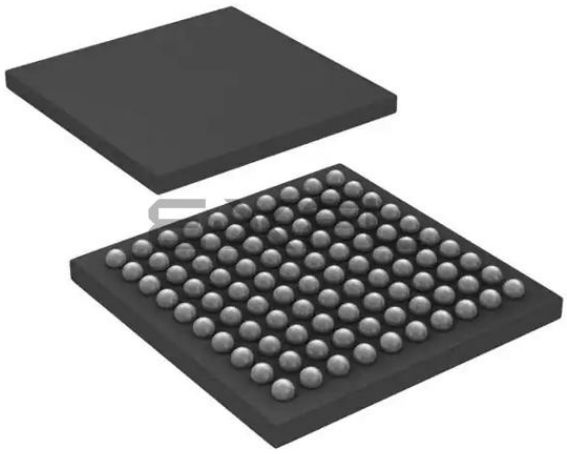


Welcome to E-XFL.COM

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.

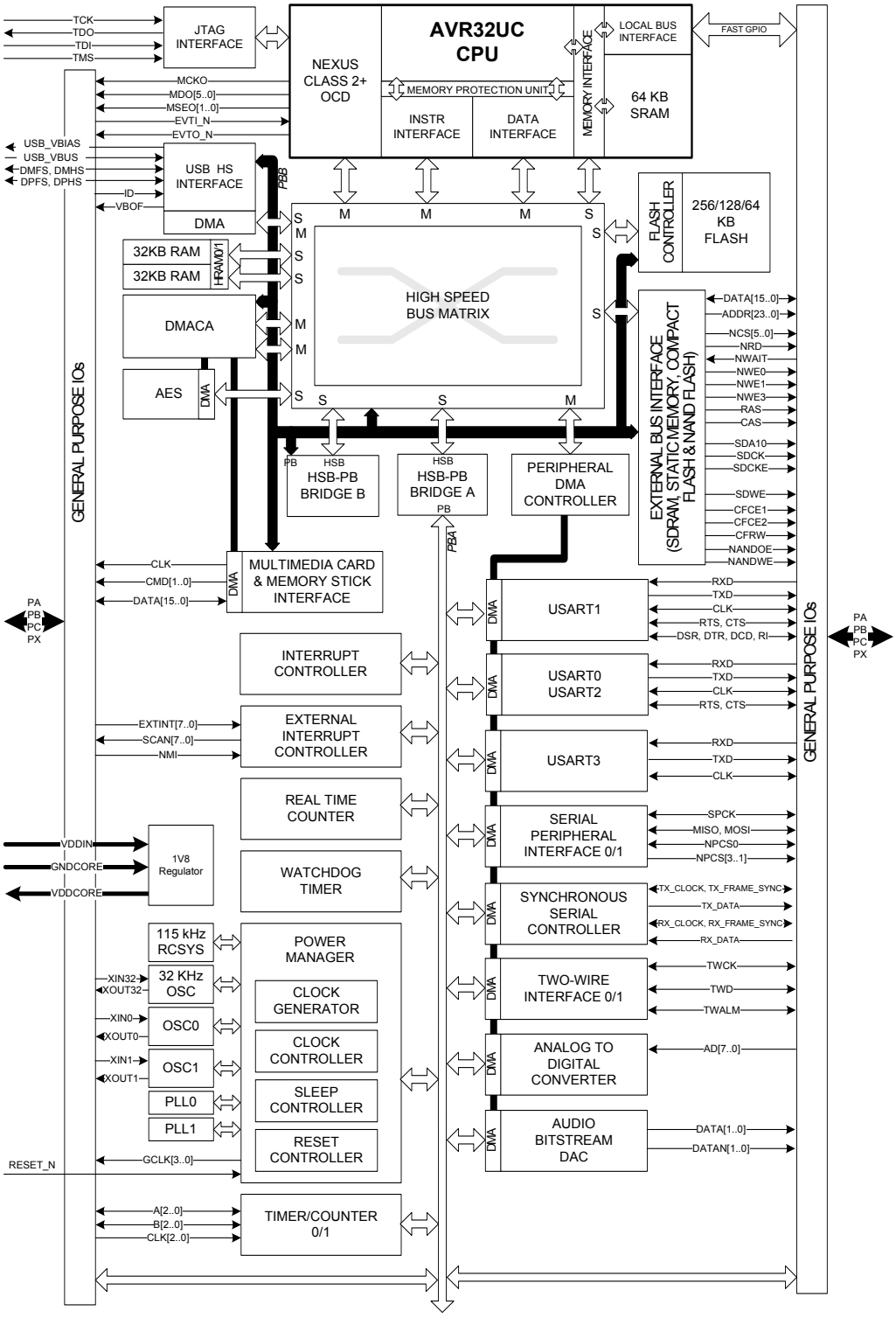
Applications of "Embedded - Microcontrollers"

