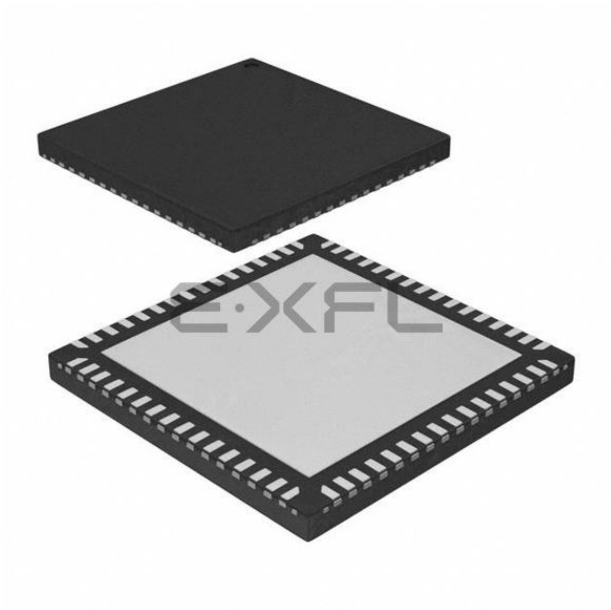


Welcome to E-XFL.COM

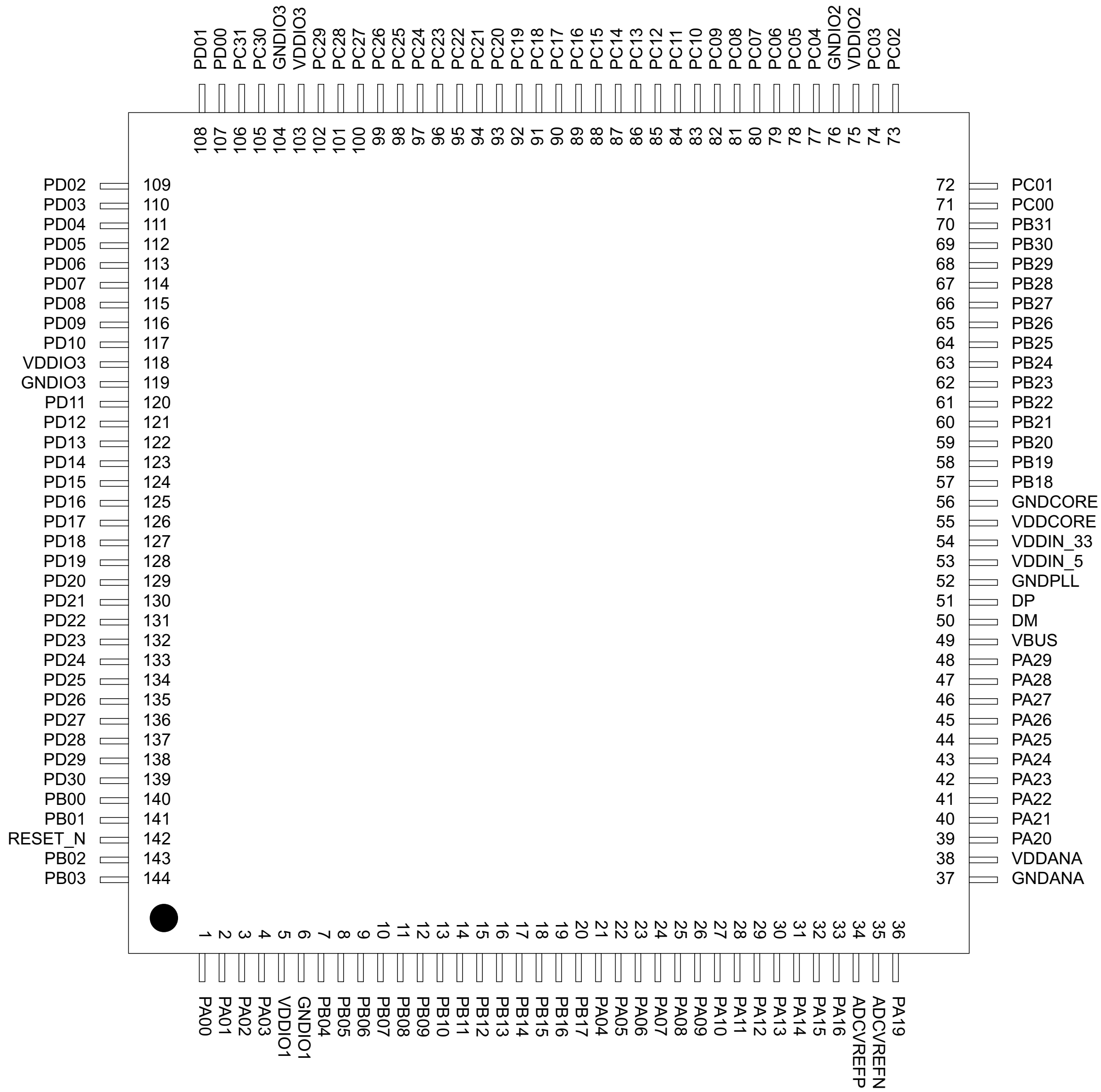
What is "Embedded - Microcontrollers"?

