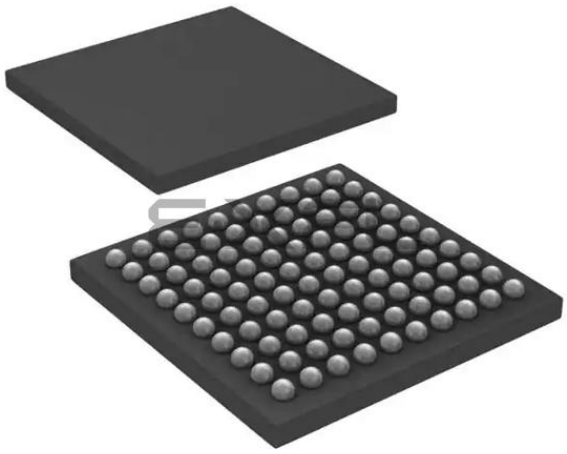


Welcome to E-XFL.COM

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	64KB (64K 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/at32uc3a464s-c1ut

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
L6	84	H9(1)	PX18	69	VDDIO	x2	EBI - ADDR[16]	DMACA - DMAACK[1]	TC0 - A2	
D5	35	F1(1)	PX19	70	VDDIO	x2	EBI - ADDR[15]	EIC - SCAN[0]	TC0 - B2	
L4	73	H6(1)	PX20	71	VDDIO	x2	EBI - ADDR[14]	EIC - SCAN[1]	TC0 - CLK0	
M5	80	H2	PX21	72	VDDIO	x2	EBI - ADDR[13]	EIC - SCAN[2]	TC0 - CLK1	
M1	72	K10(1)	PX22	73	VDDIO	x2	EBI - ADDR[12]	EIC - SCAN[3]	TC0 - CLK2	
M6	85	K1	PX23	74	VDDIO	x2	EBI - ADDR[11]	EIC - SCAN[4]	SSC - TX_CLOCK	
M7	86	J2	PX24	75	VDDIO	x2	EBI - ADDR[10]	EIC - SCAN[5]	SSC - TX_DATA	
M8	92	H4	PX25	76	VDDIO	x2	EBI - ADDR[9]	EIC - SCAN[6]	SSC - RX_DATA	
L9	90	J3	PX26	77	VDDIO	x2	EBI - ADDR[8]	EIC - SCAN[7]	SSC - RX_FRAME_SYNC	
K9	89	K2	PX27	78	VDDIO	x2	EBI - ADDR[7]	SPI0 - MISO	SSC - TX_FRAME_SYNC	
L10	91	K3	PX28	79	VDDIO	x2	EBI - ADDR[6]	SPI0 - MOSI	SSC - RX_CLOCK	
K11	94	J4	PX29	80	VDDIO	x2	EBI - ADDR[5]	SPI0 - SPCK		
M11	96	G5	PX30	81	VDDIO	x2	EBI - ADDR[4]	SPI0 - NPCS[0]		
M10	97	H5	PX31	82	VDDIO	x2	EBI - ADDR[3]	SPI0 - NPCS[1]		
M9	93	K4(1)	PX32	83	VDDIO	x2	EBI - ADDR[2]	SPI0 - NPCS[2]		
M12	95		PX33	84	VDDIO	x2	EBI - ADDR[1]	SPI0 - NPCS[3]		
J3	61		PX34	85	VDDIO	x2	EBI - ADDR[0]	SPI1 - MISO	PM - GCLK[0]	
C2	38		PX35	86	VDDIO	x2	EBI - DATA[15]	SPI1 - MOSI	PM - GCLK[1]	
D3	44		PX36	87	VDDIO	x2	EBI - DATA[14]	SPI1 - SPCK	PM - GCLK[2]	
D2	45		PX37	88	VDDIO	x2	EBI - DATA[13]	SPI1 - NPCS[0]	PM - GCLK[3]	
E1	51		PX38	89	VDDIO	x2	EBI - DATA[12]	SPI1 - NPCS[1]	USART1 - DCD	
F1	52		PX39	90	VDDIO	x2	EBI - DATA[11]	SPI1 - NPCS[2]	USART1 - DSR	
A1	36		PX40	91	VDDIO	x2		MCI - CLK		
M2	71		PX41	92	VDDIO	x2	EBI - CAS			
M3	69		PX42	93	VDDIO	x2	EBI - RAS			
L7	88		PX43	94	VDDIO	x2	EBI - SDA10	USART1 - RI		
K2	66		PX44	95	VDDIO	x2	EBI - SDWE	USART1 - DTR		
L3	70	J7(1)	PX45	96	VDDIO	x3	EBI - SDCK			
K4	74	G6(1)	PX46	97	VDDIO	x2	EBI - SDCKE			
D4	39	E1(1)	PX47	98	VDDIO	x2	EBI - NANDOE	ADC - TRIGGER	MCI - DATA[11]	
F5	41		PX48	99	VDDIO	x2	EBI - ADDR[23]	USB - VBOF	MCI - DATA[10]	
F4	43		PX49	100	VDDIO	x2	EBI - CFRNW	USB - ID	MCI - DATA[9]	
G4	75		PX50	101	VDDIO	x2	EBI - CFCE2	TC1 - B2	MCI - DATA[8]	
G5	77		PX51	102	VDDIO	x2	EBI - CFCE1	DMACA - DMAACK[0]	MCI - DATA[15]	
K7	87		PX52	103	VDDIO	x2	EBI - NCS[3]	DMACA - DMARQ[0]	MCI - DATA[14]	
E4	42	D4(1)	PX53	104	VDDIO	x2	EBI - NCS[2]		MCI - DATA[13]	
E3	46		PX54	105	VDDIO	x2	EBI - NWAIT	USART3 - TXD	MCI - DATA[12]	
J5	79		PX55	106	VDDIO	x2	EBI - ADDR[22]	EIC - SCAN[3]	USART2 - RXD	

