

Welcome to [E-XFL.COM](https://www.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	EBI/EMI, I ² C, IrDA, Memory Card, SPI, SSC, UART/USART, USB OTG
Peripherals	Brown-out Detect/Reset, DMA, POR, WDT
Number of I/O	88
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.75V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-VFBGA
Supplier Device Package	100-VFBGA (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3a4256s-c1ut

Figure 3-2. LQFP144 Pinout

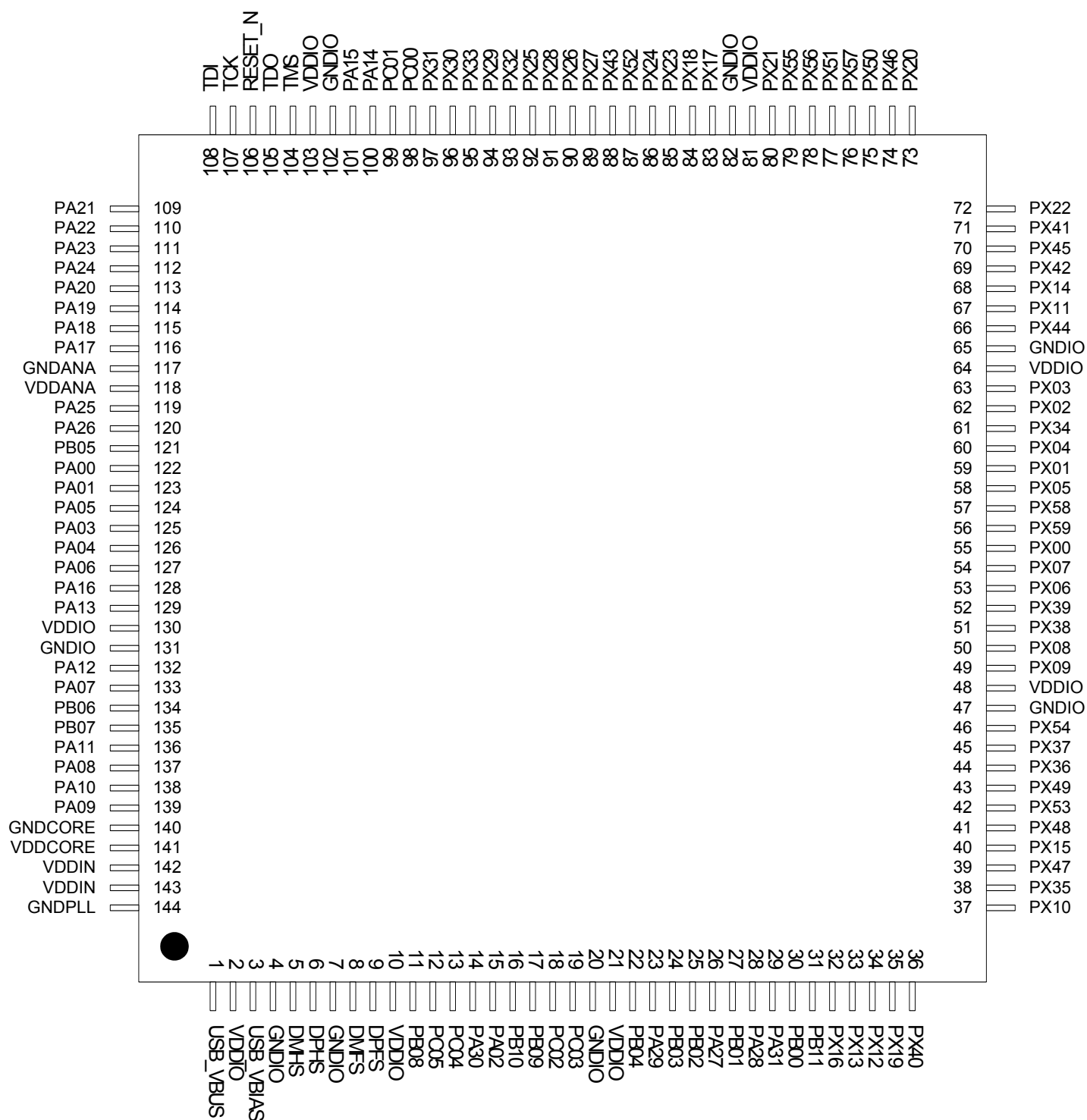


Figure 3-3. VFBGA100 Pinout (top view)

