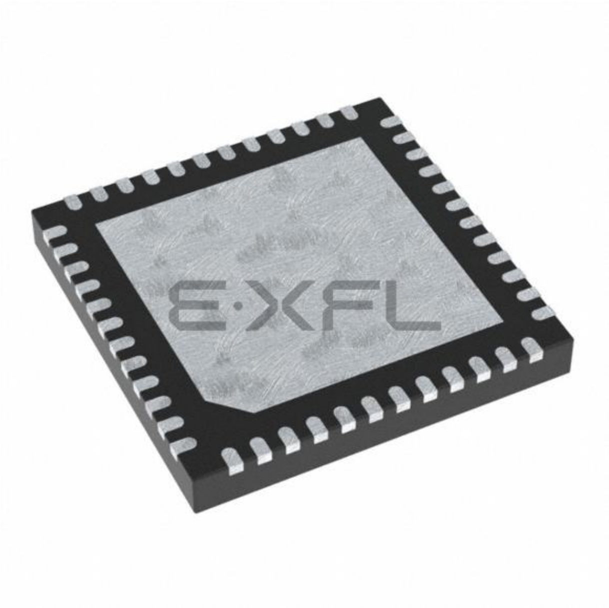


Welcome to E-XFL.COM

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	60MHz
Connectivity	I²C, IrDA, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	28
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	96K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 3.6V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3b1512-z1ut

Table 4-1. GPIO Controller Function Multiplexing

10	12	PA06	GPIO 6	EIC - EXTINT[1]	ADC - AD[3]	USART1 - DSR	ABDAC - DATAN[1]
11	13	PA07	GPIO 7	PWM - PWM[0]	ADC - AD[4]	USART1 - DTR	SSC - RX_FRAME_SYNC
12	14	PA08	GPIO 8	PWM - PWM[1]	ADC - AD[5]	USART1 - RI	SSC - RX_CLOCK
20	28	PA09	GPIO 9	TWI - SCL	SPI0 - NPCS[2]	USART1 - CTS	
21	29	PA10	GPIO 10	TWI - SDA	SPI0 - NPCS[3]	USART1 - RTS	
22	30	PA11	GPIO 11	USART0 - RTS	TC - A2	PWM - PWM[0]	SSC - RX_DATA
23	31	PA12	GPIO 12	USART0 - CTS	TC - B2	PWM - PWM[1]	USART1 - TXD
25	33	PA13	GPIO 13	EIC - NMI	PWM - PWM[2]	USART0 - CLK	SSC - RX_CLOCK
26	34	PA14	GPIO 14	SPI0 - MOSI	PWM - PWM[3]	EIC - EXTINT[2]	PM - GCLK[2]
27	35	PA15	GPIO 15	SPI0 - SCK	PWM - PWM[4]	USART2 - CLK	
28	36	PA16	GPIO 16	SPI0 - NPCS[0]	TC - CLK1	PWM - PWM[4]	
29	37	PA17	GPIO 17	SPI0 - NPCS[1]	TC - CLK2	SPI0 - SCK	USART1 - RXD
30	39	PA18	GPIO 18	USART0 - RXD	PWM - PWM[5]	SPI0 - MISO	SSC - RX_FRAME_SYNC
31	40	PA19	GPIO 19	USART0 - TXD	PWM - PWM[6]	SPI0 - MOSI	SSC - TX_CLOCK
32	44	PA20	GPIO 20	USART1 - CLK	TC - CLK0	USART2 - RXD	SSC - TX_DATA
33	45	PA21	GPIO 21	PWM - PWM[2]	TC - A1	USART2 - TXD	SSC - TX_FRAME_SYNC
34	46	PA22	GPIO 22	PWM - PWM[6]	TC - B1	ADC - TRIGGER	ABDAC - DATA[0]
35	47	PA23	GPIO 23	USART1 - TXD	SPI0 - NPCS[1]	EIC - EXTINT[3]	PWM - PWM[0]
43	59	PA24	GPIO 24	USART1 - RXD	SPI0 - NPCS[0]	EIC - EXTINT[4]	PWM - PWM[1]
44	60	PA25	GPIO 25	SPI0 - MISO	PWM - PWM[3]	EIC - EXTINT[5]	
45	61	PA26	GPIO 26	USBB - USB_ID	USART2 - TXD	TC - A0	ABDAC - DATA[1]
46	62	PA27	GPIO 27	USBB - USB_VBOF	USART2 - RXD	TC - B0	ABDAC - DATAN[1]
	41	PA28	GPIO 28	USART0 - CLK	PWM - PWM[4]	SPI0 - MISO	ABDAC - DATAN[0]
	42	PA29	GPIO 29	TC - CLK0	TC - CLK1	SPI0 - MOSI	
	15	PA30	GPIO 30	ADC - AD[6]	EIC - SCAN[0]	PM - GCLK[2]	
	16	PA31	GPIO 31	ADC - AD[7]	EIC - SCAN[1]	PWM - PWM[6]	
	6	PB00	GPIO 32	TC - A0	EIC - SCAN[2]	USART2 - CTS	
	7	PB01	GPIO 33	TC - B0	EIC - SCAN[3]	USART2 - RTS	
	24	PB02	GPIO 34	EIC - EXTINT[6]	TC - A1	USART1 - TXD	
	25	PB03	GPIO 35	EIC - EXTINT[7]	TC - B1	USART1 - RXD	
	26	PB04	GPIO 36	USART1 - CTS	SPI0 - NPCS[3]	TC - CLK2	
	27	PB05	GPIO 37	USART1 - RTS	SPI0 - NPCS[2]	PWM - PWM[5]	
	38	PB06	GPIO 38	SSC - RX_CLOCK	USART1 - DCD	EIC - SCAN[4]	ABDAC - DATA[0]
	43	PB07	GPIO 39	SSC - RX_DATA	USART1 - DSR	EIC - SCAN[5]	ABDAC - DATAN[0]
	54	PB08	GPIO 40	SSC - RX_FRAME_SYNC	USART1 - DTR	EIC - SCAN[6]	ABDAC - DATA[1]