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

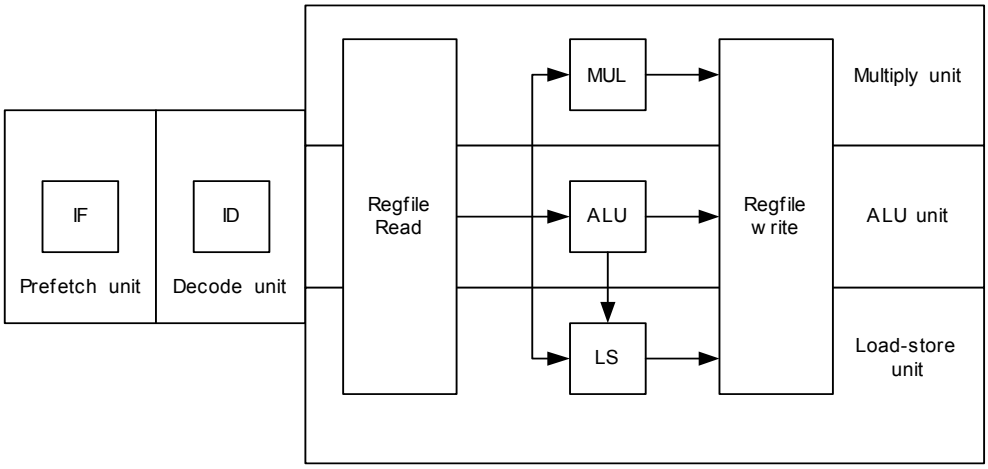
The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced OCD system, no caches, and a Memory Protection Unit (MPU). Java acceleration hardware is not implemented.

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.

