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	60MHz
Connectivity	I²C, IrDA, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	44
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K 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-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3b0128-a2ut

1. Description

The AT32UC3B is a complete System-On-Chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 60 MHz. 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.

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

The AT32UC3B incorporates on-chip Flash and SRAM memories for secure and fast access.

The Peripheral Direct Memory Access controller enables data transfers between peripherals and memories without processor involvement. PDCA drastically reduces processing overhead when transferring continuous and large data streams between modules within the MCU.

The Power Manager improves design flexibility and security: the on-chip Brown-Out Detector monitors the power supply, the CPU runs from the on-chip RC oscillator or from one of external oscillator sources, a Real-Time Clock and its associated timer keeps track of the time.

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

The PWM modules provides seven independent channels with many configuration options including polarity, edge alignment and waveform non overlap control. One PWM channel can trigger ADC conversions for more accurate close loop control implementations.

The AT32UC3B also features many communication interfaces for communication intensive applications. In addition to standard serial interfaces like USART, SPI or TWI, other interfaces like flexible Synchronous Serial Controller and USB are available. The USART supports different communication modes, like SPI mode.

The Synchronous Serial Controller provides easy access to serial communication protocols and audio standards like I^2S, UART or SPI.

The Full-Speed USB 2.0 Device interface supports several USB Classes at the same time thanks to the rich End-Point configuration. The Embedded Host interface allows device like a USB Flash disk or a USB printer to be directly connected to the processor.

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 included fully debounced reporting of touch keys and includes 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.

AT32UC3B integrates a class 2+ Nexus 2.0 On-Chip Debug (OCD) System, with non-intrusive real-time trace, full-speed read/write memory access in addition to basic runtime control. The Nanotrace interface enables trace feature for JTAG-based debuggers.

3. Configuration Summary

The table below lists all AT32UC3B memory and package configurations:

Table 3-1. Configuration Summary

Feature	AT32UC3B0512	AT32UC3B0256/128/64	AT32UC3B1512	AT32UC3B1256/128/64
Flash	512 KB	256/128/64 KB	512 KB	256/128/64 KB
SRAM	96KB	32/32/16KB	96KB	32/16/16KB
GPIO	44		28	
External Interrupts	8		6	
TWI	1			
USART	3			
Peripheral DMA Channels	7			
SPI	1			
Full Speed USB	Mini-Host + Device		Device	
SSC	1		0	
Audio Bitstream DAC	1	0	1	0
Timer/Counter Channels	3			
PWM Channels	7			
Watchdog Timer	1			
Real-Time Clock Timer	1			
Power Manager	1			
Oscillators	PLL 80-240 MHz (PLL0/PLL1) Crystal Oscillators 0.4-20 MHz (OSC0) Crystal Oscillator 32 KHz (OSC32K) RC Oscillator 115 kHz (RCSYS)			
	Crystal Oscillators 0.4-20 MHz (OSC1)			
10-bit ADC number of channels	8		6	
JTAG	1			
Max Frequency	60 MHz			
Package	TQFP64, QFN64		TQFP48, QFN48	

Table 5-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
Serial Peripheral Interface - SPI0				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
NPCS0 - NPCS3	SPI Peripheral Chip Select	I/O	Low	
SCK	Clock	Output		
Synchronous Serial Controller - SSC				
RX_CLOCK	SSC Receive Clock	I/O		
RX_DATA	SSC Receive Data	Input		
RX_FRAME_SYNC	SSC Receive Frame Sync	I/O		
TX_CLOCK	SSC Transmit Clock	I/O		
TX_DATA	SSC Transmit Data	Output		
TX_FRAME_SYNC	SSC Transmit Frame Sync	I/O		
Timer/Counter - TIMER				
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 - TWI				
SCL	Serial Clock	I/O		
SDA	Serial Data	I/O		
Universal Synchronous Asynchronous Receiver Transmitter - USART0, USART1, USART2				
CLK	Clock	I/O		
CTS	Clear To Send	Input		

5.2 RESET_N pin

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

5.3 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.

5.4 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 Controllers. After reset, I/O lines default as inputs with pull-up resistors disabled, except when indicated otherwise in the column “Reset Value” of the GPIO Controller user interface table.

5.5 High drive pins

The four pins PA20, PA21, PA22, PA23 have high drive output capabilities.