"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	66MHz
Connectivity	CANbus, Ethernet, I²C, IrDA, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	45
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°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/at32uc3c2128c-z2zr

Figure 3-3. LQFP144 Pinout



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 F
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 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 3-3. Oscillator pinout

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pad	Oscillator pin
31	47	69	PB30	xin0
	99	143	PB02	xin1
62	96	140	PB00	xin32
32	48	70	PB31	xout0
	100	144	PB03	xout1
63	97	141	PB01	xout32

3.2.4 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 3-4. JTAG pinout

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pin name	JTAG pin
2	2	2	PA01	TDI
3	3	3	PA02	TDO
4	4	4	PA03	TMS
1	1	1	PA00	TCK

3.2.5 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 GPIO configuration. Three different OCD trace pin mappings are possible, 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

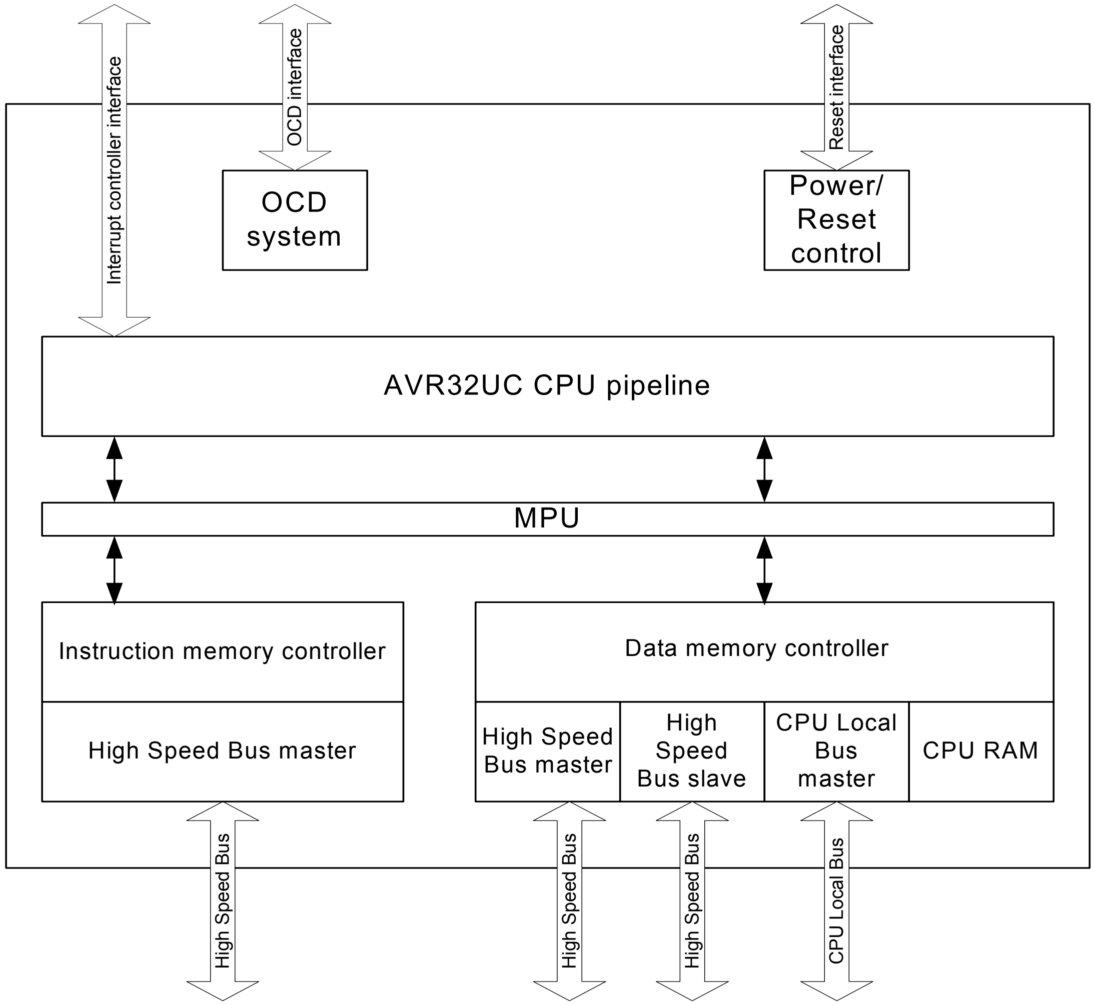
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced On-Chip Debug (OCD) system, no caches, and a Memory Protection Unit (MPU). A hardware Floating Point Unit (FPU) is also provided through the coprocessor instruction space. Java acceleration hardware is not implemented.