4.4 Programming Model

4.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 4-3. The AVR32UC Register File

Application	Supervisor	INT0	INT1	INT2	INT3	Exception	NMI	Secure
Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0
PC	PC	PC	PC	PC	PC	PC	PC	PC
LR	LR	LR	LR	LR	LR	LR	LR	LR
SP_APP	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SEC
R12	R12	R12	R12	R12	R12	R12	R12	R12
R11	R11	R11	R11	R11	R11	R11	R11	R11
R10	R10	R10	R10	R10	R10	R10	R10	R10
R9	R9	R9	R9	R9	R9	R9	R9	R9
R8	R8	R8	R8	R8	R8	R8	R8	R8
R7	R7	R7	R7	R7	R7	R7	R7	R7
R6	R6	R6	R6	R6	R6	R6	R6	R6
R5	R5	R5	R5	R5	R5	R5	R5	R5
R4	R4	R4	R4	R4	R4	R4	R4	R4
R3	R3	R3	R3	R3	R3	R3	R3	R3
R2	R2	R2	R2	R2	R2	R2	R2	R2
R1	R1	R1	R1	R1	R1	R1	R1	R1
R0	R0	R0	R0	R0	R0	R0	R0	R0
SR	SR	SR	SR	SR	SR	SR	SR	SR

SS_STATUS
SS_ADRF
SS_ADRR
SS_ADR0
SS_ADR1
SS_SP_SYS
SS_SP_APP
SS_RAR
SS_RSR

4.4.2 Status Register Configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 4-4 on page 26 and Figure 4-5 on page 27. The lower word contains the C, Z, N, V, and Q condition code flags and the R, T, and L bits, while the upper halfword contains information about the mode and state the processor executes in. Refer to the *AVR32 Architecture Manual* for details.

Figure 4-4. The Status Register High Halfword

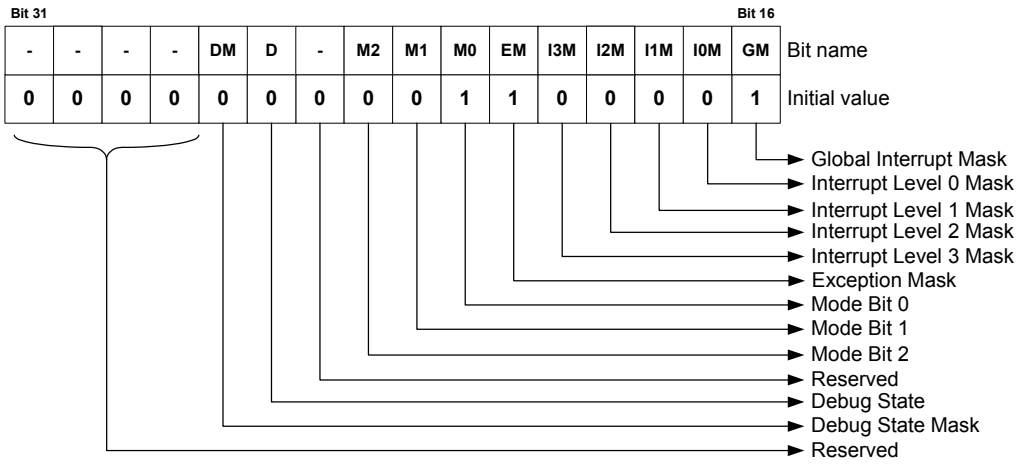
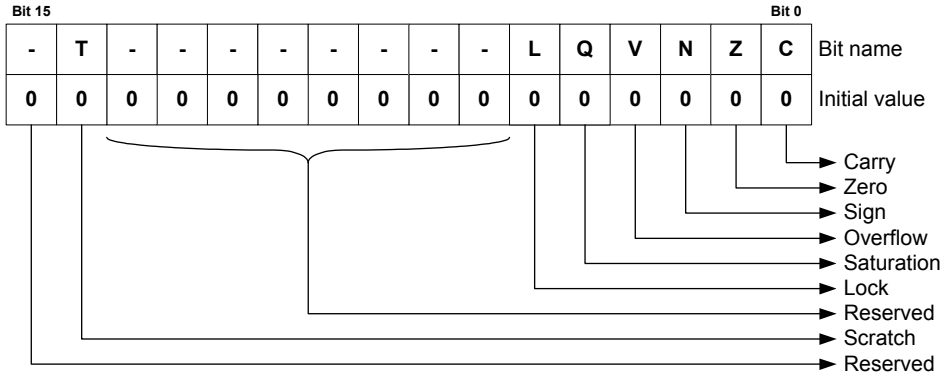


Figure 4-5. The Status Register Low Halfword



4.4.3 Processor States

4.4.3.1 Normal RISC State

The AVR32 processor supports several different execution contexts as shown in Table 4-2 on page 27.

