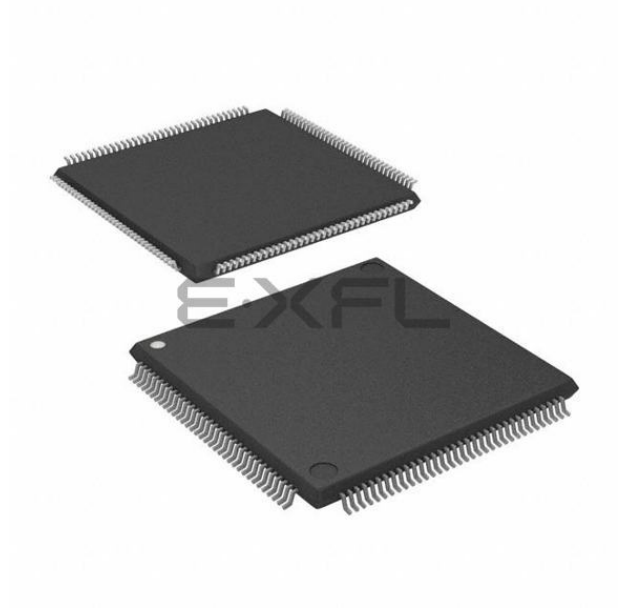


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

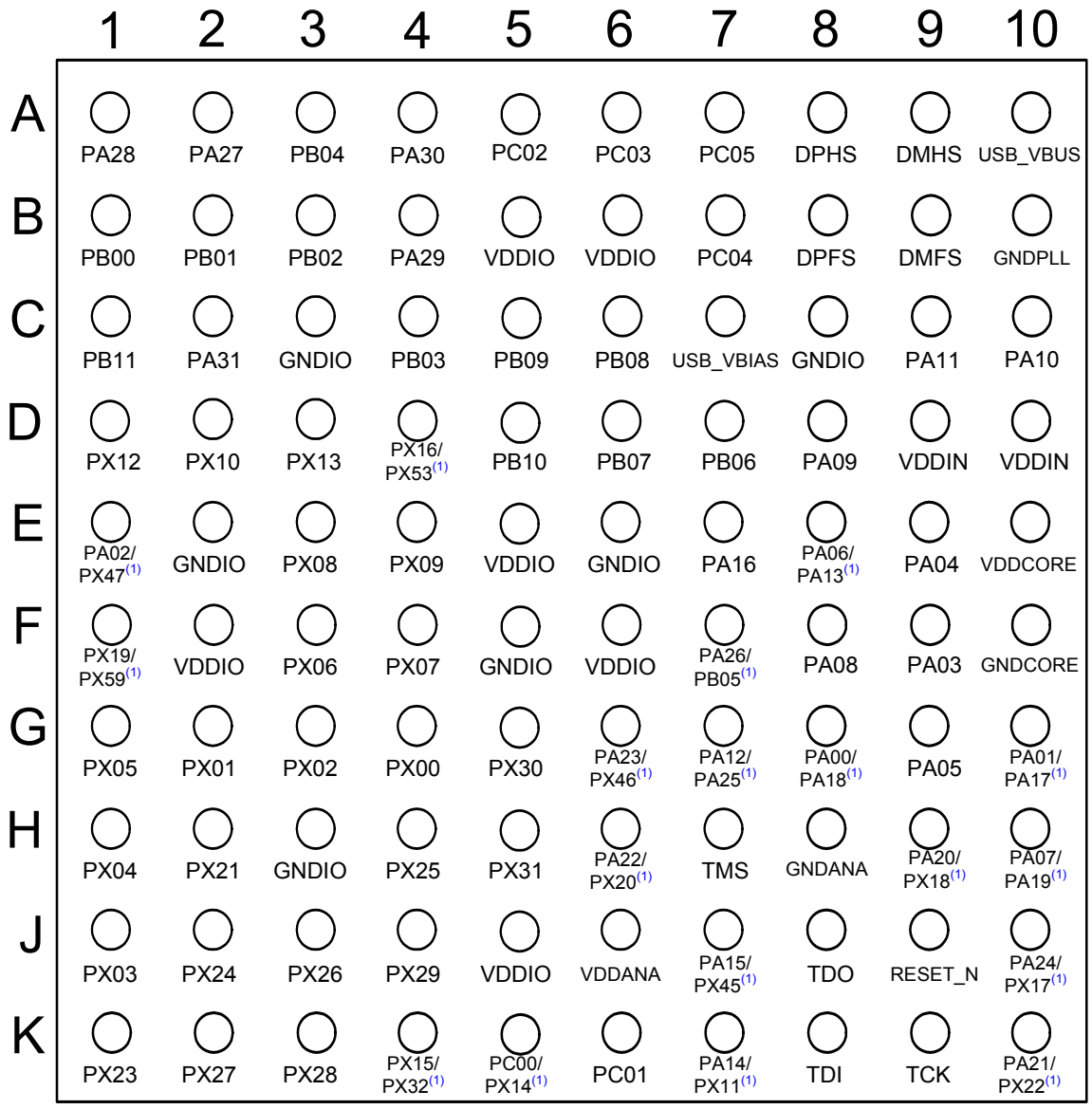
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB OTG
Peripherals	Brown-out Detect/Reset, DMA, POR, WDT
Number of I/O	110
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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3a3256s-alut