The following table shows the instructions with support for unaligned addresses. All other instructions require aligned addresses.

Table 6-1. Instructions with Unaligned Reference Support

Instruction	Supported alignment
ld.d	Word
st.d	Word

6.3.6 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

6.3.7 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.

6.4 Programming Model

6.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 6-3. 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

6.4.2 Status Register Configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 6-4 on page 22 and Figure 6-5 on page 23. 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 6-4. The Status Register High Halfword

Bit 31															Bit 16	
-	-	-	-	DM	D	-	M2	M1	M0	EM	I3M	I2M	I1M	I0M	GM	Bit name
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	Initial value

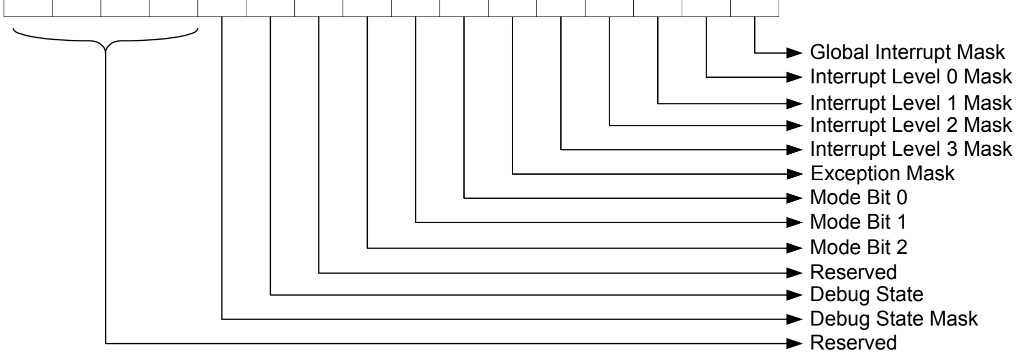


Table 6-3. System Registers (Continued)

Reg #	Address	Name	Function
92	368	MPUPSR4	MPU Privilege Select Register region 4
93	372	MPUPSR5	MPU Privilege Select Register region 5
94	376	MPUPSR6	MPU Privilege Select Register region 6
95	380	MPUPSR7	MPU Privilege Select Register region 7
96	384	MPUCRA	Unused in this version of AVR32UC
97	388	MPUCRB	Unused in this version of AVR32UC
98	392	MPUBRA	Unused in this version of AVR32UC
99	396	MPUBRB	Unused in this version of AVR32UC
100	400	MPUAPRA	MPU Access Permission Register A
101	404	MPUAPRB	MPU Access Permission Register B
102	408	MPUCR	MPU Control Register
103-191	448-764	Reserved	Reserved for future use
192-255	768-1020	IMPL	IMPLEMENTATION DEFINED

6.5 Exceptions and Interrupts

AVR32UC incorporates a powerful exception handling scheme. The different exception sources, like Illegal Op-code and external interrupt requests, have different priority levels, ensuring a well-defined behavior when multiple exceptions are received simultaneously. Additionally, pending exceptions of a higher priority class may preempt handling of ongoing exceptions of a lower priority class.