5.6 Power Considerations

5.6.1 Power Supplies

The AT32UC3B has several types of power supply pins:

- **VDDIO: Powers I/O lines. Voltage is 3.3V nominal.**
- **VDDANA: Powers the ADC Voltage is 3.3V nominal.**
- **VDDIN: Input voltage for the voltage regulator. Voltage is 3.3V nominal.**
- **VDDCORE: Powers the core, memories, and peripherals. Voltage is 1.8V nominal.**
- **VDDPLL: Powers the PLL. Voltage is 1.8V nominal.**

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

Refer to Electrical Characteristics section for power consumption on the various supply pins.

The main requirement for power supplies connection is to respect a star topology for all electrical connection.

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.

6.3 The AVR32UC CPU

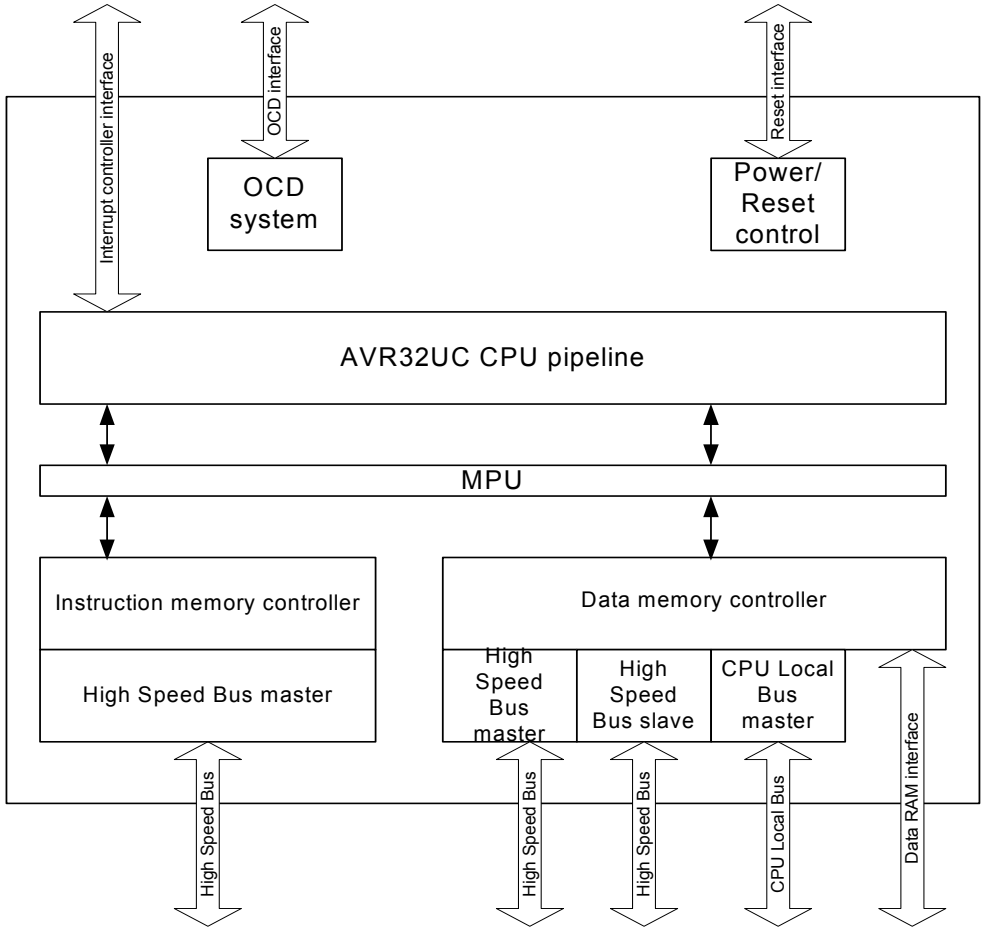
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced OCD system, no caches, and a Memory Protection Unit (MPU). Java acceleration hardware is not implemented.

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 fast GPIO ports. This local bus has to be enabled by writing 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 Memories chapter of this data sheet.

Figure 6-1 on page 19 displays the contents of AVR32UC.

Figure 6-1. Overview of the AVR32UC CPU



6.3.1 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 6-2 on page 20 shows an overview of the AVR32UC pipeline stages.

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.

6.5.2 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 6-4, 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.