  - Support for SPI and LIN
  - Optionnal support for IrDA, ISO7816, Hardware Handshaking, RS485 interfaces and Modem Line
- Two Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
- One Synchronous Serial Protocol Controller
  - Supports I2S and Generic Frame-Based Protocols
- Two Master/Slave Two-Wire Interface (TWI), 400 kbit/s I2C-compatible
- 16-bit Stereo Audio Bitstream
  - Sample Rate Up to 50 KHz
- QTouch® Library Support
  - Capacitive Touch Buttons, Sliders, and Wheels
  - QTouch and QMatrix Acquisition
- On-Chip Debug System (JTAG interface)
  - Nexus Class 2+, Runtime Control, Non-Intrusive Data and Program Trace
- 110 General Purpose Input/Output (GPIOs)
  - Standard or High Speed mode
  - Toggle capability: up to 84 MHz
- Packages
  - 144-ball TFBGA, 11x11 mm, pitch 0.8 mm
  - 144-pin LQFP, 22x22 mm, pitch 0.5 mm
  - 100-ball VFBGA, 7x7 mm, pitch 0.65 mm
- Single 3.3 V Power Supply

Figure 3-3. VFBGA100 Pinout (top view)



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	GPIO	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 carrefully 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 Mananger (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 carrefully the GPIO configuration to avoid electrical conflict

3.2.4 JTAG port connections

Table 3-4. JTAG Pinout

TFBGA144	QFP144	VFBGA100	Pin name	JTAG pin
K12	107	K9	TCK	TCK
L12	108	K8	TDI	TDI
J11	105	J8	TDO	TDO
J10	104	H7	TMS	TMS

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, irrespective of the GPIO configuration. Three differents OCD trace pin mappings are possible, depending on the configuration of the OCD AXS register. For details, see the AVR32 UC Technical Reference Manual.

Table 3-5. Nexus OCD AUX port connections

Pin	AXS=0	AXS=1	AXS=2
EVTI_N	PB05	PA08	PX00
MDO[5]	PA00	PX56	PX06
MDO[4]	PA01	PX57	PX05
MDO[3]	PA03	PX58	PX04
MDO[2]	PA16	PA24	PX03
MDO[1]	PA13	PA23	PX02
MDO[0]	PA12	PA22	PX01
MSEO[1]	PA10	PA07	PX08
MSEO[0]	PA11	PX55	PX07
MCKO	PB07	PX00	PB09
EVTO_N	PB06	PB06	PB06

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		

4. Processor and Architecture

Rev: 1.4.2.0

This chapter gives an overview of the AVR32UC CPU. AVR32UC is an implementation of the AVR32 architecture. A summary of the programming model, instruction set, and MPU is presented. For further details, see the *AVR32 Architecture Manual* and the *AVR32UC Technical Reference Manual*.