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	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-c1ur

2. Overview

2.1 Block Diagram

Figure 2-1. Block Diagram



3. Package and Pinout

3.1 Package

The device pins are multiplexed with peripheral functions as described in the Peripheral Multiplexing on I/O Line section.

Figure 3-1. TFBGA144 Pinout (top view)

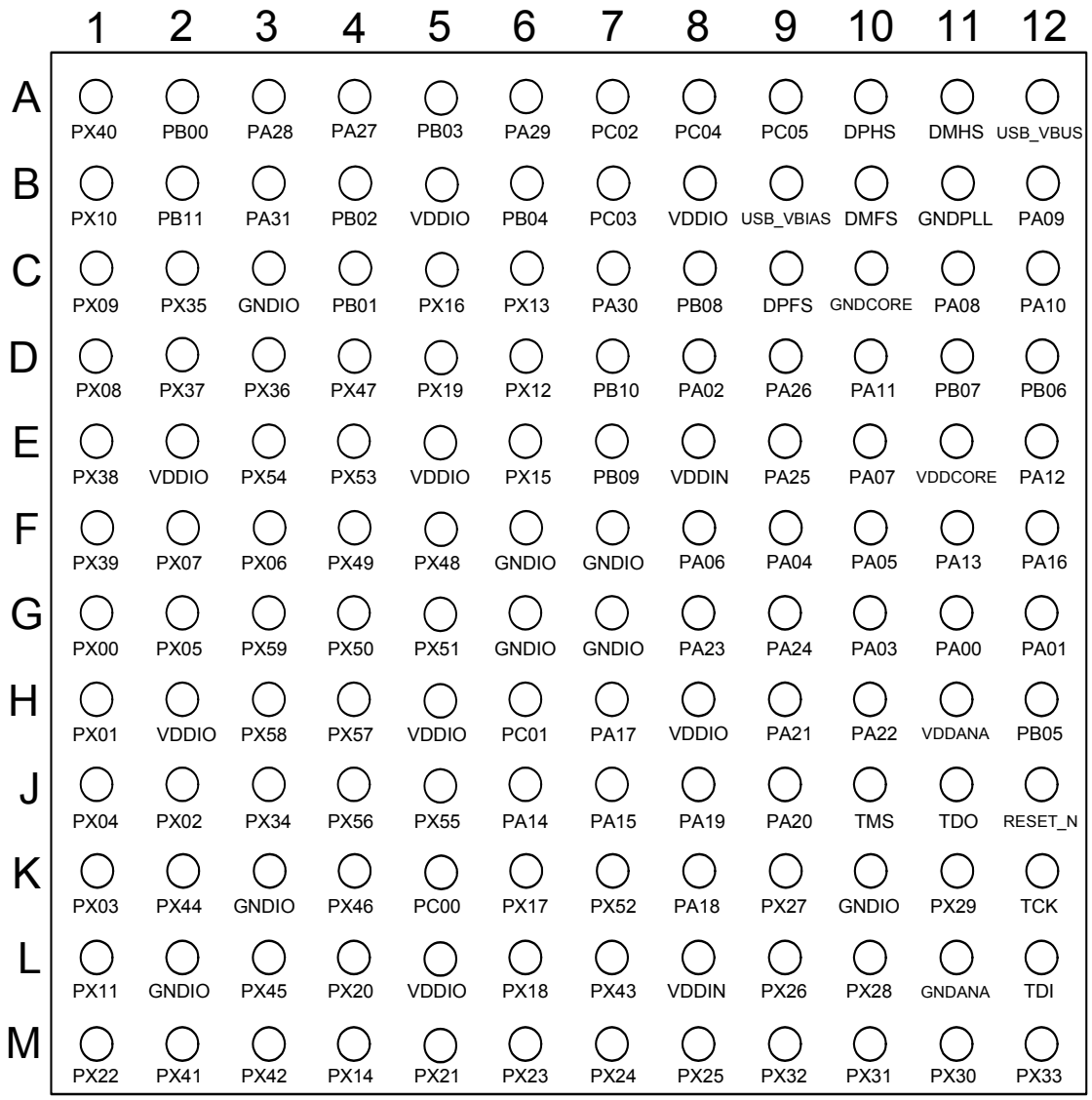


Table 3-6. Signal Description List

Signal Name	Function	Type	Active Level	Comments
DMHS	USB High Speed Data -	Analog		
DPHS	USB High Speed Data +	Analog		
USB_VBIAS	USB VBIAS reference	Analog		Connect to the ground through a 6810 ohms (+/- 1%) resistor in parallel with a 10pf capacitor. If USB hi-speed feature is not required, leave this pin unconnected to save power
USB_VBUS	USB VBUS signal	Output		
VBOF	USB VBUS on/off bus power control port	Output		
ID	ID Pin fo the USB bus	Input		

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.

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

4.3.6 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

4.3.7 CPU and Architecture Revision

Three major revisions of the AVR32UC CPU currently exist.

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.

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

Table 5-1. AT32UC3A3A4 Physical Memory Map

Device	Start Address	Size AT32UC3A3256S AT32UC3A3256 AT32UC3A4256S AT32UC3A4256	Size AT32UC3A3128S AT32UC3A3128 AT32UC3A4128S AT32UC3A4128	Size AT32UC3A364S AT32UC3A364 AT32UC3A464S AT32UC3A464
HRAMC0	0xFF000000	32KByte	32KByte	32KByte
HRAMC1	0xFF008000	32KByte	32KByte	32KByte
HSB-PB Bridge A	0xFFFF0000	64KByte	64KByte	64KByte
HSB-PB Bridge B	0xFFFE0000	64KByte	64KByte	64KByte

5.3 Peripheral Address Map

Table 5-2. Peripheral Address Mapping

Address		Peripheral Name
0xFF100000	DMACA	DMA Controller - DMACA
0xFFFD0000	AES	Advanced Encryption Standard - AES