	1	2	3	4	5	6	7	8	9	10
A	PA28	PA27	PB04	PA30	PC02	PC03	PC05	DPHS	DMHS	USB_VBUS
B	PB00	PB01	PB02	PA29	VDDIO	VDDIO	PC04	DPFS	DMFS	GNDPLL
C	PB11	PA31	GNDIO	PB03	PB09	PB08	USB_VBIAS	GNDIO	PA11	PA10
D	PX12	PX10	PX13	PX16/ PX53 ⁽¹⁾	PB10	PB07	PB06	PA09	VDDIN	VDDIN
E	PA02/ PX47 ⁽¹⁾	GNDIO	PX08	PX09	VDDIO	GNDIO	PA16	PA06/ PA13 ⁽¹⁾	PA04	VDDCORE
F	PX19/ PX59 ⁽¹⁾	VDDIO	PX06	PX07	GNDIO	VDDIO	PA26/ PB05 ⁽¹⁾	PA08	PA03	GNDCORE
G	PX05	PX01	PX02	PX00	PX30	PA23/ PX46 ⁽¹⁾	PA12/ PA25 ⁽¹⁾	PA00/ PA18 ⁽¹⁾	PA05	PA01/ PA17 ⁽¹⁾
H	PX04	PX21	GNDIO	PX25	PX31	PA22/ PX20 ⁽¹⁾	TMS	GNDANA	PA20/ PX18 ⁽¹⁾	PA07/ PA19 ⁽¹⁾
J	PX03	PX24	PX26	PX29	VDDIO	VDDANA	PA15/ PX45 ⁽¹⁾	TDO	RESET_N	PA24/ PX17 ⁽¹⁾
K	PX23	PX27	PX28	PX15/ PX32 ⁽¹⁾	PC00/ PX14 ⁽¹⁾	PC01	PA14/ PX11 ⁽¹⁾	TDI	TCK	PA21/ PX22 ⁽¹⁾

Note: 1. Those balls are physically connected to 2 GPIOs. Software must managed carrefully the GPIO configuration to avoid electrical conflict

Table 3-1. GPIO Controller Function Multiplexing

BGA 144	QFP 144	BGA 100	PIN	G P I O	Supply	PIN Type (2)	GPIO function			
							A	B	C	D
J4	78		PX56	107	VDDIO	x2	EBI - ADDR[21]	EIC - SCAN[2]	USART2 - TXD	
H4	76		PX57	108	VDDIO	x2	EBI - ADDR[20]	EIC - SCAN[1]	USART3 - RXD	
H3	57		PX58	109	VDDIO	x2	EBI - NCS[0]	EIC - SCAN[0]	USART3 - TXD	
G3	56	F1 ⁽¹⁾	PX59	110	VDDIO	x2	EBI - NANDWE		MCI - CMD[1]	

- Note:
1. Those balls are physically connected to 2 GPIOs. Software must managed carefully the GPIO configuration to avoid electrical conflict.
 2. Refer to ["Electrical Characteristics" on page 40](#) for a description of the electrical properties of the pad types used..

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 D
Nexus OCD AUX port connections	OCD trace system
JTAG port connections	JTAG debug port
Oscillators	OSC0, OSC1, 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 Power Manager (PM). Please refer to the PM chapter for more information about this.

Table 3-3.Oscillator Pinout

TFBGA144	QFP144	VFBGA100	Pin name	Oscillator pin
A7	18	A5	PC02	XIN0
B7	19	A6	PC03	XOUT0
A8	13	B7	PC04	XIN1
A9	12	A7	PC05	XOUT1
K5	98	K5 ⁽¹⁾	PC00	XIN32
H6	99	K6	PC01	XOUT32

- Note:
1. This ball is physically connected to 2 GPIOs. Software must managed carefully the GPIO configuration to avoid electrical conflict