6.5.3 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.

6.5.4 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

9. Electrical Characteristics

9.1 Absolute Maximum Ratings*

Operating Temperature	-40°C to +85°C
Storage Temperature	-60°C to +150°C
Voltage on GPIO Pins with respect to Ground for TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	-0.3 to 3.6V
Voltage on GPIO Pins with respect to Ground except for TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	-0.3 to 5.5V
Maximum Operating Voltage (VDDCORE, VDDPLL)	1.95V
Maximum Operating Voltage (VDDIO,VDDIN,VDDANA)	3.6V
Total DC Output Current on all I/O Pin	
for 48-pin package	200 mA
for 64-pin package	265 mA

*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.

9.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are certified for a junction temperature up to T_J = 100°C.

Table 9-1. DC Characteristics

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Unit
$V_{VDDCORE}$	DC Supply Core			1.65		1.95	V
V_{VDDPLL}	DC Supply PLL			1.65		1.95	V
V_{VDDIO}	DC Supply Peripheral I/Os			3.0		3.6	V
V_{IL}	Input Low-level Voltage			-0.3		+0.8	V
V_{IH}	Input High-level Voltage	AT32UC3B064 AT32UC3B0128 AT32UC3B0256 AT32UC3B164 AT32UC3B1128 AT32UC3B1256	All I/O pins except TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		5.5	V
			TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		3.6	V
		AT32UC3B0512 AT32UC3B1512	All I/O pins except TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		5.5	V
			TCK, RESET_N	2.5		3.6	V
			PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		3.6	V
V_{OL}	Output Low-level Voltage	I_{OL}= -4mA for all I/O except PA20, PA21, PA22, PA23				0.4	V
		I_{OL}= -8mA for PA20, PA21, PA22, PA23				0.4	V
V_{OH}	Output High-level Voltage	I_{OL}= -4mA for all I/O except PA20, PA21, PA22, PA23		V_{VDDIO} -0.4			V
		I_{OL}= -8mA for PA20, PA21, PA22, PA23		V_{VDDIO} -0.4			V
I_{OL}	Output Low-level Current	All I/O pins except PA20, PA21, PA22, PA23				-4	mA
		PA20, PA21, PA22, PA23				-8	mA
I_{OH}	Output High-level Current	All I/O pins except for PA20, PA21, PA22, PA23				4	mA
		PA20, PA21, PA22, PA23				8	mA
I_{LEAK}	Input Leakage Current	Pullup resistors disabled				1	µA

9.5.1 Power Consumtion for Different Sleep Modes

Table 9-10. Power Consumption for Different Sleep Modes for AT32UC3B064, AT32UC3B0128, AT32UC3B0256, AT32UC3B164, AT32UC3B1128, AT32UC3B1256

Mode	Conditions		Typ.	Unit
Active	- CPU running a recursive Fibonacci Algorithm from flash and clocked from PLL0 at f MHz. - Voltage regulator is on. - XIN0: external clock. Xin1 Stopped. XIN32 stopped. - All peripheral clocks activated with a division by 8. - GPIOs are inactive with internal pull-up, JTAG unconnected with external pull-up and Input pins are connected to GND		0.3xf(MHz)+0.443	mA/MHz
	Same conditions at 60 MHz		18.5	mA
Idle	See Active mode conditions		0.117xf(MHz)+0.28	mA/MHz
	Same conditions at 60 MHz		7.3	mA
Frozen	See Active mode conditions		0.058xf(MHz)+0.115	mA/MHz
	Same conditions at 60 MHz		3.6	mA
Standby	See Active mode conditions		0.042xf(MHz)+0.115	mA/MHz
	Same conditions at 60 MHz		2.7	mA
Stop	- CPU running in sleep mode - XIN0, Xin1 and XIN32 are stopped. - All peripheral clocks are desactived. - GPIOs are inactive with internal pull-up, JTAG unconnected with external pull-up and Input pins are connected to GND.		37.8	µA
Deepstop	See Stop mode conditions		24.9	µA
Static	See Stop mode conditions	Voltage Regulator On	13.9	µA
		Voltage Regulator Off	8.9	µA

Notes: 1. Core frequency is generated from XIN0 using the PLL so that 140 MHz < f_{PLL0} < 160 MHz and 10 MHz < f_{XIN0} < 12 MHz.