0xFFFE0000	USB	USB 2.0 Device and Host Interface - USB
0xFFFE1000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE1400	FLASHC	Flash Controller - FLASHC
0xFFFE1C00	SMC	Static Memory Controller - SMC
0xFFFE2000	SDRAMC	SDRAM Controller - SDRAMC
0xFFFE2400	ECCHRS	Error code corrector Hamming and Reed Solomon - ECCHRS
0xFFFE2800	BUSMON	Bus Monitor module - BUSMON
0xFFFE4000	MCI	Mulitmedia Card Interface - MCI
0xFFFE8000	MSI	Memory Stick Interface - MSI
0xFFFF0000	PDCA	Peripheral DMA Controller - PDCA
0xFFFF0800	INTC	Interrupt controller - INTC

Table 5-2. Peripheral Address Mapping

0xFFFF0C00	PM	Power Manager - PM
0xFFFF0D00	RTC	Real Time Counter - RTC
0xFFFF0D30	WDT	Watchdog Timer - WDT
0xFFFF0D80	EIC	External Interrupt Controller - EIC
0xFFFF1000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF1400	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0
0xFFFF1800	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1
0xFFFF1C00	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2
0xFFFF2000	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter - USART3
0xFFFF2400	SPI0	Serial Peripheral Interface - SPI0
0xFFFF2800	SPI1	Serial Peripheral Interface - SPI1
0xFFFF2C00	TWIM0	Two-wire Master Interface - TWIM0
0xFFFF3000	TWIM1	Two-wire Master Interface - TWIM1
0xFFFF3400	SSC	Synchronous Serial Controller - SSC
0xFFFF3800	TC0	Timer/Counter - TC0
0xFFFF3C00	ADC	Analog to Digital Converter - ADC
0xFFFF4000	ABDAC	Audio Bitstream DAC - ABDAC
0xFFFF4400	TC1	Timer/Counter - TC1

Table 5-2. Peripheral Address Mapping

0xFFFF5000	TWIS0	Two-wire Slave Interface - TWIS0
0xFFFF5400	TWIS1	Two-wire Slave Interface - TWIS1

5.4 CPU Local Bus Mapping

Some of the registers in the GPIO module are mapped onto the CPU local bus, in addition to being mapped on the Peripheral Bus. These registers can therefore be reached both by accesses on the Peripheral Bus, and by accesses on the local bus.