Table 3-6. Signal Description List

Signal Name	Function	Type	Active Level	Comments
RESET_N	Reset Pin	Input	Low	
DMA Controller - DMACA (optional)				
DMAACK[1:0]	DMA Acknowledge	Output		
DMARQ[1:0]	DMA Requests	Input		
External Interrupt Controller - EIC				
EXTINT[7:0]	External Interrupt Pins	Input		
SCAN[7:0]	Keypad Scan Pins	Output		
NMI	Non-Maskable Interrupt Pin	Input	Low	
General Purpose Input/Output pin - GPIOA, GPIOB, GPIOC, GPIOX				
PA[31:0]	Parallel I/O Controller GPIO port A	I/O		
PB[11:0]	Parallel I/O Controller GPIO port B	I/O		
PC[5:0]	Parallel I/O Controller GPIO port C	I/O		
PX[59:0]	Parallel I/O Controller GPIO port X	I/O		
External Bus Interface - EBI				
ADDR[23:0]	Address Bus	Output		
CAS	Column Signal	Output	Low	
CFCE1	Compact Flash 1 Chip Enable	Output	Low	
CFCE2	Compact Flash 2 Chip Enable	Output	Low	
CFRNW	Compact Flash Read Not Write	Output		
DATA[15:0]	Data Bus	I/O		
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
NCS[5:0]	Chip Select	Output	Low	
NRD	Read Signal	Output	Low	
NWAIT	External Wait Signal	Input	Low	
NWE0	Write Enable 0	Output	Low	
NWE1	Write Enable 1	Output	Low	
RAS	Row Signal	Output	Low	

Table 3-6. Signal Description List

Signal Name	Function	Type	Active Level	Comments
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 - TWI0, TWI1				
TWCK	Serial Clock	I/O		
TWD	Serial Data	I/O		
TWALM	SMBALERT signal	I/O		
Universal Synchronous Asynchronous Receiver Transmitter - USART0, USART1, USART2, USART3				
CLK	Clock	I/O		
CTS	Clear To Send	Input		
DCD	Data Carrier Detect			Only USART1
DSR	Data Set Ready			Only USART1
DTR	Data Terminal Ready			Only USART1
RI	Ring Indicator			Only USART1
RTS	Request To Send	Output		
RXD	Receive Data	Input		
TXD	Transmit Data	Output		
Analog to Digital Converter - ADC				
AD0 - AD7	Analog input pins	Analog input		
Audio Bitstream DAC (ABDAC)				
DATA0-DATA1	D/A Data out	Output		
DATAN0-DATAN1	D/A Data inverted out	Output		
Universal Serial Bus Device - USB				
DMFS	USB Full Speed Data -	Analog		
DPFS	USB Full Speed Data +	Analog		

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.

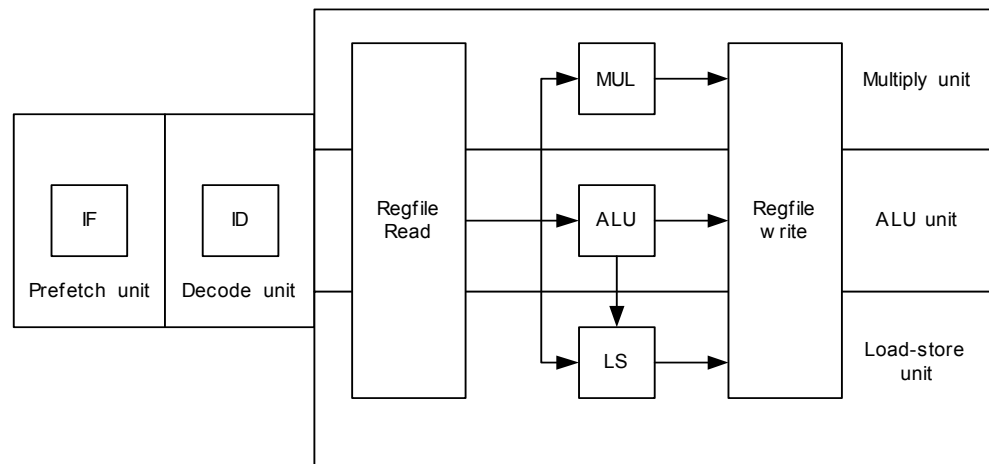
4.3 The AVR32UC CPU

The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced 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 4-1 on page 23](#) displays the contents of AVR32UC.

Figure 4-2. The AVR32UC Pipeline

4.3.2 AVR32A Microarchitecture Compliance

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

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

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

4.3.3 Java Support

AVR32UC does not provide Java hardware acceleration.

4.3.4 Memory Protection

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

4.3.5 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.

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

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

Debug state is exited by the *retd* instruction.