When an event occurs, the execution of the instruction stream is halted, and execution control is passed to an event handler at an address specified in Table 6-4 on page 29. Most of the handlers are placed sequentially in the code space starting at the address specified by EVBA, with four bytes between each handler. This gives ample space for a jump instruction to be placed there, jumping to the event routine itself. A few critical handlers have larger spacing between them, allowing the entire event routine to be placed directly at the address specified by the EVBA-relative offset generated by hardware. All external interrupt sources have autovectored interrupt service routine (ISR) addresses. This allows the interrupt controller to directly specify the ISR address as an address relative to EVBA. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes. The target address of the event handler is calculated as (EVBA | event_handler_offset), not (EVBA + event_handler_offset), so EVBA and exception code segments must be set up appropriately. The same mechanisms are used to service all different types of events, including external interrupt requests, yielding a uniform event handling scheme.

An interrupt controller does the priority handling of the external interrupts and provides the autovector offset to the CPU.

6.5.1 System Stack Issues

Event handling in AVR32UC uses the system stack pointed to by the system stack pointer, SP_SYS, for pushing and popping R8-R12, LR, status register, and return address. Since event code may be timing-critical, SP_SYS should point to memory addresses in the IRAM section, since the timing of accesses to this memory section is both fast and deterministic.

6.6 Module Configuration

All AT32UC3B parts do not implement the same CPU and Architecture Revision.

Table 6-5. CPU and Architecture Revision

Part Name	Architecture Revision
AT32UC3Bx512	2
AT32UC3Bx256	1
AT32UC3Bx128	1
AT32UC3Bx64	1

7. Memories

7.1 Embedded Memories

- **Internal High-Speed Flash**
  - **512KBytes (AT32UC3B0512, AT32UC3B1512)**
  - **256 KBytes (AT32UC3B0256, AT32UC3B1256)**
  - **128 KBytes (AT32UC3B0128, AT32UC3B1128)**
  - **64 KBytes (AT32UC3B064, AT32UC3B164)**
    - **- 0 Wait State Access at up to 30 MHz in Worst Case Conditions**
    - **- 1 Wait State Access at up to 60 MHz in Worst Case Conditions**
    - **- Pipelined Flash Architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access**
    - **- 100 000 Write Cycles, 15-year Data Retention Capability**
    - **- 4 ms Page Programming Time, 8 ms Chip Erase Time**
    - **- 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, Single-cycle access at full speed**
  - **96KBytes ((AT32UC3B0512, AT32UC3B1512)**
  - **32KBytes (AT32UC3B0256, AT32UC3B0128, AT32UC3B1256 and AT32UC3B1128)**
  - **16KBytes (AT32UC3B064 and AT32UC3B164)**

7.2 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 7-1. AT32UC3B Physical Memory Map

Device		Embedded SRAM	Embedded Flash	USB Data	HSB-PB Bridge A	HSB-PB Bridge B
Start Address		0x0000_0000	0x8000_0000	0xD000_0000	0xFFFF_0000	0xFFFE_0000
Size	AT32UC3B0512 AT32UC3B1512	96 Kbytes	512 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B0256 AT32UC3B1256	32 Kbytes	256 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B0128 AT32UC3B1128	32 Kbytes	128 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B064 AT32UC3B164	16 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes

Table 7-2. Peripheral Address Mapping

Address	Peripheral	Description
0xFFFF3C00	ADC	Analog to Digital Converter - ADC
0xFFFF4000	ABDAC	Audio Bitstream DAC - ABDAC

7.4 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 7-3. Local bus mapped GPIO registers

Port	Register	Mode	Local Bus Address	Access
0	Output Driver Enable Register (ODER)	WRITE	0x4000_0040	Write-only
		SET	0x4000_0044	Write-only
		CLEAR	0x4000_0048	Write-only
		TOGGLE	0x4000_004C	Write-only
	Output Value Register (OVR)	WRITE	0x4000_0050	Write-only
		SET	0x4000_0054	Write-only
		CLEAR	0x4000_0058	Write-only
		TOGGLE	0x4000_005C	Write-only
	Pin Value Register (PVR)	-	0x4000_0060	Read-only
1	Output Driver Enable Register (ODER)	WRITE	0x4000_0140	Write-only
		SET	0x4000_0144	Write-only
		CLEAR	0x4000_0148	Write-only
		TOGGLE	0x4000_014C	Write-only
	Output Value Register (OVR)	WRITE	0x4000_0150	Write-only
		SET	0x4000_0154	Write-only
		CLEAR	0x4000_0158	Write-only
		TOGGLE	0x4000_015C	Write-only
	Pin Value Register (PVR)	-	0x4000_0160	Read-only

9.2 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 to T_J = 100°C.