Mapping these registers on the local bus allows cycle-deterministic toggling of GPIO pins since the CPU and GPIO are the only modules connected to this bus. Also, since the local bus runs at CPU speed, one write or read operation can be performed per clock cycle to the local bus-mapped GPIO registers.

The following GPIO registers are mapped on the local bus:

Table 5-3. Local Bus Mapped GPIO Registers

Port	Register	Mode	Local Bus Address	Access
0	Output Driver Enable Register (ODER)	WRITE	0x40000040	Write-only
		SET	0x40000044	Write-only
		CLEAR	0x40000048	Write-only
		TOGGLE	0x4000004C	Write-only
	Output Value Register (OVR)	WRITE	0x40000050	Write-only
		SET	0x40000054	Write-only
		CLEAR	0x40000058	Write-only
		TOGGLE	0x4000005C	Write-only
	Pin Value Register (PVR)	-	0x40000060	Read-only
1	Output Driver Enable Register (ODER)	WRITE	0x40000140	Write-only
		SET	0x40000144	Write-only
		CLEAR	0x40000148	Write-only
		TOGGLE	0x4000014C	Write-only
	Output Value Register (OVR)	WRITE	0x40000150	Write-only
		SET	0x40000154	Write-only
		CLEAR	0x40000158	Write-only
		TOGGLE	0x4000015C	Write-only
	Pin Value Register (PVR)	-	0x40000160	Read-only

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. 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

Table 7-29. Dynamic Power Consumption

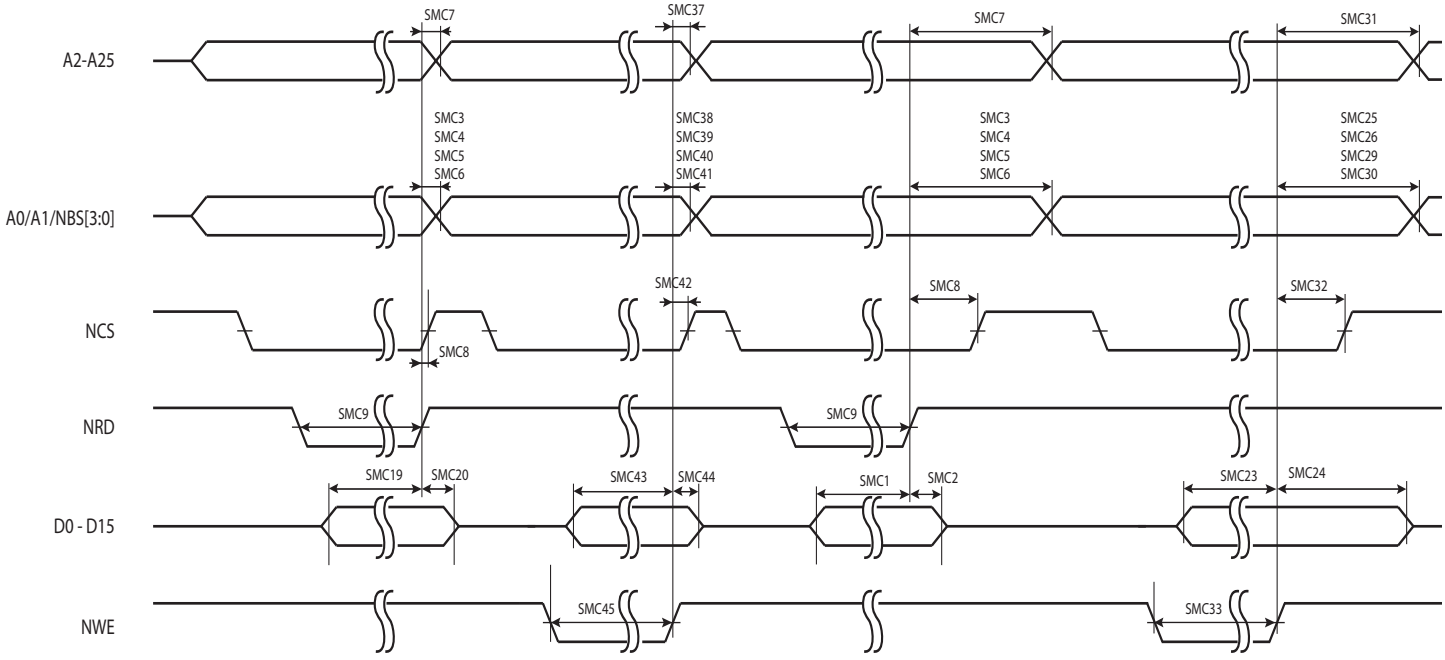
Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$I_{VDDUTMI}$	HS Transceiver current consumption	HS transmission		47	60	mA
	HS Transceiver current consumption	HS reception		18	27	mA
	FS/HS Transceiver current consumption	FS transmission 0m cable (1)		4	6	mA
	FS/HS Transceiver current consumption	FS transmission 5m cable		26	30	mA
	FS/HS Transceiver current consumption	FS reception		3	4.5	mA