4.4.4 System Registers

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

Table 4-3. System Registers

Reg #	Address	Name	Function
0	0	SR	Status Register
1	4	EVBA	Exception Vector Base Address
2	8	ACBA	Application Call Base Address
3	12	CPUCR	CPU Control Register
4	16	ECR	Exception Cause Register
5	20	RSR_SUP	Unused in AVR32UC
6	24	RSR_INT0	Unused in AVR32UC
7	28	RSR_INT1	Unused in AVR32UC
8	32	RSR_INT2	Unused in AVR32UC
9	36	RSR_INT3	Unused in AVR32UC
10	40	RSR_EX	Unused in AVR32UC
11	44	RSR_NMI	Unused in AVR32UC
12	48	RSR_DBG	Return Status Register for Debug mode
13	52	RAR_SUP	Unused in AVR32UC
14	56	RAR_INT0	Unused in AVR32UC
15	60	RAR_INT1	Unused in AVR32UC
16	64	RAR_INT2	Unused in AVR32UC
17	68	RAR_INT3	Unused in AVR32UC
18	72	RAR_EX	Unused in AVR32UC
19	76	RAR_NMI	Unused in AVR32UC
20	80	RAR_DBG	Return Address Register for Debug mode
21	84	JECCR	Unused in AVR32UC
22	88	JOSP	Unused in AVR32UC
23	92	JAVA_LV0	Unused in AVR32UC
24	96	JAVA_LV1	Unused in AVR32UC
25	100	JAVA_LV2	Unused in AVR32UC

Table 4-3. System Registers (Continued)

Reg #	Address	Name	Function
26	104	JAVA_LV3	Unused in AVR32UC
27	108	JAVA_LV4	Unused in AVR32UC
28	112	JAVA_LV5	Unused in AVR32UC
29	116	JAVA_LV6	Unused in AVR32UC
30	120	JAVA_LV7	Unused in AVR32UC
31	124	JTBA	Unused in AVR32UC
32	128	JBCR	Unused in AVR32UC
33-63	132-252	Reserved	Reserved for future use
64	256	CONFIG0	Configuration register 0
65	260	CONFIG1	Configuration register 1
66	264	COUNT	Cycle Counter register
67	268	COMPARE	Compare register
68	272	TLBEHI	Unused in AVR32UC
69	276	TLBELO	Unused in AVR32UC
70	280	PTBR	Unused in AVR32UC
71	284	TLBEAR	Unused in AVR32UC
72	288	MMUCR	Unused in AVR32UC
73	292	TLBARLO	Unused in AVR32UC
74	296	TLBARHI	Unused in AVR32UC
75	300	PCCNT	Unused in AVR32UC
76	304	PCNT0	Unused in AVR32UC
77	308	PCNT1	Unused in AVR32UC
78	312	PCCR	Unused in AVR32UC
79	316	BEAR	Bus Error Address Register
80	320	MPUAR0	MPU Address Register region 0
81	324	MPUAR1	MPU Address Register region 1
82	328	MPUAR2	MPU Address Register region 2
83	332	MPUAR3	MPU Address Register region 3
84	336	MPUAR4	MPU Address Register region 4
85	340	MPUAR5	MPU Address Register region 5
86	344	MPUAR6	MPU Address Register region 6
87	348	MPUAR7	MPU Address Register region 7
88	352	MPUPSR0	MPU Privilege Select Register region 0
89	356	MPUPSR1	MPU Privilege Select Register region 1
90	360	MPUPSR2	MPU Privilege Select Register region 2
91	364	MPUPSR3	MPU Privilege Select Register region 3

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 0x8000_0000. This places the reset address in the boot flash memory area.

TLB miss exceptions and *sca//* 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 external interrupt requests have entry points located at an offset relative to EVBA. This autovector offset is specified by an external 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. If events occur on several instructions at different locations in the pipeline, the events on the oldest instruction are always handled before any events on any younger instruction, even if the younger instruction has events of higher priority than the oldest instruction. An instruction B is younger than an instruction A if it was sent down the pipeline later than A.