Table 4-2. Overview of Execution Modes, their Priorities and Privilege Levels.

Priority	Mode	Security	Description
1	Non Maskable Interrupt	Privileged	Non Maskable high priority interrupt mode
2	Exception	Privileged	Execute exceptions
3	Interrupt 3	Privileged	General purpose interrupt mode
4	Interrupt 2	Privileged	General purpose interrupt mode
5	Interrupt 1	Privileged	General purpose interrupt mode
6	Interrupt 0	Privileged	General purpose interrupt mode
N/A	Supervisor	Privileged	Runs supervisor calls
N/A	Application	Unprivileged	Normal program execution mode

Mode changes can be made under software control, or can be caused by external interrupts or exception processing. A mode can be interrupted by a higher priority mode, but never by one with lower priority. Nested exceptions can be supported with a minimal software overhead.

When running an operating system on the AVR32, user processes will typically execute in the application mode. The programs executed in this mode are restricted from executing certain instructions. Furthermore, most system registers together with the upper halfword of the status register cannot be accessed. Protected memory areas are also not available. All other operating modes are privileged and are collectively called System Modes. They have full access to all privileged and unprivileged resources. After a reset, the processor will be in supervisor mode.

4.4.3.2 Debug State

The AVR32 can be set in a debug state, which allows implementation of software monitor routines that can read out and alter system information for use during application development. This implies that all system and application registers, including the status registers and program counters, are accessible in debug state. The privileged instructions are also available.

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

5. Memories

5.1 Embedded Memories

- **Internal High-Speed Flash**
  - **256KBytes (AT32UC3A3256/S)**
  - **128Kbytes (AT32UC3A3128/S)**
  - **64Kbytes (AT32UC3A364/S)**
    - **0 wait state access at up to 42MHz in worst case conditions**
    - **1 wait state access at up to 84MHz in worst case conditions**
    - **Pipelined Flash architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access**
    - **Pipelined Flash architecture typically reduces the cycle penalty of 1 wait state operation to only 15% compared to 0 wait state operation**
    - **100 000 write cycles, 15-year data retention capability**
    - **Sector lock capabilities, Bootloader protection, Security Bit**
    - **32 Fuses, Erased During Chip Erase**
    - **User page for data to be preserved during Chip Erase**
- **Internal High-Speed SRAM**
  - **64KBytes, Single-cycle access at full speed on CPU Local Bus and accessible through the High Speed Bud (HSB) matrix**
  - **2x32KBytes, accessible independently through the High Speed Bud (HSB) matrix**

5.2 Physical Memory Map

The System Bus is implemented as a bus matrix. All system bus addresses are fixed, and they are never remapped in any way, not even in boot.

Note that AVR32 UC CPU uses unsegmented translation, as described in the AVR32UC Technical Architecture Manual.

The 32-bit physical address space is mapped as follows:

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
Embedded CPU SRAM	0x00000000	64KByte	64KByte	64KByte
Embedded Flash	0x80000000	256KByte	128KByte	64KByte
EBI SRAM CS0	0xC0000000	16MByte	16MByte	16MByte
EBI SRAM CS2	0xC8000000	16MByte	16MByte	16MByte
EBI SRAM CS3	0xCC000000	16MByte	16MByte	16MByte
EBI SRAM CS4	0xD8000000	16MByte	16MByte	16MByte
EBI SRAM CS5	0xDC000000	16MByte	16MByte	16MByte
EBI SRAM CS1 /SDRAM CS0	0xD0000000	128MByte	128MByte	128MByte
USB Data	0xE0000000	64KByte	64KByte	64KByte

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 5-3. 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

6. Boot Sequence

This chapter summarizes the boot sequence of the AT32UC3A3/A4. The behavior after power-up is controlled by the Power Manager. For specific details, refer to Section 7. ”Power Manager (PM)” on page 86.