1. Including 1 mA due to Pull-up/Pull-down current consumption.

34.5.5 USB High Speed Design Guidelines

In order to facilitate hardware design, Atmel provides an application note on www.atmel.com.

Figure 7-8. SMC Signals for NRD and NRW Controlled Accesses.



7.11.2 SDRAM Signals

These timings are given for 10 pF load on SDCK and 40 pF on other signals.

Table 7-35. SDRAM Clock Signal.

Symbol	Parameter	Conditions	Min.	Max.(1)	Unit
$1/(t_{CPSDCK})$	SDRAM Controller Clock Frequency			$1/(t_{cpcpu})$	MHz

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

Table 7-36. SDRAM Clock Signal

Symbol	Parameter	Conditions	Min.	Max.	Unit
$SDRAMC_1$	SDCKE High before SDCK Rising Edge		7.4		ns
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge		3.2		ns
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge		7		ns
$SDRAMC_4$	SDCKE High after SDCK Rising Edge		2.9		ns
$SDRAMC_5$	SDCS Low before SDCK Rising Edge		7.5		ns
$SDRAMC_6$	SDCS High after SDCK Rising Edge		1.6		ns
$SDRAMC_7$	RAS Low before SDCK Rising Edge		7.2		ns
$SDRAMC_8$	RAS High after SDCK Rising Edge		2.3		ns
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge		7.6		ns
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge		1.9		ns
$SDRAMC_{11}$	Address Change before SDCK Rising Edge		6.2		ns
$SDRAMC_{12}$	Address Change after SDCK Rising Edge		2.2		ns