Table 9-11. Power Consumption for Different Sleep Modes for AT32UC3B0512, AT32UC3B1512

Mode	Conditions	Typ.	Unit
Active	- CPU running a recursive Fibonacci Algorithm from flash and clocked from PLL0 at f MHz. - Voltage regulator is on. - XIN0: external clock. Xin1 Stopped. XIN32 stopped. - All peripheral clocks activated with a division by 8. - GPIOs are inactive with internal pull-up, JTAG unconnected with external pull-up and Input pins are connected to GND	0.359xf(MHz)+1.53	mA/MHz
	Same conditions at 60 MHz	24	mA
Idle	See Active mode conditions	0.146xf(MHz)+0.291	mA/MHz
	Same conditions at 60 MHz	9	mA

9.12 Flash Memory Characteristics

The following table gives the device maximum operating frequency depending on the field FWS of the Flash FSR register. This field defines the number of wait states required to access the Flash Memory. Flash operating frequency equals the CPU/HSB frequency.

Table 9-28. Flash Operating Frequency

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
F_{FOP}	Flash Operating Frequency	FWS = 0			33	MHz
		FWS = 1			60	MHz

Table 9-29. Programming TIme

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
T_{FPP}	Page Programming Time			4		ms
T_{FFP}	Fuse Programming Time			0.5		ms
T_{FCE}	Chip Erase Time			4		ms

Table 9-30. Flash Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
N_{FARRAY}	Flash Array Write/Erase cycle				100K	Cycle
N_{FFUSE}	General Purpose Fuses write cycle				1000	Cycle
T_{FDR}	Flash Data Retention Time			15		Year

11. Ordering Information

Device	Ordering Code	Package	Conditioning	Temperature Operating Range
AT32UC3B0512	AT32UC3B0512-A2UES	TQFP 64	-	Industrial (-40°C to 85°C)
	AT32UC3B0512-A2UR	TQFP 64	Reel	Industrial (-40°C to 85°C)
	AT32UC3B0512-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0512-Z2UES	QFN 64	-	Industrial (-40°C to 85°C)
	AT32UC3B0512-Z2UR	QFN 64	Reel	Industrial (-40°C to 85°C)
	AT32UC3B0512-Z2UT	QFN 64	Tray	Industrial (-40°C to 85°C)
AT32UC3B0256	AT32UC3B0256-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0256-A2UR	TQFP 64	Reel	Industrial (-40°C to 85°C)
	AT32UC3B0256-Z2UT	QFN 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0256-Z2UR	QFN 64	Reel	Industrial (-40°C to 85°C)
AT32UC3B0128	AT32UC3B0128-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0128-A2UR	TQFP 64	Reel	Industrial (-40°C to 85°C)
	AT32UC3B0128-Z2UT	QFN 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0128-Z2UR	QFN 64	Reel	Industrial (-40°C to 85°C)
AT32UC3B064	AT32UC3B064-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B064-A2UR	TQFP 64	Reel	Industrial (-40°C to 85°C)
	AT32UC3B064-Z2UT	QFN 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B064-Z2UR	QFN 64	Reel	Industrial (-40°C to 85°C)
AT32UC3B1512	AT32UC3B1512-Z1UT	QFN 48	-	Industrial (-40°C to 85°C)
	AT32UC3B1512-Z1UR	QFN 48	-	Industrial (-40°C to 85°C)
AT32UC3B1256	AT32UC3B1256-AUT	TQFP 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B1256-AUR	TQFP 48	Reel	Industrial (-40°C to 85°C)
	AT32UC3B1256-Z1UT	QFN 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B1256-Z1UR	QFN 48	Reel	Industrial (-40°C to 85°C)
AT32UC3B1128	AT32UC3B1128-AUT	TQFP 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B1128-AUR	TQFP 48	Reel	Industrial (-40°C to 85°C)
	AT32UC3B1128-Z1UT	QFN 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B1128-Z1UR	QFN 48	Reel	Industrial (-40°C to 85°C)
AT32UC3B164	AT32UC3B164-AUT	TQFP 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B164-AUR	TQFP 48	Reel	Industrial (-40°C to 85°C)
	AT32UC3B164-Z1UT	QFN 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B164-Z1UR	QFN 48	Reel	Industrial (-40°C to 85°C)

7. **TC**

8. **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.