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 27 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 28 shows an overview of the AVR32UC pipeline stages.

4.3.2.5 *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

4.3.2.6 *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

4.3.2.7 *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.

4.5.3 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 38. 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

Table 5-3. Peripheral Address Mapping

Address	Peripheral	Description
0xFFFF0C00	AST	Asynchronous Timer - AST
0xFFFF1000	WDT	Watchdog Timer - WDT
0xFFFF1400	EIC	External Interrupt Controller - EIC
0xFFFF1800	FREQM	Frequency Meter - FREQM
0xFFFF2000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF2800	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0
0xFFFF2C00	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2
0xFFFF3000	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter - USART3
0xFFFF3400	SPI1	Serial Peripheral Interface - SPI1
0xFFFF3800	TWIM0	Two-wire Master Interface - TWIM0
0xFFFF3C00	TWIM1	Two-wire Master Interface - TWIM1
0xFFFF4000	TWIS0	Two-wire Slave Interface - TWIS0
0xFFFF4400	TWIS1	Two-wire Slave Interface - TWIS1
0xFFFF4800	IISC	Inter-IC Sound (I2S) Controller - IISC
0xFFFF4C00	PWM	Pulse Width Modulation Controller - PWM
0xFFFF5000	QDEC0	Quadrature Decoder - QDEC0
0xFFFF5400	QDEC1	Quadrature Decoder - QDEC1
0xFFFF5800	TC1	Timer/Counter - TC1
0xFFFF5C00	PEVC	Peripheral Event Controller - PEVC

6.2 Startup Considerations

This chapter summarizes the boot sequence of the AT32UC3C. 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

At power-up, the BOD33 and the BOD18 are enabled. The device will be held in a reset state by the power-up circuitry, until the VDDIN_33 (resp. VDDCORE) has reached the reset threshold of the BOD33 (resp BOD18). Refer to the Electrical Characteristics for the BOD thresholds. Once the power has stabilized, the device will use the System RC Oscillator (RCSYS, 115KHz typical frequency) as clock source. The BOD18 and BOD33 are kept enabled or are disabled according to the fuse settings (See the Fuse Setting section in the Flash Controller chapter).

On system start-up, the PLLs are 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 internal RC Oscillator.

6.2.2 Fetching of initial instructions

After reset has been released, the AVR32UC CPU starts fetching instructions from the reset address, which is 0x8000_0000. This address points to the first address in the internal Flash.

The internal Flash uses VDDIO voltage during read and write operations. It is recommended to use the BOD33 to monitor this voltage and make sure the VDDIO is above the minimum level (3.0V).

The code read from the internal Flash is free to configure the system to use for example the PLLs, to divide the frequency of the clock routed to some of the peripherals, and to gate the clocks to unused peripherals.