7.12 JTAG Characteristics

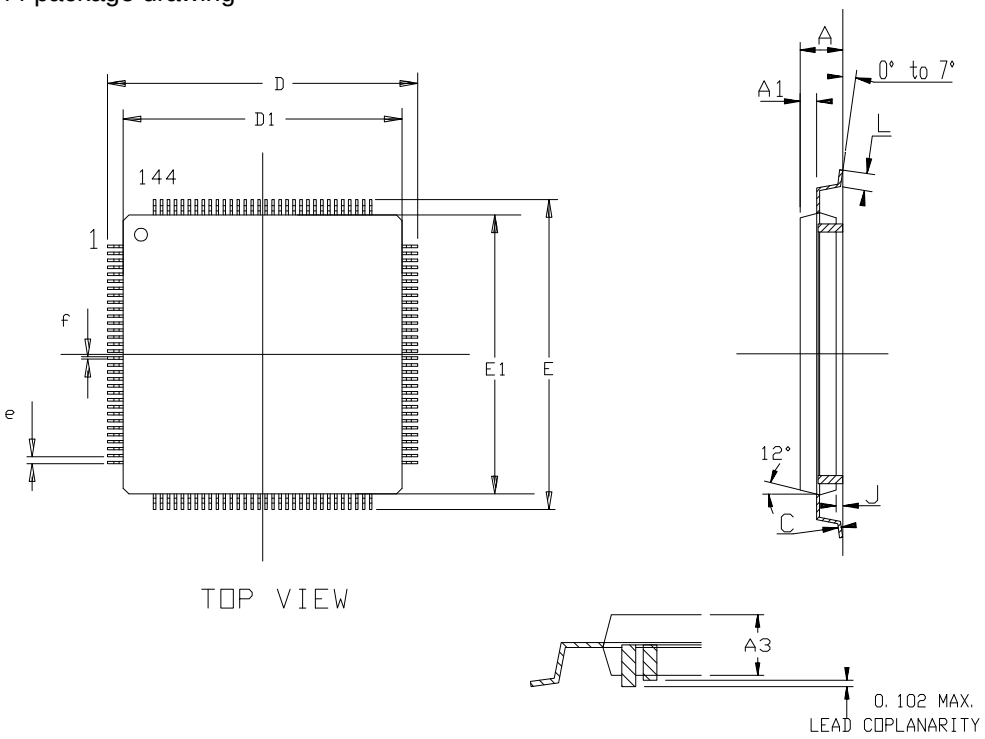
7.12.1 JTAG Interface Signals

Table 7-37. JTAG Interface Timing Specification

Symbol	Parameter	Conditions [1]	Min.	Max.	Unit
$JTAG_0$	TCK Low Half-period		6		ns
$JTAG_1$	TCK High Half-period		3		ns
$JTAG_2$	TCK Period		9		ns
$JTAG_3$	TDI, TMS Setup before TCK High		1		ns
$JTAG_4$	TDI, TMS Hold after TCK High		0		ns
$JTAG_5$	TDO Hold Time		4		ns
$JTAG_6$	TCK Low to TDO Valid			6	ns
$JTAG_7$	Device Inputs Setup Time				ns
$JTAG_8$	Device Inputs Hold Time				ns
$JTAG_9$	Device Outputs Hold Time				ns
$JTAG_{10}$	TCK to Device Outputs Valid				ns

1. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF

Figure 8-2. LQFP-144 package drawing



	MM			INCH		
	Min	Nom	Max	Min	Nom	Max
A	–	–	1.60	–	–	.063
C	0.09	–	0.20	.004	–	.008
A3	1.35	1.40	1.45	.053	.055	.057
D	21.90	22.00	22.10	.862	.866	.870
D1	19.90	20.00	20.10	.783	.787	.791
E	21.90	22.00	22.10	.862	.866	.870
E1	19.90	20.00	20.10	.783	.787	.791
J	0.05	–	0.15	.002	–	.006
L	0.45	0.60	0.75	.018	.024	.030
e	0.50 BSC			.0197 BSC		
f	0.22 BSC			.009 BSC		

Table 8-2. Device and Package Maximum Weight

1300	mg

Table 8-3. Package Characteristics

Moisture Sensitivity Level	MSL3

Table 8-4. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

10. Errata

10.1 Rev. H

10.1.1 General

Devices with Date Code lower than 1233 cannot operate with CPU frequency higher than 66MHz in 1WS and 36MHz in 0WS in the whole temperature range
Fix/Workaround
None

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.

10.1.2 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.

10.1.3 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.

10.1.4 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

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.

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.

10.1.8 Power Manager

OSC32 not functionnal in Crystal Modes (OSC32CTRL.MODE=1 or OSC32CTRL.MODE=2)
OSC32 clock output is not active even if the oscillation signal is present on XIN32/XOUT32 pins.
OSC32RDY bit may still set even if the CLK32 is not active.
External clock mode (OSC32CTRL.MODE=0) is not affected.

Fix/Workaround
None.

Clock sources will not be stopped in STATIC sleep mode if the difference between CPU and PBx division factor is too high
If the division factor between the CPU/HSB and PBx frequencies is more than 4 when going to a sleep mode where the system RC oscillator is turned off, then high speed clock sources will not be turned off. This will result in a significantly higher power consumption during the sleep mode.
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.

10.1.9 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.

10.1.10 AES

URAD (Unspecified Register Access Detection Status) does not detect read accesses to the write-only KEYW[5..8]R registers
Fix/Workaround
None.

10.1.11 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.

10.1.12 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.

SMBALERT bit may be set after reset
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.

10.1.13 TWIS

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.

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.