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

Table 5-3. Local Bus Mapped GPIO Registers

Port	Register	Mode	Local Bus Address	Access
2	Output Driver Enable Register (ODER)	WRITE	0x40000240	Write-only
		SET	0x40000244	Write-only
		CLEAR	0x40000248	Write-only
		TOGGLE	0x4000024C	Write-only
	Output Value Register (OVR)	WRITE	0x40000250	Write-only
		SET	0x40000254	Write-only
		CLEAR	0x40000258	Write-only
		TOGGLE	0x4000025C	Write-only
	Pin Value Register (PVR)	-	0x40000260	Read-only
3	Output Driver Enable Register (ODER)	WRITE	0x40000340	Write-only
		SET	0x40000344	Write-only
		CLEAR	0x40000348	Write-only
		TOGGLE	0x4000034C	Write-only
	Output Value Register (OVR)	WRITE	0x40000350	Write-only
		SET	0x40000354	Write-only
		CLEAR	0x40000358	Write-only
		TOGGLE	0x4000035C	Write-only
	Pin Value Register (PVR)	-	0x40000360	Read-only

7.2 DC Characteristics

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

Table 7-1. DC Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V _{VDDIO}	DC Supply Peripheral I/Os		3.0		3.6	V
V _{VDDANA}	DC Analog Supply		3.0		3.6	V
V _{IL}	Input Low-level Voltage	All I/O pins except TWCK, TWD, RESET_N, TCK, TDI	-0.3		+0.8	V
		TWCK, TWD	V _{VDDIO} x0.7		V _{VDDIO} +0.5	V
		RESET_N, TCK, TDI	+0.8V			V
V _{IH}	Input High-level Voltage	All I/O pins except TWCK, TWD	2.0		3.6	V
		TWCK, TWD				V
V _{OL}	Output Low-level Voltage	I _{OL} = -2mA for Pin drive x1 I _{OL} = -4mA for Pin drive x2 I _{OL} = -8mA for Pin drive x3			0.4	V
V _{OH}	Output High-level Voltage	I _{OH} = 2mA for Pin drive x1 I _{OH} = 4mA for Pin drive x2 I _{OH} = 8mA for Pin drive x3	V _{VDDIO} -0.4			V
I _{LEAK}	Input Leakage Current	Pullup resistors disabled		0.05	1	μA
C _{IN}	Input Capacitance			7		pF
R _{PULLUP}	Pull-up Resistance	All I/O pins except RESET_N, TCK, TDI, TMS	9	15	25	KΩ
		RESET_N, TCK, TDI, TMS	5		25	KΩ
I _O	Output Current Pin drive 1x Pin drive 2x Pin drive 3x				2.0 4.0 8.0	mA
I _{SC}	Static Current	On V _{VDDIN} = 3.3V, CPU in static mode	T _A = 25°C	30		μA
			T _A = 85°C	175		μA

7.2.1 I/O Pin Output Level Typical Characteristics

Figure 7-1. I/O Pin drive x2 Output Low Level Voltage (VOL) vs. Source Current

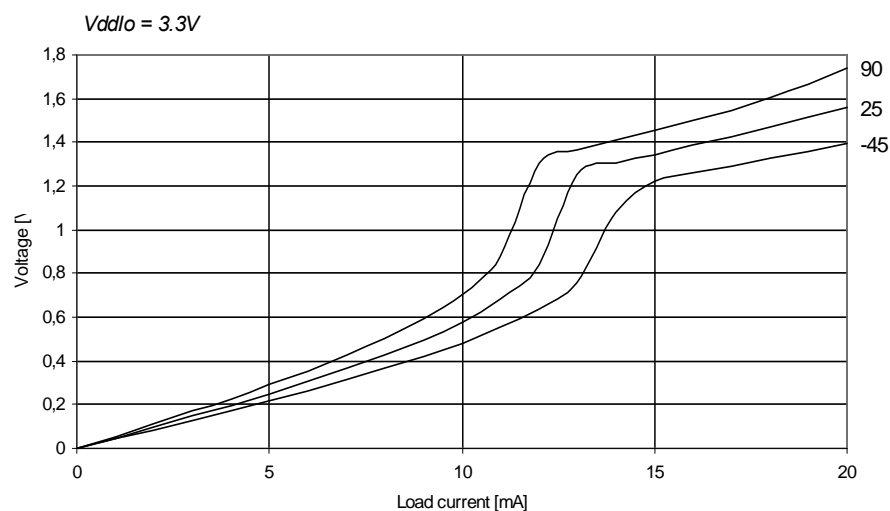
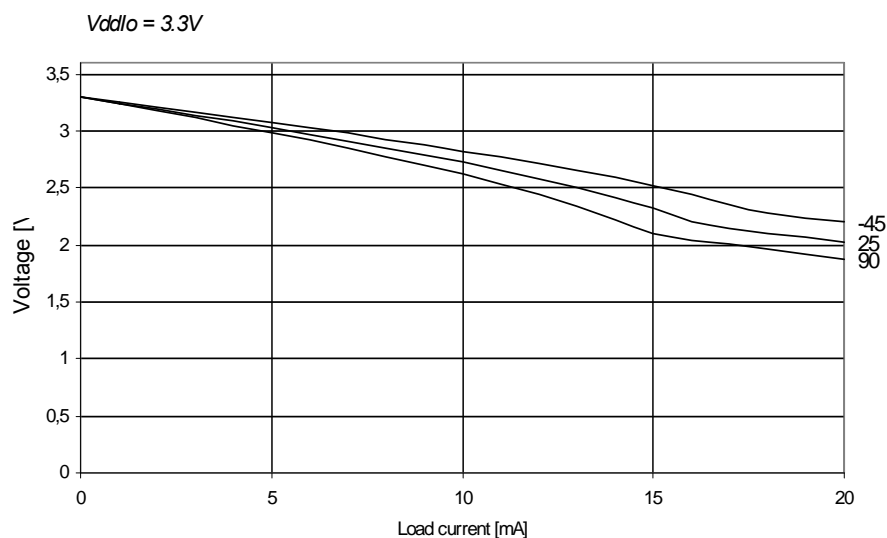