4.1 Features

- **32-bit load/store AVR32A RISC architecture**
  - **15 general-purpose 32-bit registers**
  - **32-bit Stack Pointer, Program Counter and Link Register reside in register file**
  - **Fully orthogonal instruction set**
  - **Privileged and unprivileged modes enabling efficient and secure Operating Systems**
  - **Innovative instruction set together with variable instruction length ensuring industry leading code density**
  - **DSP extention with saturating arithmetic, and a wide variety of multiply instructions**
- **3-stage pipeline allows one instruction per clock cycle for most instructions**
  - **Byte, halfword, word and double word memory access**
  - **Multiple interrupt priority levels**
- **MPU allows for operating systems with memory protection**

4.2 AVR32 Architecture

AVR32 is a high-performance 32-bit RISC microprocessor architecture, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption and high code density. In addition, the instruction set architecture has been tuned to allow a variety of micro-architectures, enabling the AVR32 to be implemented as low-, mid-, or high-performance processors. AVR32 extends the AVR family into the world of 32- and 64-bit applications.

Through a quantitative approach, a large set of industry recognized benchmarks has been compiled and analyzed to achieve the best code density in its class. In addition to lowering the memory requirements, a compact code size also contributes to the core's low power characteristics. The processor supports byte and halfword data types without penalty in code size and performance.

Memory load and store operations are provided for byte, halfword, word, and double word data with automatic sign- or zero extension of halfword and byte data. The C-compiler is closely linked to the architecture and is able to exploit code optimization features, both for size and speed.

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

Another feature of the instruction set is that frequently used instructions, like add, have a compact format with two operands as well as an extended format with three operands. The larger format increases performance, allowing an addition and a data move in the same instruction in a single cycle. Load and store instructions have several different formats in order to reduce code size and speed up execution.

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	JECR	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

7. Electrical Characteristics

7.1 Absolute Maximum Ratings*

Operating Temperature	-40°C to +85°C
Storage Temperature	-60°C to +150°C
Voltage on Input Pin with respect to Ground	-0.3V to 3.6V
Maximum Operating Voltage (VDDCORE)	1.95V
Maximum Operating Voltage (VDDIO)	3.6V
Total DC Output Current on all I/O Pin	
for TQFP144 package	370 mA
for TFBGA144 package	370 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.

Table 7-2. Normal I/O Pin Characteristics

Symbol	Parameter	Conditions	drive x2	drive x2	drive x3	Unit
f_{MAX}	Output frequency	10pf	40	66	100	MHz
		30pf	18.2	35.7	61.6	MHz
		60pf	7.5	18.5	36.3	MHz
t_{RISE}	Rise time	10pf	2.7	1.4	0.9	ns
		30pf	6.9	3.5	1.9	ns
		60pf	13.4	6.7	3.5	ns
t_{FALL}	Fall time	10pf	3.2	1.7	0.9	ns
		30pf	8.6	4.3	2.26	ns
		60pf	16.5	8.3	4.3	ns

7.4 Regulator characteristics

Table 7-3. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{VDDIN}	Supply voltage (input)		3.0	3.3	3.6	V
$V_{VDDCORE}$	Supply voltage (output)		1.75	1.85	1.95	V
I_{OUT}	Maximum DC output current	V_{VDDIN} = 3.3V			100	mA

Table 7-4. Decoupling Requirements

Symbol	Parameter	Conditions	Typ.	Technology	Unit
C_{IN1}	Input Regulator Capacitor 1		1	NPO	nF
C_{IN2}	Input Regulator Capacitor 2		4.7	X7R	µF
C_{OUT1}	Output Regulator Capacitor 1		470	NPO	pF
C_{OUT2}	Output Regulator Capacitor 2		2.2	X7R	µF

7.9 ADC Characteristics

Table 7-22. Channel Conversion Time and ADC Clock

Parameter	Conditions	Min.	Typ.	Max.	Unit
ADC Clock Frequency	10-bit resolution mode			5	MHz
	8-bit resolution mode			8	MHz
Startup Time	Return from Idle Mode			20	µs
Track and Hold Acquisition Time		600			ns
Conversion Time	ADC Clock = 5 MHz			2	µs
	ADC Clock = 8 MHz			1.25	µs
Throughput Rate	ADC Clock = 5 MHz			384 (1)	kSPS
	ADC Clock = 8 MHz			533 (2)	kSPS