7.5 I/O Pin Characteristics

Table 7-6. Normal I/O Pin Characteristics[1]

Symbol	Parameter	Condition		Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance	V_{VDD} = 3V		5		26	kOhm
		V_{VDD} = 5V		5		16	kOhm
$R_{PULLDOWN}$	Pull-down resistance			2		16	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3V				0.3*V_{VDDIO}	V
		V_{VDD} = 4.5V				0.3*V_{VDDIO}	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V		0.7*V_{VDDIO}			V
		V_{VDD} = 5.5V		0.7*V_{VDDIO}			
V_{OL}	Output low-level voltage	I_{OL} = -3.5mA, pin drive x1[2]				0.5	V
		I_{OL} = -7mA, pin drive x2[2]					
		I_{OL} = -14mA, pin drive x4[2]					
V_{OH}	Output high-level voltage	I_{OH} = 3.5mA, pin drive x1[2]		V_{VDD} - 0.8			V
		I_{OH} = 7mA, pin drive x2[2]					
		I_{OH} = 14mA, pin drive x4[2]					
f_{MAX}	Output frequency[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			30	MHz
			load = 10pF, pin drive x2[2]			50	
			load = 10pF, pin drive x4[2]			60	
			load = 30pF, pin drive x1[2]			15	
			load = 30pF, pin drive x2[2]			25	
			load = 30pF, pin drive x4[2]			40	
		V_{VDD} =4.5V	load = 10pF, pin drive x1[2]			45	
			load = 10pF, pin drive x2[2]			65	
			load = 10pF, pin drive x4[2]			85	
			load = 30pF, pin drive x1[2]			20	
			load = 30pF, pin drive x2[2]			40	
			load = 30pF, pin drive x4[2]			60	

Table 7-31. ADC Transfer Characteristics (Continued)12-bit Resolution Mode[(1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 0 (F_{adc} = 1.5MHz)			12	Bit
INL	Integral Non-Linearity				5	LSB
DNL	Differential Non-Linearity				4	LSB
	Offset error		-20		20	mV
	Gain error		-30		30	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 7-32. ADC Transfer Characteristics 10-bit Resolution Mode[(1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 1 (F_{adc} = 1.5MHz)			10	Bit
INL	Integral Non-Linearity				1.25	LSB
DNL	Differential Non-Linearity				1.25	LSB
	Offset error		-10		10	mV
	Gain error		-20		20	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$= 3V, ADCFIA.SEQCFGn.SRES = 1 (F_{adc} = 1.5MHz)			10	Bit
INL	Integral Non-Linearity				1.25	LSB
DNL	Differential Non-Linearity				1.25	LSB
	Offset error		-20		20	mV
	Gain error		-25		25	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 7-33. ADC Transfer Characteristics 8-bit Resolution Mode[(1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 2 (F_{adc} =1.5MHz)			8	Bit
INL	Integral Non-Linearity				0.3	LSB
DNL	Differential Non-Linearity				0.3	LSB
	Offset error		-10		10	mV
	Gain error		-20		20	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 2 (F_{adc} = 1.5MHz)			8	Bit
INL	Integral Non-Linearity				0.3	LSB
DNL	Differential Non-Linearity				0.25	LSB
	Offset error		-25		25	mV
	Gain error		-25		25	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 7-36. ADC and S/H Transfer Characteristics (Continued)10-bit Resolution Mode and S/H gain from 1 to 16[(1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 1, S/H gain from 1 to 16 (F_{adc} = 1.5MHz)			10	Bit
INL	Integral Non-Linearity				2	LSB
DNL	Differential Non-Linearity				2	LSB
	Offset error		-30		30	mV
	Gain error		-30		30	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

7.8.7 Digital to Analog Converter (DAC) Characteristics

Table 7-37. Channel Conversion Time and DAC Clock

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{DAC}	DAC clock frequency				1	MHz
$t_{STARTUP}$	Startup time				3	µs
t_{CONV}	Conversion time (latency)	No S/H enabled, internal DAC			1	µs
		One S/H			1.5	µs
		Two S/H			2	µs
	Throughput rate				$1/t_{CONV}$	MSPS

Table 7-38. External Voltage Reference Input

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{DACREF}	DACREF input voltage range		1.2		V_{VDDANA}-0.7	V

Table 7-39. DAC Outputs

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	Output range	with external DAC reference	0.2		V_{DACREF}	V
		with internal DAC reference	0.2		V_{VDDANA}-0.7	
C_{LOAD}	Output capacitance		0		100	pF
R_{LOAD}	Output resitance		2			kΩ

