Tray | TQFP 64 | JESD97 Classification E3 | Industrial (-40·C to 85·C) |
| | ATUC64D3-A2UR | Tape & Reel | TQFP 64 | | |
| | ATUC64D3-Z2UT | Tray | QFN 64 | | |
| | ATUC64D3-Z2UR | Tape & Reel | QFN 64 | | |
| ATUC64D4 | ATUC64D4-AUT | Tray | TQFP 48 | | |
| | ATUC64D4-AUR | Tape & Reel | TQFP 48 | | |
| | ATUC64D4-Z1UT | Tray | QFN 48 | | |
| | ATUC64D4-Z1UR | Tape & Reel | QFN 48 | | |

# 11. Errata

## 11.1 Rev. C

### 11.1.1 SPI

1. **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).

2. **PCS field in receive data register is inaccurate**
   The PCS field in the SPI_RDR register does not accurately indicate from which slave the received data is read.
   **Fix/Workaround**
   None.

3. **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.

4. **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.

5. **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.

6. **Timer Counter**

7. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   **Fix/Workaround**
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

4. **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.

5. **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.

6. **Timer Counter**

7. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   **Fix/Workaround**
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

### 11.2.3 TWIS

1. **Clearing the NAK bit before the BTF bit is set locks up the TWI bus**
   When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
   **Fix/Workaround**
   Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

### 11.2.4 PWMA

1. **The SR.READY bit cannot be cleared by writing to SCR.READY**
   The Ready bit in the Status Register will not be cleared when writing a one to the corresponding bit in the Status Clear register. The Ready bit will be cleared when the Busy bit is set.
   **Fix/Workaround**
   Disable the Ready interrupt in the interrupt handler when receiving the interrupt. When an operation that triggers the Busy/Ready bit is started, wait until the ready bit is low in the Status Register before enabling the interrupt.

### 11.3.3 Timer Counter

1. **Channel chaining skips first pulse for upper channel**
   When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
   **Fix/Workaround**
   Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

### 11.3.4 TWIS

1. **TWIS stretch on Address match error**
   When the TWIS stretches TWCK due to a slave address match, it also holds TWD low for the same duration if it is to be receiving data. When TWIS releases TWCK, it releases TWD at the same time. This can cause a TWI timing violation.
   **Fix/Workaround**
   None.

2. **Clearing the NAK bit before the BTF bit is set locks up the TWI bus**
   When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
   **Fix/Workaround**
   Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

3. **CAT**

4. **CAT module does not terminate QTouch burst on detect**
   The CAT module does not terminate a QTouch burst when the detection voltage is reached on the sense capacitor. This can cause the sense capacitor to be charged more than necessary. Depending on the dielectric absorption characteristics of the capacitor, this can lead to unstable measurements.
   **Fix/Workaround**
   Use the minimum possible value for the MAX field in the ATCFG1, TG0CFG1, and TG1CFG1 registers.

### 11.3.5 PWMA

1. **The SR.READY bit cannot be cleared by writing to SCR.READY**
   The Ready bit in the Status Register will not be cleared when writing a one to the corresponding bit in the Status Clear register. The Ready bit will be cleared when the Busy bit is set.
   **Fix/Workaround**
   Disable the Ready interrupt in the interrupt handler when receiving the interrupt. When an operation that triggers the Busy/Ready bit is started, wait until the ready bit is low in the Status Register before enabling the interrupt.

2.