- *Processor and Architecture*

1. **LDM instruction with PC in the register list and without ++ increments Rp**
   For LDM with PC in the register list: the instruction behaves as if the ++ field is always set, ie the pointer is always updated. This happens even if the ++ field is cleared. Specifically, the increment of the pointer is done in parallel with the testing of R12.
   Fix/Workaround
   None.

2. **RETE instruction does not clear SREG[L] from interrupts**
   The RETE instruction clears SREG[L] as expected from exceptions.
   Fix/Workaround
   When using the STCOND instruction, clear SREG[L] in the stacked value of SR before returning from interrupts with RETE.

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.

4. **USART**

5. **ISO7816 info register US_NER cannot be read**
   The NER register always returns zero.
   Fix/Workaround
   None.

6. **ISO7816 Mode T1: RX impossible after any TX**
   RX impossible after any TX.
   Fix/Workaround
   SOFT_RESET on RX+ Config US_MR + Config_US_CR.

7. **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.

8. **Corruption after receiving too many bits in SPI slave mode**
   If the USART is in SPI slave mode and receives too much data bits (ex: 9bitsinstead of 8 bits) by the SPI master, an error occurs. After that, the next reception may be corrupted even if the frame is correct and the USART has been disabled, reset by a soft reset and re-enabled.
   Fix/Workaround
   None.

9. **USART slave synchronous mode external clock must be at least 9 times lower in frequency than CLK_USART**
   When the USART is operating in slave synchronous mode with an external clock, the frequency of the signal provided on CLK must be at least 9 times lower than CLK_USART.
   Fix/Workaround
   When the USART is operating in slave synchronous mode with an external clock, provide a signal on CLK that has a frequency at least 9 times lower than CLK_USART.

10. **HMATRIX**

11. **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.

- *DSP Operations*

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.

16. **Increased Power Consumption in VDDIO in sleep modes**
    If the 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
    Disable the OSC0 through the Power Manager (PM) before going to any sleep mode where the OSC0 is disabled, or pull down or up XIN0 and XOUT0 with 1Mohm resistor.

17. **SSC**

18. **Additional delay on TD output**
    A delay from 2 to 3 system clock cycles is added to TD output when:
    TCMR.START = Receive Start,
    TCMR.STTDLY = more than ZERO,
    RCMR.START = Start on falling edge / Start on Rising edge / Start on any edge,
    RFMR.FSOS = None (input).
    Fix/Workaround
    None.

19. **TF output is not correct**
    TF output is not correct (at least emitted one serial clock cycle later than expected) when:
    TFMR.FSOS = Driven Low during data transfer/ Driven High during data transfer
    TCMR.START = Receive start
    RFMR.FSOS = None (Input)
    RCMR.START = any on RF (edge/level)
    Fix/Workaround
    None.

20. **Frame Synchro and Frame Synchro Data are delayed by one clock cycle**
    The frame synchro and the frame synchro data are delayed from 1 SSC_CLOCK when:
    - Clock is CKDIV
    - The START is selected on either a frame synchro edge or a level
    - Frame synchro data is enabled
    - Transmit clock is gated on output (through CKO field)
    Fix/Workaround
    Transmit or receive CLOCK must not be gated (by the mean of CKO field) when START condition is performed on a generated frame synchro.

21. **USB**

22. **UPCFGn.INTFRQ is irrelevant for isochronous pipe**
    As a consequence, isochronous IN and OUT tokens are sent every 1ms (Full Speed), or every 125uS (High Speed).
    Fix/Workaround
    For higher polling time, the software must freeze the pipe for the desired period in order to prevent any "extra" token.

- *ADC*

1. **Sleep Mode activation needs additional A to D conversion**
    If the ADC sleep mode is activated when the ADC is idle the ADC will not enter sleep mode before after the next AD conversion.
    Fix/Workaround
    Activate the sleep mode in the mode register and then perform an AD conversion.

12.2 AT32UC3B0256, AT32UC3B0128, AT32UC3B064, AT32UC3B1256, AT32UC3B1128, AT32UC3B164

All industrial parts labelled with -UES (for engineering samples) are revision B parts.

12.2.1 Rev I, J, K

- *PWM*

1. **PWM channel interrupt enabling triggers an interrupt**
   When enabling a PWM channel that is configured with center aligned period (CALG=1), an interrupt is signalled.
   Fix/Workaround
   When using center aligned mode, enable the channel and read the status before channel interrupt is enabled.