Figure 7-4. DAC output

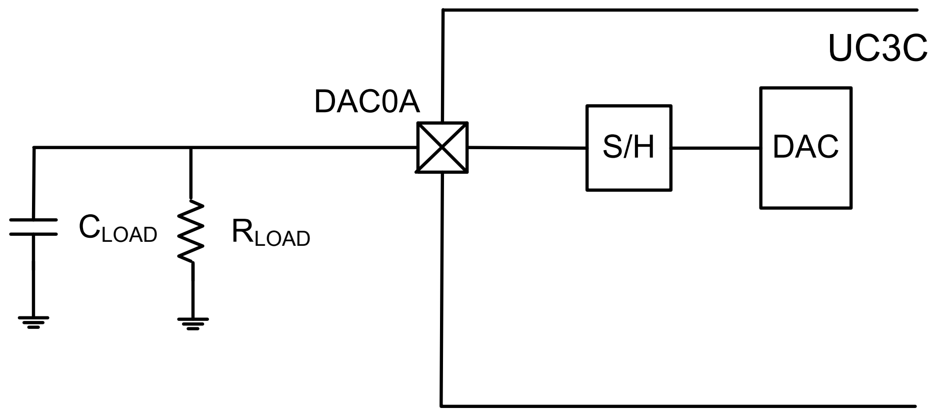


Table 7-40. Transfer Characteristics[(1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	V_{VDDANA} = 3V, V_{DACREF} = 2V, One S/H			12	Bit
INL	Integral Non-Linearity				20	LSB
DNL	Differential Non-linearity				20	LSB
	Offset error				80	mV
	Gain error				100	mV
RES	Resolution	V_{VDDANA} = 5V, V_{DACREF} = 3V, One S/H			12	Bit
INL	Integral Non-Linearity				20	LSB
DNL	Differential Non-linearity				20	LSB
	Offset error				120	mV
	Gain error				100	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

7.9.7 EBI Timings

See EBI I/O lines description for more details.

Table 7-52. SMC Clock Signal.

Symbol	Parameter	Max[1]	Units
$1/(t_{CPSMC})$	SMC Controller clock frequency	f_{cpu}	MHz

Note: 1. The maximum frequency of the SMC interface is the same as the max frequency for the HSB.

Table 7-53. SMC Read Signals with Hold Settings[1]

Symbol	Parameter	Conditions	Min	Units
		NRD Controlled (READ_MODE = 1)		
SMC_1	Data setup before NRD high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	34.4	ns
SMC_2	Data hold after NRD high		0	
SMC_3	NRD high to NBS0/A0 change[2]		nrd hold length * tCPSMC - 1.5	
SMC_4	NRD high to NBS1 change[2]		nrd hold length * tCPSMC - 0	
SMC_5	NRD high to NBS2/A1 change[2]		nrd hold length * tCPSMC - 0	
SMC_7	NRD high to A2 - A25 change[2]		nrd hold length * tCPSMC - 5.9	
SMC_8	NRD high to NCS inactive[2]		(nrd hold length - ncs rd hold length) * tCPSMC - 1.3	
SMC_9	NRD pulse width		nrd pulse length * tCPSMC - 0.9	
		NRD Controlled (READ_MODE = 0)		
SMC_{10}	Data setup before NCS high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	36.1	ns
SMC_{11}	Data hold after NCS high		0	
SMC_{12}	NCS high to NBS0/A0 change[2]		ncs rd hold length * tCPSMC - 3.2	
SMC_{13}	NCS high to NBS0/A0 change[2]		ncs rd hold length * tCPSMC - 2.2	
SMC_{14}	NCS high to NBS2/A1 change[2]		ncs rd hold length * tCPSMC - 1.2	
SMC_{16}	NCS high to A2 - A25 change[2]		ncs rd hold length * tCPSMC - 7.6	
SMC_{17}	NCS high to NRD inactive[2]		(ncs rd hold length - nrd hold length) * tCPSMC - 2.4	
SMC_{18}	NCS pulse width		ncs rd pulse length * tCPSMC - 3.3	

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.

2. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs rd hold length” or “nrd hold length”.

Table 7-54. SMC Read Signals with no Hold Settings[(1)]

Symbol	Parameter	Conditions	Min	Units
NRD Controlled (READ_MODE = 1)				
SMC_{19}	Data setup before NRD high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	34.4	ns
SMC_{20}	Data hold after NRD high		0	
NRD Controlled (READ_MODE = 0)				
SMC_{21}	Data setup before NCS high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	30.2	ns
SMC_{22}	Data hold after NCS high		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.