1. Corresponds to 13 clock cycles: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.
2. Corresponds to 15 clock cycles: 5 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

Table 7-23. ADC Power Consumption

Parameter	Conditions	Min.	Typ.	Max.	Unit
Current Consumption on VDDANA (1)	On 13 samples with ADC clock = 5 MHz			1.25	mA

1. Including internal reference input current

Table 7-24. Analog Inputs

Parameter	Conditions	Min.	Typ.	Max.	Unit
Input Voltage Range		0		VDDANA	V
Input Leakage Current				1	µA
Input Capacitance			7		pF
Input Resistance			350	850	Ohm

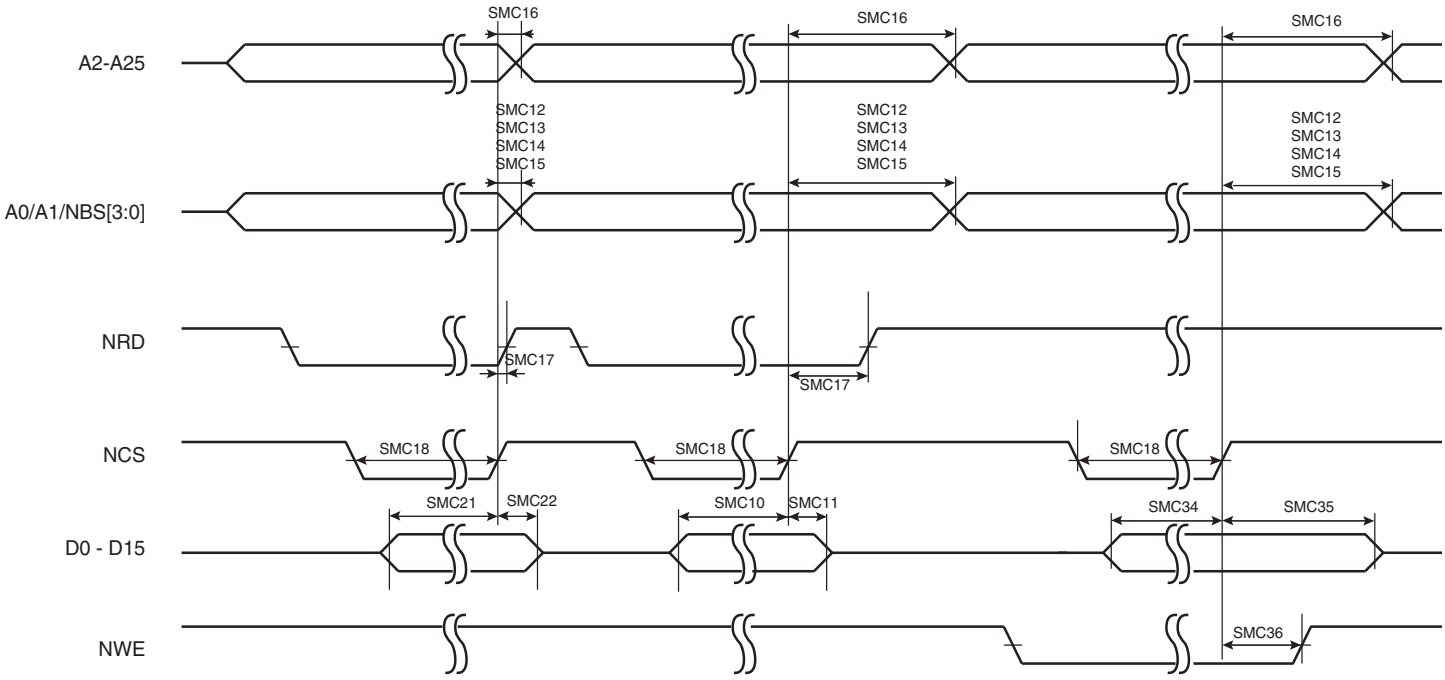
Table 7-25. Transfer Characteristics in 8-bit mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Resolution			8		Bit
Absolute Accuracy	ADC Clock = 5 MHz			0.8	LSB
	ADC Clock = 8 MHz			1.5	LSB
Integral Non-linearity	ADC Clock = 5 MHz		0.35	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.5	LSB
Differential Non-linearity	ADC Clock = 5 MHz		0.3	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.5	LSB
Offset Error	ADC Clock = 5 MHz	-1.5		1.5	LSB
Gain Error	ADC Clock = 5 MHz	-0.5		0.5	LSB