Figure 7-2. I/O Pin drive x2 Output High Level Voltage (VOH) vs. Source Current

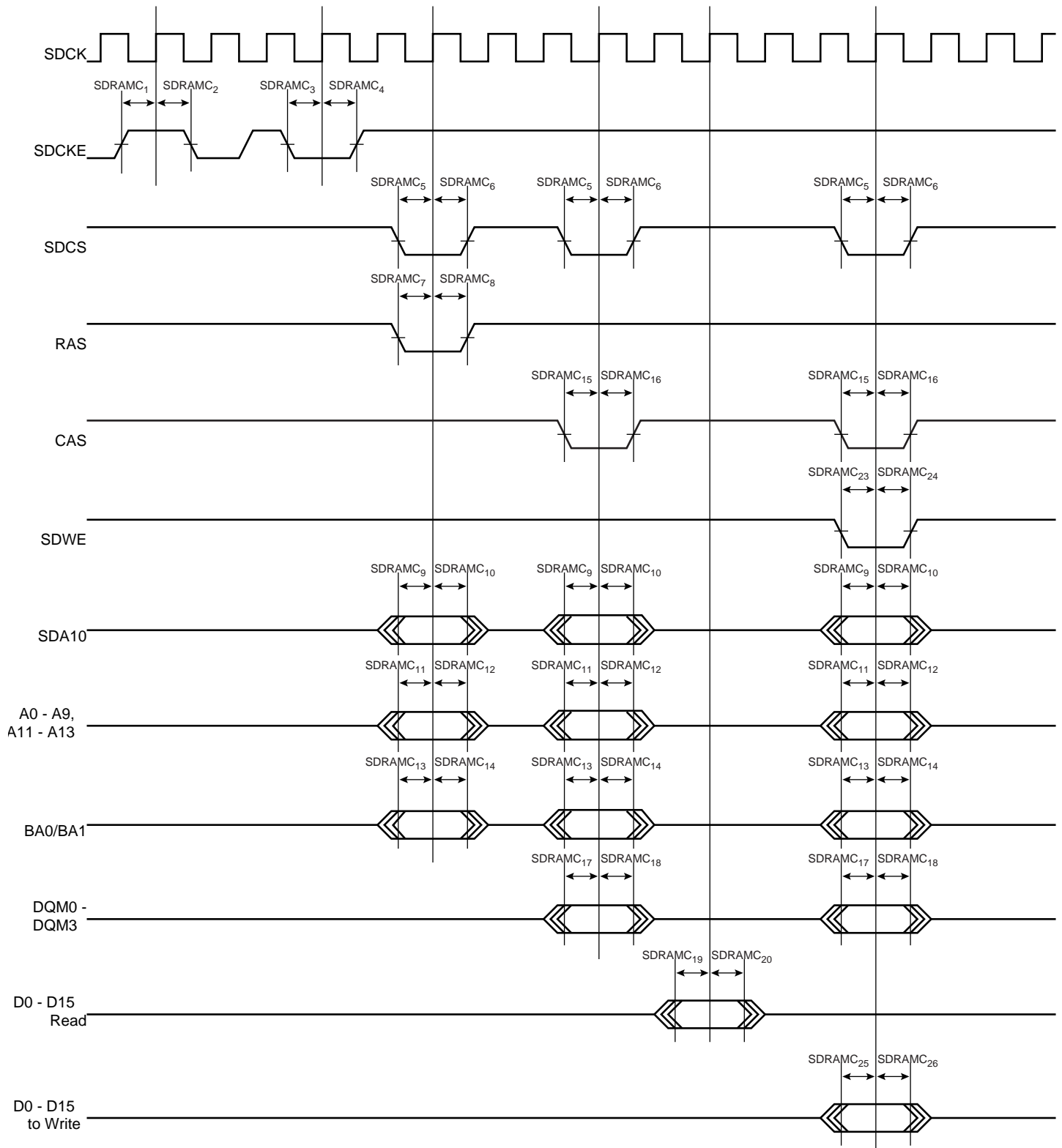


7.3 I/O pin Characteristics

These parameters are given in the following conditions:

- $V_{DDCORE} = 1.8V$
- $V_{DDIO} = 3.3V$
- Ambient Temperature = 25°C

Figure 7-9. SDRAMC Signals relative to SDCK.



8. Mechanical Characteristics

8.1 Thermal Considerations

8.1.1 Thermal Data

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

Table 8-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TQFP144	40.3	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP144	9.5	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TFBGA144	28.5	°C/W
θ_{JC}	Junction-to-case thermal resistance		TFBGA144	6.9	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	VFBGA100	31.1	°C/W
θ_{JC}	Junction-to-case thermal resistance		VFBGA100	6.9	

8.1.2 Junction Temperature

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

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

where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in [Table 8-1 on page 68](#).
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in [Table 8-1 on page 68](#).
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section "[Regulator characteristics](#)" on page 43.
- T_A = ambient temperature (°C).

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

9. Ordering Information

Device	Ordering Code	Package	Conditioning	Temperature Operating Range
AT32UC3A3256S	AT32UC3A3256S-ALUT	144-lead LQFP	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3256S-ALUR	144-lead LQFP	Reels	Industrial (-40-C to 85-C)
	AT32UC3A3256S-CTUT	144-ball TFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3256S-CTUR	144-ball TFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A3256	AT32UC3A3256-ALUT	144-lead LQFP	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3256-ALUR	144-lead LQFP	Reels	Industrial (-40-C to 85-C)
	AT32UC3A3256-CTUT	144-ball TFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3256-CTUR	144-ball TFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A3128S	AT32UC3A3128S-ALUT	144-lead LQFP	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3128S-ALUR	144-lead LQFP	Reels	Industrial (-40-C to 85-C)