6.1 Starting of Clocks

After power-up, the device will be held in a reset state by the Power-On Reset circuitry, until the power has stabilized throughout the device. Once the power has stabilized, the device will use the internal RC Oscillator as clock source.

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 receives a clock with the same frequency as the internal RC Oscillator.

6.2 Fetching of Initial Instructions

After reset has been released, the AVR32 UC 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. BOD33 monitors this voltage and maintains the device under reset until VDDIO reaches the minimum voltage, preventing any spurious execution from flash.

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.

When powering up the device, there may be a delay before the voltage has stabilized, depending on the rise time of the supply used. The CPU can start executing code as soon as the supply is above the POR threshold, and before the supply is stable. Before switching to a high-speed clock source, the user should use the BOD to make sure the VDDCORE is above the minimum-level (1.62V).

7.2 DC Characteristics

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

7.5 Analog characteristics

7.5.1 ADC

Table 7-5. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{VDDANA}	Analog Power Supply		3.0		3.6	V

Table 7-6. Decoupling Requirements

Symbol	Parameter	Conditions	Typ.	Technology	Unit
C_{VDDANA}	Power Supply Capacitor		100	NPO	nF

7.5.2 BOD

Table 7-7. 1.8V BOD Level Values

Symbol	Parameter Value	Conditions	Min.	Typ.	Max.	Unit
BODLEVEL	00 1111b			1.79		V
	01 0111b			1.70		V
	01 1111b			1.61		V
	10 0111b			1.52		V

Table 7-7 describes the values of the BODLEVEL field in the flash FGPFR register.

Table 7-8. 3.3V BOD Level Values

Symbol	Parameter Value	Conditions	Min.	Typ.	Max.	Unit
BOD33LEVEL	Reset value			2.71		V
	1011			2.27		V
	1010			2.37		V
	1001			2.46		V
	1000			2.56		V
	0111			2.66		V
	0110			2.76		V
	0101			2.86		V
	0100			2.96		V
	0011			3.06		V
	0010			3.15		V
	0001			3.25		V
	0000			3.35		V

Table 7-8 describes the values of the BOD33.LEVEL field in the PM module

Table 7-36. SDRAM Clock Signal

Symbol	Parameter	Conditions	Min.	Max.	Unit
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge		6.3		ns
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge		2.4		ns
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge		7.4		ns
$SDRAMC_{16}$	CAS High after SDCK Rising Edge		1.9		ns
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge		6.4		ns
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge		2.2		ns
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge		9		ns
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge		0		ns
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge		7.6		ns
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge		1.8		ns
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge		7.1		ns
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge		1.5		ns

Figure 7-12. SPI Master mode with (CPOL= 0 and NCPHA= 1) or (CPOL= 1 and NCPHA= 0)