Table 7-55. SMC Write Signals with Hold Settings[(1)]

Symbol	Parameter	Conditions	Min	Units
NRD Controlled (READ_MODE = 1)				
SMC_{23}	Data Out valid before NWE high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	(nwe pulse length - 1) * t_{CPSMC} - 1.7	ns
SMC_{24}	Data Out valid after NWE high[(2)]		nwe pulse length * t_{CPSMC} - 5.1	
SMC_{25}	NWE high to NBS0/A0 change[(2)]		nwe pulse length * t_{CPSMC} - 2.8	
SMC_{29}	NWE high to NBS2/A1 change[(2)]		nwe pulse length * t_{CPSMC} - 0.8	
SMC_{31}	NWE high to A2 - A25 change[(2)]		nwe pulse length * t_{CPSMC} - 7.2	
SMC_{32}	NWE high to NCS inactive[(2)]		(nwe hold pulse - ncs wr hold length) * t_{CPSMC} - 2.6	
SMC_{33}	NWE pulse width		nwe pulse length * t_{CPSMC} - 0.4	
NRD Controlled (READ_MODE = 0)				
SMC_{34}	Data Out valid before NCS high	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	(ncs wr pulse length - 1) * t_{CPSMC} - 2.5	ns
SMC_{35}	Data Out valid after NCS high[(2)]		ncs wr hold length * t_{CPSMC} - 5.5	
SMC_{36}	NCS high to NWE inactive[(2)]		(ncs wr hold length - nwe hold length) * t_{CPSMC} - 2.2	

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.

2. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “nwe hold length”

9. Ordering Information

Table 9-1. Ordering Information

Device	Ordering Code	Carrier Type	Package	Temperature Operating Range
AT32UC3C0512C	AT32UC3C0512C-ALZT	Tray	LQFP 144	Automotive (-40°C to 125°C)
	AT32UC3C0512C-ALZR	Tape & Reel		
AT32UC3C1512C	AT32UC3C1512C-AZT	Tray	TQFP 100	
	AT32UC3C1512C-AZR	Tape & Reel		
AT32UC3C1256C	AT32UC3C1256C-AZT	Tray	TQFP 100	
	AT32UC3C1256C-AZR	Tape & Reel		
AT32UC3C2512C	AT32UC3C2512C-A2ZT	Tray	TQFP 64	
	AT32UC3C2512C-A2ZR	Tape & Reel		
AT32UC3C2512C	AT32UC3C2512C-Z2ZT	Tray	QFN 64	
	AT32UC3C2512C-Z2ZR	Tape & Reel		
AT32UC3C2256C	AT32UC3C2256C-A2ZT	Tray	TQFP 64	
	AT32UC3C2256C-A2ZR	Tape & Reel		
AT32UC3C2256C	AT32UC3C2256C-Z2ZT	Tray	QFN 64	
	AT32UC3C2256C-Z2ZR	Tape & Reel		
AT32UC3C2128C	AT32UC3C2128C-A2ZT	Tray	TQFP 64	
	AT32UC3C2128C-A2ZR	Tape & Reel		
AT32UC3C2128C	AT32UC3C2128C-Z2ZT	Tray	QFN 64	
	AT32UC3C2128C-Z2ZR	Tape & Reel		

10.1.5 SCIF

1 **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

2 **PLL lock might not clear after disable**
Under certain circumstances, the lock signal from the Phase Locked Loop (PLL) oscillator may not go back to zero after the PLL oscillator has been disabled. This can cause the propagation of clock signals with the wrong frequency to parts of the system that use the PLL clock.
Fix/Workaround
PLL must be turned off before entering STOP, DEEPSTOP or STATIC sleep modes. If PLL has been turned off, a delay of 30us must be observed after the PLL has been enabled again before the SCIF.PLL0LOCK bit can be used as a valid indication that the PLL is locked.

3 **BOD33 reset locks the device**
If BOD33 is enabled as a reset source (SCIF.BOD33.CTRL=0x1) and when VDDIN_33 power supply voltage falls below the BOD33 voltage (SCIF.BOD33.LEVEL), the device is locked permanently under reset even if the power supply goes back above BOD33 reset level. In order to unlock the device, an external reset event should be applied on RESET_N.
Fix/Workaround
Use an external BOD on VDDIN_33 or an external reset source.

10.1.6 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.

10.1.7 TC

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.