Table 9-1. DC Characteristics

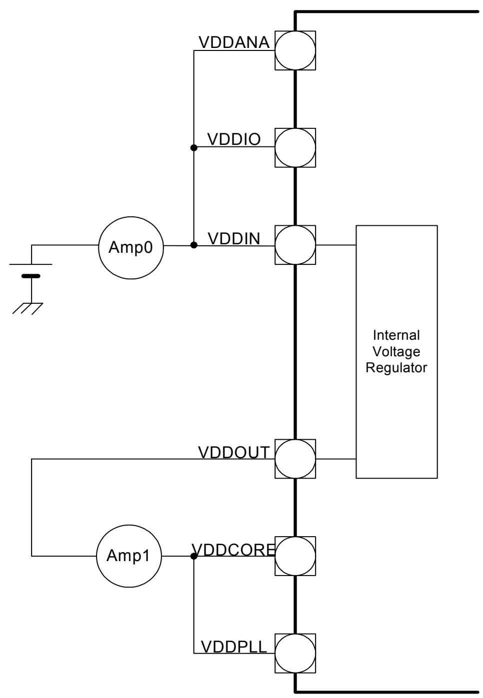
Symbol	Parameter	Conditions		Min.	Typ.	Max.	Unit
$V_{VDDCORE}$	DC Supply Core			1.65		1.95	V
V_{VDDPLL}	DC Supply PLL			1.65		1.95	V
V_{VDDIO}	DC Supply Peripheral I/Os			3.0		3.6	V
V_{IL}	Input Low-level Voltage			-0.3		+0.8	V
V_{IH}	Input High-level Voltage	AT32UC3B064 AT32UC3B0128 AT32UC3B0256 AT32UC3B164 AT32UC3B1128 AT32UC3B1256	All I/O pins except TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		5.5	V
			TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		3.6	V
		AT32UC3B0512 AT32UC3B1512	All I/O pins except TCK, RESET_N, PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		5.5	V
			TCK, RESET_N	2.5		3.6	V
			PA03, PA04, PA05, PA06, PA07, PA08, PA11, PA12, PA18, PA19, PA28, PA29, PA30, PA31	2.0		3.6	V
V_{OL}	Output Low-level Voltage	I_{OL}= -4mA for all I/O except PA20, PA21, PA22, PA23				0.4	V
		I_{OL}= -8mA for PA20, PA21, PA22, PA23				0.4	V
V_{OH}	Output High-level Voltage	I_{OL}= -4mA for all I/O except PA20, PA21, PA22, PA23		V_{VDDIO} -0.4			V
		I_{OL}= -8mA for PA20, PA21, PA22, PA23		V_{VDDIO} -0.4			V
I_{OL}	Output Low-level Current	All I/O pins except PA20, PA21, PA22, PA23				-4	mA
		PA20, PA21, PA22, PA23				-8	mA
I_{OH}	Output High-level Current	All I/O pins except for PA20, PA21, PA22, PA23				4	mA
		PA20, PA21, PA22, PA23				8	mA
I_{LEAK}	Input Leakage Current	Pullup resistors disabled				1	µA

9.5 Power Consumption

The values in Table 9-10, Table 9-11 on page 43 and Table 9-12 on page 44 are measured values of power consumption with operating conditions as follows:

- $V_{DDIO} = V_{DDANA} = 3.3V$
- $V_{DDCORE} = V_{DDPLL} = 1.8V$
- $T_A = 25°C$, $T_A = 85°C$
- I/Os are configured in input, pull-up enabled.

Figure 9-5. Measurement Setup



The following tables represent the power consumption measured on the power supplies.