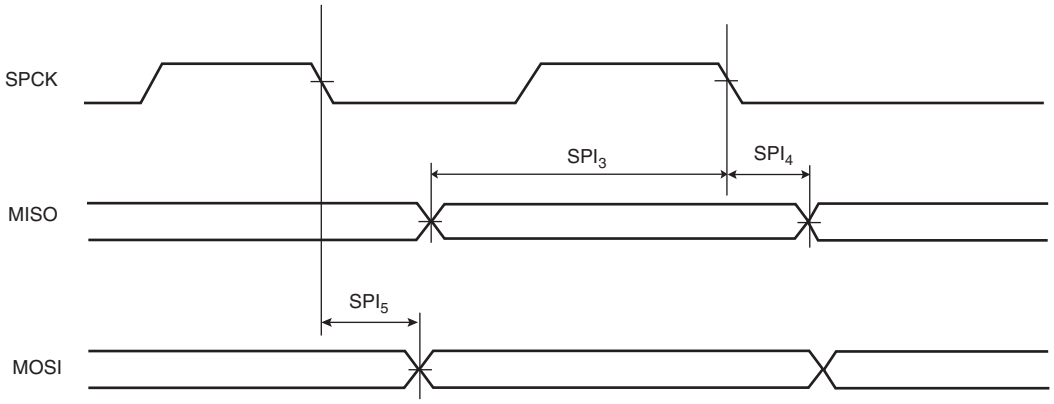


Figure 7-13. SPI Slave mode with (CPOL= 0 and NCPHA= 1) or (CPOL= 1 and NCPHA= 0)

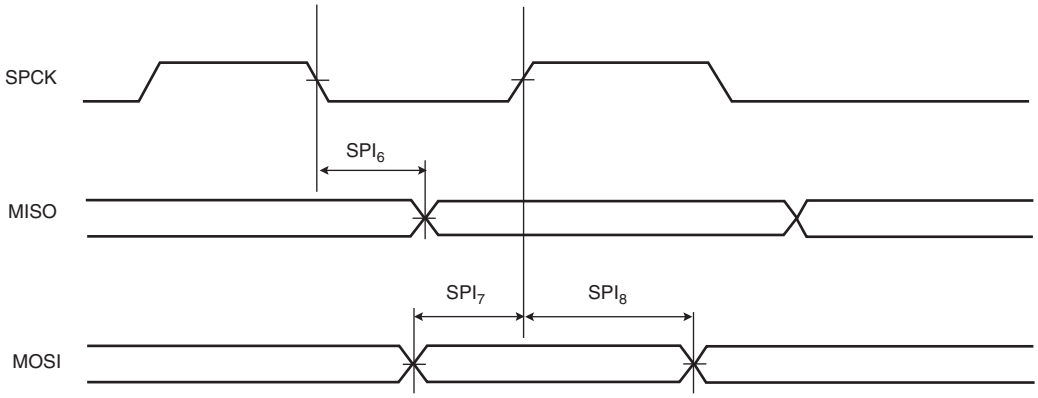


Figure 7-14. SPI Slave mode with (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)