	AT32UC3A3128S-CTUT	144-ball TFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3128S-CTUR	144-ball TFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A3128	AT32UC3A3128-ALUT	144-lead LQFP	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3128-ALUR	144-lead LQFP	Reels	Industrial (-40-C to 85-C)
	AT32UC3A3128-CTUT	144-ball TFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A3128-CTUR	144-ball TFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A364S	AT32UC3A364S-ALUT	144-lead LQFP	Tray	Industrial (-40-C to 85-C)
	AT32UC3A364S-ALUR	144-lead LQFP	Reels	Industrial (-40-C to 85-C)
	AT32UC3A364S-CTUT	144-ball TFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A364S-CTUR	144-ball TFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A364	AT32UC3A364-ALUT	144-lead LQFP	Tray	Industrial (-40-C to 85-C)
	AT32UC3A364-ALUR	144-lead LQFP	Reels	Industrial (-40-C to 85-C)
	AT32UC3A364-CTUT	144-ball TFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A364-CTUR	144-ball TFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A4256S	AT32UC3A4256S-C1UT	100-ball VFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A4256S-C1UR	100-ball VFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A4256	AT32UC3A4256-C1UT	100-ball VFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A4256-C1UR	100-ball VFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A4128S	AT32UC3A4128S-C1UT	100-ball VFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A4128S-C1UR	100-ball VFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A4128	AT32UC3A4128-C1UT	100-ball VFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A4128-C1UR	100-ball VFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A464S	AT32UC3A464S-C1UT	100-ball VFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A464S-C1UR	100-ball VFBGA	Reels	Industrial (-40-C to 85-C)
AT32UC3A464	AT32UC3A464-C1UT	100-ball VFBGA	Tray	Industrial (-40-C to 85-C)
	AT32UC3A464-C1UR	100-ball VFBGA	Reels	Industrial (-40-C to 85-C)

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.