Table 7-34. SMC Write Signals with No Hold Settings (NWE Controlled only)

Symbol	Parameter	Min.	Unit
SMC_{43}	Data Out Valid before NWE Rising	(nwe pulse length - 1) * t_{CPSMC} - 1.2	ns
SMC_{44}	Data Out Valid after NWE Rising	5	ns
SMC_{45}	NWE Pulse Width	nwe pulse length * t_{CPSMC} - 0.9	ns

Figure 7-7. SMC Signals for NCS Controlled Accesses.



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:

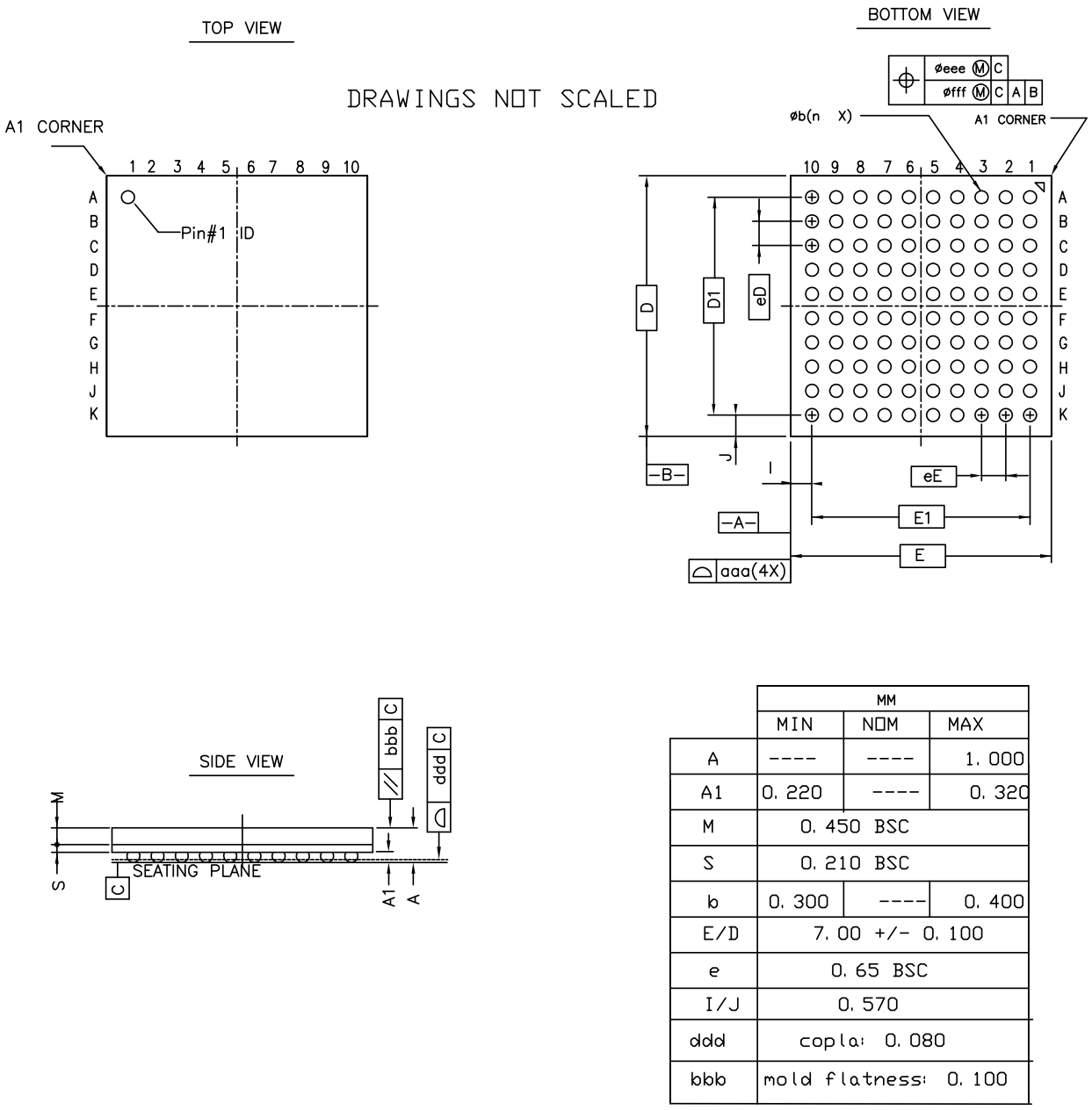
1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