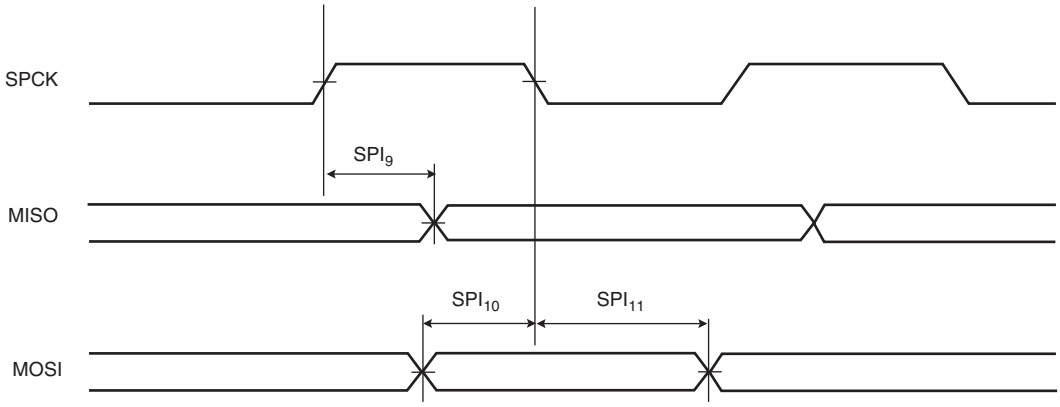
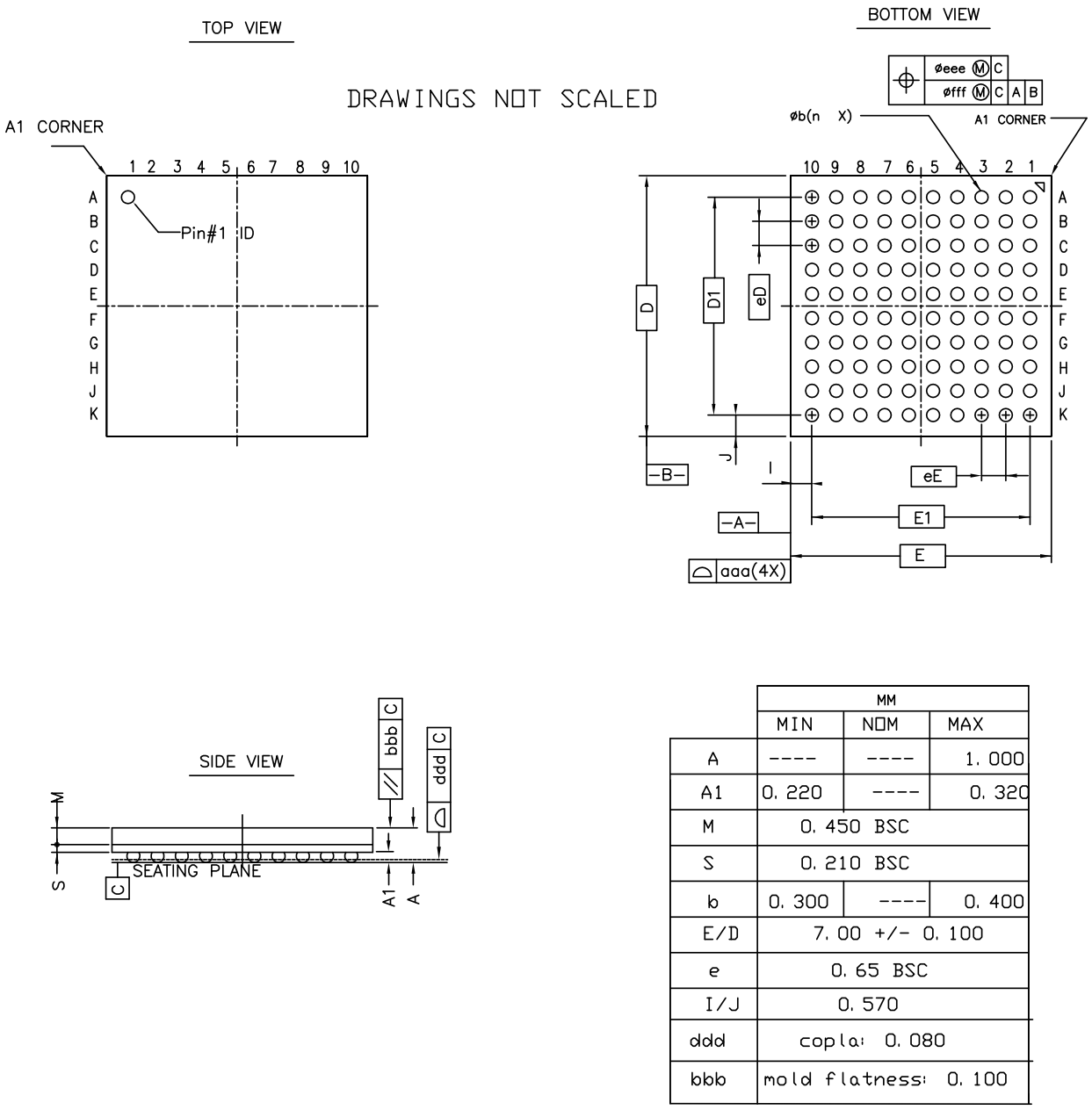


Figure 8-3. VFBGA-100 package drawing



	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
SOFT_RESET on RX+ Config US_MR + Config_US_CR.

10.2.7 SPI

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).

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.

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.

10.2.8 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.

10.3.7 SPI

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).

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.

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.

10.3.8 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.

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.