10.1.8 TWIM

1 **SMBALERT bit may be set after reset**
For TWIM0 and TWIM1 modules, the SMBus Alert (SMBALERT) bit in the Status Register (SR) might be erroneously set after system reset.
Fix/Workaround
After system reset, clear the SR.SMBALERT bit before commencing any TWI transfer.

For TWIM2 module, the SMBus Alert (SMBALERT) is not implemented but the bit in the Status Register (SR) is erroneously set once TWIM2 is enabled.
Fix/Workaround
None.

10.1.9 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.

10.1.10 USBC

1 **UPINRQx.INRQ field is limited to 8-bits**
In Host mode, when using the UPINRQx.INRQ feature together with the multi-packet mode to launch a finite number of packet among multi-packet, the multi-packet size (located in the descriptor table) is limited to the UPINRQx.INRQ value multiply by the pipe size.
Fix/Workaround
UPINRQx.INRQ value shall be less than the number of configured multi-packet.

2 **In USB host mode, downstream resume feature does not work (UHCON.RESUME=1).**

2 Requesting clocks in idle sleep modes will mask all other PB clocks than the requested

In idle or frozen sleep mode, all the PB clocks will be frozen if the TWIS or the AST need to wake the cpu up.

Fix/Workaround

Disable the TWIS or the AST before entering idle or frozen sleep mode.

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.

10.2.6 SCIF

1 PLLCOUNT value larger than zero can cause PLLEN glitch

Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.

Fix/Workaround

The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

2 PLL lock might not clear after disable

Under certain circumstances, the lock signal from the Phase Locked Loop (PLL) oscillator may not go back to zero after the PLL oscillator has been disabled. This can cause the propagation of clock signals with the wrong frequency to parts of the system that use the PLL clock.

Fix/Workaround

PLL must be turned off before entering STOP, DEEPSTOP or STATIC sleep modes. If PLL has been turned off, a delay of 30us must be observed after the PLL has been enabled again before the SCIF.PLL0LOCK bit can be used as a valid indication that the PLL is locked.

3 BOD33 reset locks the device

If BOD33 is enabled as a reset source (SCIF.BOD33.CTRL=0x1) and when VDDIN_33 power supply voltage falls below the BOD33 voltage (SCIF.BOD33.LEVEL), the device is locked permanently under reset even if the power supply goes back above BOD33 reset level. In order to unlock the device, an external reset event should be applied on RESET_N.

Fix/Workaround

Use an external BOD on VDDIN_33 or an external reset source.

10.2.7 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.

10.2.8 TC

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.

10.2.9 TWIM

1 **SMBALERT bit may be set after reset**
For TWIM0 and TWIM1 modules, the SMBus Alert (SMBALERT) bit in the Status Register (SR) might be erroneously set after system reset.
Fix/Workaround
After system reset, clear the SR.SMBALERT bit before commencing any TWI transfer.

For TWIM2 module, the SMBus Alert (SMBALERT) is not implemented but the bit in the Status Register (SR) is erroneously set once TWIM2 is enabled.
Fix/Workaround
None.

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.

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. D – 01/12

1	Errata: Updated
2	PM: Clock Mask Table Updated
3	Fixed PLLOPT field description in SCIF chapter
4	MDMA: Swapped bit descriptions for IER and IDR
5	MACB: USRIO register description and bit descriptions for IMR/IDR/IER Updated
6	USBC: UPCON.PFREEZE and UPINRQn description Updated
7	ACIFA: Updated
8	ADCIFA: CFG.MUXSET, SSMQ description and conversion results section Updated
9	DACIFB: Calibration section Updated
10	Electrical Characteristics: ADCREFP/ADCREFN added
11	Add devices: C1256C, C2256C, C2128C

11.2 Rev. C – 08/11

1	Electrical Characteristics Updated: - I/O Pins characteristics - 8MHz/1MHz RC Oscillator (RC8M) characteristics - 1.8V Voltage Regulator characteristics - 3.3V Voltage Regulator characteristics - 1.8VBrown Out Detector (BOD18) characteristics - 3.3VBrown Out Detector (BOD33) characteristics - 5VBrown Out Detector (BOD50) characteristics - Analog to Digital Converter (ADC) and sample and hold (S/DH) Characteristics - Analog Comparator characteristics
2	Errata: Updated
3	TWIS: Updated

11.3 Rev. B – 02/11

1	Package and pinout: Added supply column. Updated peripheral functions
2	Supply and Startup Considerations: Updated I/O lines power
3	PM: Added AWEN description