10.1.14 SSC

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.

10.1.15 FLASHC

Corrupted read in flash may happen after fuses write or erase operations (FLASHC LP, UP, WGPB, EGPB, SSB, PGPFB, EAGPF commands)

After a flash fuse write or erase operation (FLASHC LP, UP, WGPB, EGPB, SSB, PGPFB, EAGPF commands), reading (data read or code fetch) in flash may fail. This may lead to an exception or to other errors derived from this corrupted read access.

Fix/Workaround

Before the flash fuse write or erase operation, enable the flash high speed mode (FLASHC HSEN command). The flash fuse write or erase operations (FLASHC LP, UP, WGPB, EGPB, SSB, PGPFB, EAGPF commands) must be issued from RAM or through the EBI. After these commands, read 3 times one flash page initialized to 00h. Disable the flash high speed mode (FLASHC HSDIS command). It is then possible to safely read or code fetch the flash.

10.2 Rev. E

10.2.1 General

Devices cannot operate with CPU frequency higher than 66MHz in 1WS and 36MHz in 0WS

Fix/Workaround

None

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 System Control Interface (SCIF) before going to any sleep mode where the OSC0 is disabled, or pull down or up XIN0 and XOUT0 with 1 Mohm resistor.

Power consumption in static mode The power consumption in static mode can be up to 330µA on some parts (typical at 25°C)

Fix/Workaround

Set to 1b bit CORRS4 of the ECCHRS mode register (MD). In C-code: *((volatile int*) (0xFFFE2404))= 0x400.

DMACA data transfer fails when CTLx.SRC_TR_WIDTH is not equal to CTLx.DST_TR_WIDTH

Fix/Workaround

For any DMACA transfer make sure CTLx.SRC_TR_WIDTH = CTLx.DST_TR_WIDTH.

3.3V supply monitor is not available

FGPFRLO[30:29] are reserved and should not be used by the application.

Fix/Workaround

None.

Service access bus (SAB) can not access DMACA registers

Fix/Workaround

None.

10.2.2 Processor and Architecture

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.

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.

When the main clock is RCSYS, TIMER_CLOCK5 is equal to PBA clock

When the main clock is generated from RCSYS, TIMER_CLOCK5 is equal to PBA Clock and not PBA Clock / 128.

Fix/Workaround

None.

10.2.3 MPU

Privilege violation when using interrupts in application mode with protected system stack

If the system stack is protected by the MPU and an interrupt occurs in application mode, an MPU DTLB exception will occur.

Fix/Workaround

Make a DTLB Protection (Write) exception handler which permits the interrupt request to be handled in privileged mode.

10.2.4 USB

UPCFGn.INTFRQ is irrelevant for isochronous pipe

As a consequence, isochronous IN and OUT tokens are sent every 1 ms (Full Speed), or every 125uS (High Speed).

Fix/Workaround

Before going to sleep modes where the system RC oscillator is stopped, make sure that the factor between the CPU/HSB and PBx frequencies is less than or equal to 4.

10.2.9 PDCA

PCONTROL.CHxRES is non-functional

PCONTROL.CHxRES is non-functional. Counters are reset at power-on, and cannot be reset by software.

Fix/Workaround

Software needs to keep history of performance counters.

Transfer error will stall a transmit peripheral handshake interface

If a transfer error is encountered on a channel transmitting to a peripheral, the peripheral handshake of the active channel will stall and the PDCA will not do any more transfers on the affected peripheral handshake interface.

Fix/Workaround

Disable and then enable the peripheral after the transfer error.

10.2.10 AES

URAD (Unspecified Register Access Detection Status) does not detect read accesses to the write-only KEYW[5..8]R registers

Fix/Workaround

None.

10.2.11 HMATRIX

In the PRAS and PRBS registers, the MxPR fields are only two bits

In the PRAS and PRBS registers, the MxPR fields are only two bits wide, instead of four bits. The unused bits are undefined when reading the registers.

Fix/Workaround

Mask undefined bits when reading PRAS and PRBS.

10.2.12 TWIM

TWIM SR.IDLE goes high immediately when NAK is received

When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.

Fix/Workaround

If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

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.

11.7 Rev. B – 08/09

1. Updated the datasheet with new device AT32UC3A4.

11.8 Rev. A – 03/09

1. Initial revision.