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.

Figure 8-3. VFBGA-100 package drawing



	MM		
	MIN	NOM	MAX
A	----	----	1.000
A1	0.220	----	0.320
M	0.450 BSC		
S	0.210 BSC		
b	0.300	----	0.400
E/D	7.00 +/- 0.100		
e	0.65 BSC		
I/J	0.570		
ddd	copla: 0.080		
bbb	mold flatness: 0.100		

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.

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.

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.

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

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.

Multiply instructions do not work on RevD
All the multiply instructions do not work.
Fix/Workaround
Do not use the multiply instructions.

10.3.3 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.

10.3.4 USB

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.

10.3.5 ADC

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.

10.3.6 USART

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

The LIN ID is not transmitted in mode PDCM='0'
Fix/Workaround
Using USART in mode LIN master with the PDCM bit = '0', the LINID written at the first address of the transmit buffer is not used. The LINID must be written in the LINIR register, after the configuration and start of the PDCA transfer. Writing the LINID in the LINIR register will start the transfer whenever the PDCA transfer is ready.

The LINID interrupt is only available for the header reception and not available for the header transmission
Fix/Workaround
None.

USART LIN mode is not functional with the PDCA if PDCM bit in LINMR register is set to 1
If a PDCA transfer is initiated in USART LIN mode with PDCM bit set to 1, the transfer never starts.
Fix/Workaround
Only use PDCM=0 configuration with the PDCA transfer.

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.

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.

Fix/Workaround
None.

10.3.14 MCI

The busy signal of the responses R1b is not taken in account for CMD12 STOP_TRANSFER
It is not possible to know the busy status of the card during the response (R1b) for the commands CMD12.
Fix/Workaround
The card busy line should be polled through the GPIO Input Value register (IVR) for commands CMD12.

10.3.15 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.

10.3.16 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.

11. Datasheet Revision History

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

11.1 Rev. H– 10/12

1. Updated max frequency
2. Added Flash Read High Speed Mode description in FLASHC chapter
3. Updated Electrical Characteristics accordingly to new max frequency
4. Fixed wrong description of PLLOPT[0] in PM chapter
5. Updated Errata section according to new maximum frequency
6. Added USB hi-speed PLL electrical characteristics
7. Added OSC32 Errata in Power Management sections for Rev D,E and H

11.2 Rev. G– 11/11

1. Add recommandation for MCI connection with more than 1 slot

11.3 Rev. F – 08/11

1. Final version

11.4 Rev. E – 06/11

1. Updated Errata for E and D
2. Updated FLASHC chapter with HSEN and HSDIS commands

11.5 Rev. D – 04/11

1. Updated Errata for revision H and E
2. Updated Reset Sequence
3. Updated Peripherals' current consumption and others minor electrical charateristics
4. Updated Peripherals chapters

11.6 Rev. C – 03/10

1. Updated the datasheet with new revision H features.

11.7 Rev. B – 08/09

1. Updated the datasheet with new device AT32UC3A4.

11.8 Rev. A – 03/09

1. Initial revision.

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

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

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

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