2. **PWN counter restarts at 0x0001**
   The PWM counter restarts at 0x0001 and not 0x0000 as specified. Because of this the first PWM period has one more clock cycle.
   Fix/Workaround
   - The first period is 0x0000, 0x0001, ..., period.
   - Consecutive periods are 0x0001, 0x0002, ..., period.

3. **PWM update period to a 0 value does not work**
   It is impossible to update a period equal to 0 by the using the PWM update register (PWM_CUPD).
   Fix/Workaround
   Do not update the PWM_CUPD register with a value equal to 0.

4. **SPI**

5. **SPI Slave / PDCA transfer: no TX UNDERRUN flag**
   There is no TX UNDERRUN flag available, therefore in SPI slave mode, there is no way to be informed of a character lost in transmission.
   Fix/Workaround
   For PDCA transfer: none.

6. **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.

7. **SPI Glitch on RXREADY flag in slave mode when enabling the SPI or during the first transfer**
   In slave mode, the SPI can generate a false RXREADY signal during enabling of the SPI or during the first transfer.
   Fix/Workaround
   1. Set slave mode, set required CPOL/CPHA.
   2. Enable SPI.

14. **SSC**

15. **Additional delay on TD output**
    A delay from 2 to 3 system clock cycles is added to TD output when:
    TCMR.START = Receive Start,
    TCMR.STTDLY = more than ZERO,
    RCMR.START = Start on falling edge / Start on Rising edge / Start on any edge,
    RFMR.FSOS = None (input).
    Fix/Workaround
    None.

16. **TF output is not correct**
    TF output is not correct (at least emitted one serial clock cycle later than expected) when:
    TFMR.FSOS = Driven Low during data transfer/ Driven High during data transfer
    TCMR.START = Receive start
    RFMR.FSOS = None (Input)
    RCMR.START = any on RF (edge/level)
    Fix/Workaround
    None.

17. **Frame Synchro and Frame Synchro Data are delayed by one clock cycle**
    The frame synchro and the frame synchro data are delayed from 1 SSC_CLOCK when:
    - Clock is CKDIV
    - The START is selected on either a frame synchro edge or a level
    - Frame synchro data is enabled
    - Transmit clock is gated on output (through CKO field)
    Fix/Workaround
    Transmit or receive CLOCK must not be gated (by the mean of CKO field) when START condition is performed on a generated frame synchro.

18. **USB**

19. **UPCFGn.INTFRQ is irrelevant for isochronous pipe**
    As a consequence, isochronous IN and OUT tokens are sent every 1ms (Full Speed), or every 125uS (High Speed).
    Fix/Workaround
    For higher polling time, the software must freeze the pipe for the desired period in order to prevent any "extra" token.

- *ADC*

1. **Sleep Mode activation needs additional A to D conversion**
   If the ADC sleep mode is activated when the ADC is idle the ADC will not enter sleep mode before after the next AD conversion.
   Fix/Workaround
   Activate the sleep mode in the mode register and then perform an AD conversion.

- *PDCA*

1. **Wrong PDCA behavior when using two PDCA channels with the same PID**
   Wrong PDCA behavior when using two PDCA channels with the same PID.
   Fix/Workaround
   The same PID should not be assigned to more than one channel.

12.2.3 Rev. F

- *PWM*

1. **PWM channel interrupt enabling triggers an interrupt**
   When enabling a PWM channel that is configured with center aligned period (CALG=1), an interrupt is signalled.
   Fix/Workaround
   When using center aligned mode, enable the channel and read the status before channel interrupt is enabled.

2. **PWN counter restarts at 0x0001**
   The PWM counter restarts at 0x0001 and not 0x0000 as specified. Because of this the first PWM period has one more clock cycle.
   Fix/Workaround
   - The first period is 0x0000, 0x0001, ..., period.
   - Consecutive periods are 0x0001, 0x0002, ..., period.

3. **PWM update period to a 0 value does not work**
   It is impossible to update a period equal to 0 by the using the PWM update register (PWM_CUPD).
   Fix/Workaround
   Do not update the PWM_CUPD register with a value equal to 0.

4. **SPI**

5. **SPI Slave / PDCA transfer: no TX UNDERRUN flag**
   There is no TX UNDERRUN flag available, therefore in SPI slave mode, there is no way to be informed of a character lost in transmission.
   Fix/Workaround
   For PDCA transfer: none.

6. **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.

7. **SPI Glitch on RXREADY flag in slave mode when enabling the SPI or during the first transfer**
   In slave mode, the SPI can generate a false RXREADY signal during enabling of the SPI or during the first transfer.
   Fix/Workaround
   1. Set slave mode, set required CPOL/CPHA.
   2. Enable SPI.
   3. Set the polarity CPOL of the line in the opposite value of the required one.
   4. Set the polarity CPOL to the required one.
   5. Read the RXHOLDING register.
   Transfers can now begin and RXREADY will now behave as expected.

12.2.4 Rev. B

- *PWM*

1. **PWM channel interrupt enabling triggers an interrupt**
   When enabling a PWM channel that is configured with center aligned period (CALG=1), an interrupt is signalled.
   Fix/Workaround
   When using center aligned mode, enable the channel and read the status before channel interrupt is enabled.

2. **PWN counter restarts at 0x0001**
   The PWM counter restarts at 0x0001 and not 0x0000 as specified. Because of this the first PWM period has one more clock cycle.
   Fix/Workaround
   - The first period is 0x0000, 0x0001, ..., period.
   - Consecutive periods are 0x0001, 0x0002, ..., period.

3. **PWM update period to a 0 value does not work**
   It is impossible to update a period equal to 0 by the using the PWM update register (PWM_CUPD).
   Fix/Workaround
   Do not update the PWM_CUPD register with a value equal to 0.

4. **PWM channel status may be wrong if disabled before a period has elapsed**
   Before a PWM period has elapsed, the read channel status may be wrong. The CHIDx-bit for a PWM channel in the PWM Enable Register will read '1' for one full PWM period even if the channel was disabled before the period elapsed. It will then read '0' as expected.
   Fix/Workaround
   Reading the PWM channel status of a disabled channel is only correct after a PWM period has elapsed.

5. **The following alternate C functions PWM[4] on PA16 and PWM[6] on PA31 are not available on Rev B**
   The following alternate C functions PWM[4] on PA16 and PWM[6] on PA31 are not available on Rev B.
   Fix/Workaround
   Do not use these PWM alternate functions on these pins.

6. **SPI**

7. **SPI Slave / PDCA transfer: no TX UNDERRUN flag**
   There is no TX UNDERRUN flag available, therefore in SPI slave mode, there is no way to be informed of a character lost in transmission.
   Fix/Workaround
   For PDCA transfer: none.

- *USART*

1. **USART Manchester Encoder Not Working**
   Manchester encoding/decoding is not working.
   Fix/Workaround
   Do not use manchester encoding.

2. **USART RXBREAK problem when no timeguard**
   In asynchronous mode the RXBREAK flag is not correctly handled when the timeguard is 0 and the break character is located just after the stop bit.
   Fix/Workaround
   If the NBSTOP is 1, timeguard should be different from 0.

3. **USART Handshaking: 2 characters sent / CTS rises when TX**
   If CTS switches from 0 to 1 during the TX of a character, if the Holding register is not empty, the TXHOLDING is also transmitted.
   Fix/Workaround
   None.

4. **USART PDC and TIMEGUARD not supported in MANCHESTER**
   Manchester encoding/decoding is not working.
   Fix/Workaround
   Do not use manchester encoding.

5. **USART SPI mode is non functional on this revision**
   USART SPI mode is non functional on this revision.
   Fix/Workaround
   Do not use the USART SPI mode.

- *HMATRIX*

1. **HMatrix fixed priority arbitration does not work**
   Fixed priority arbitration does not work.
   Fix/Workaround
   Use Round-Robin arbitration instead.

- *Clock characteristic*

1. **PBA max frequency**
   The Peripheral bus A (PBA) max frequency is 30MHz instead of 60MHz.
   Fix/Workaround
   Do not set the PBA maximum frequency higher than 30MHz.

- *FLASHC*

1. **The address of Flash General Purpose Fuse Register Low (FGPFRLO) is 0xFFFE140C on revB instead of 0xFFFE1410**
   The address of Flash General Purpose Fuse Register Low (FGPFRLO) is 0xFFFE140C on revB instead of 0xFFFE1410.
   Fix/Workaround
   None.