9.10 JTAG Characteristics

9.10.1 JTAG Timing

Figure 9-6. JTAG Interface Signals

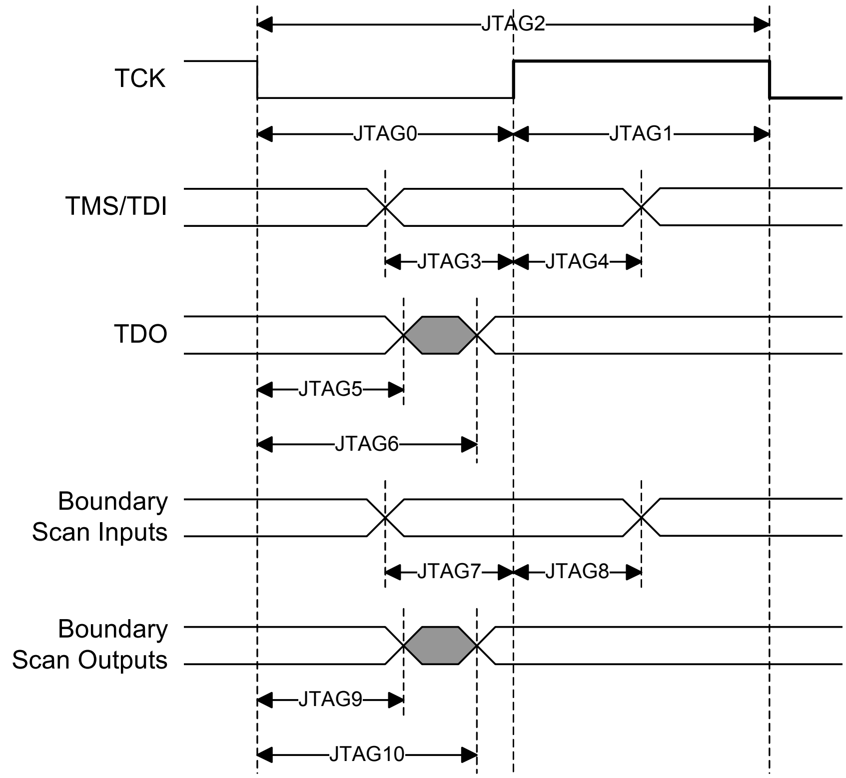


Table 9-26. JTAG Timings[(1)]

Symbol	Parameter	Conditions	Min	Max	Units
JTAG0	TCK Low Half-period	V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF	23.2		ns
JTAG1	TCK High Half-period		8.8		ns
JTAG2	TCK Period		32.0		ns
JTAG3	TDI, TMS Setup before TCK High		3.9		ns
JTAG4	TDI, TMS Hold after TCK High		0.6		ns
JTAG5	TDO Hold Time		4.5		ns
JTAG6	TCK Low to TDO Valid			23.2	ns
JTAG7	Boundary Scan Inputs Setup Time		0		ns
JTAG8	Boundary Scan Inputs Hold Time		5.0		ns
JTAG9	Boundary Scan Outputs Hold Time		8.7		ns
JTAG10	TCK to Boundary Scan Outputs Valid			17.7	ns

Note: 1. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same pro-cess technology. These values are not covered by test limits in production.

Figure 9-10. SPI Slave mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

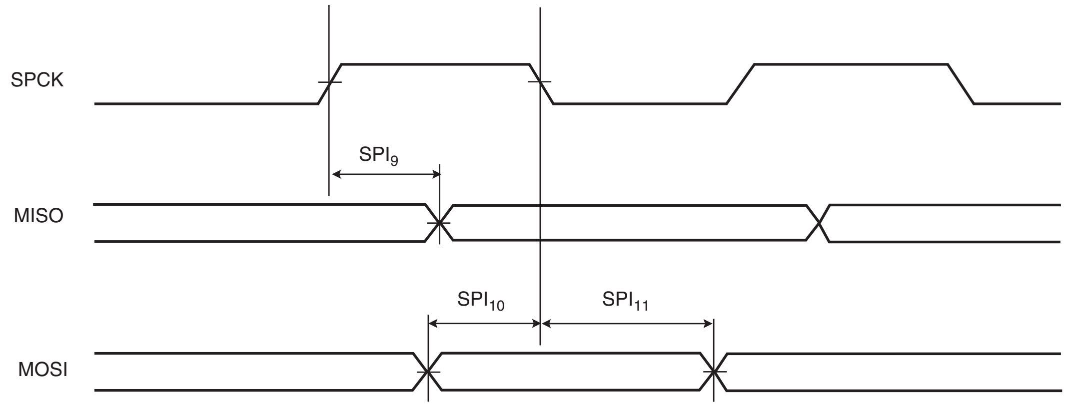
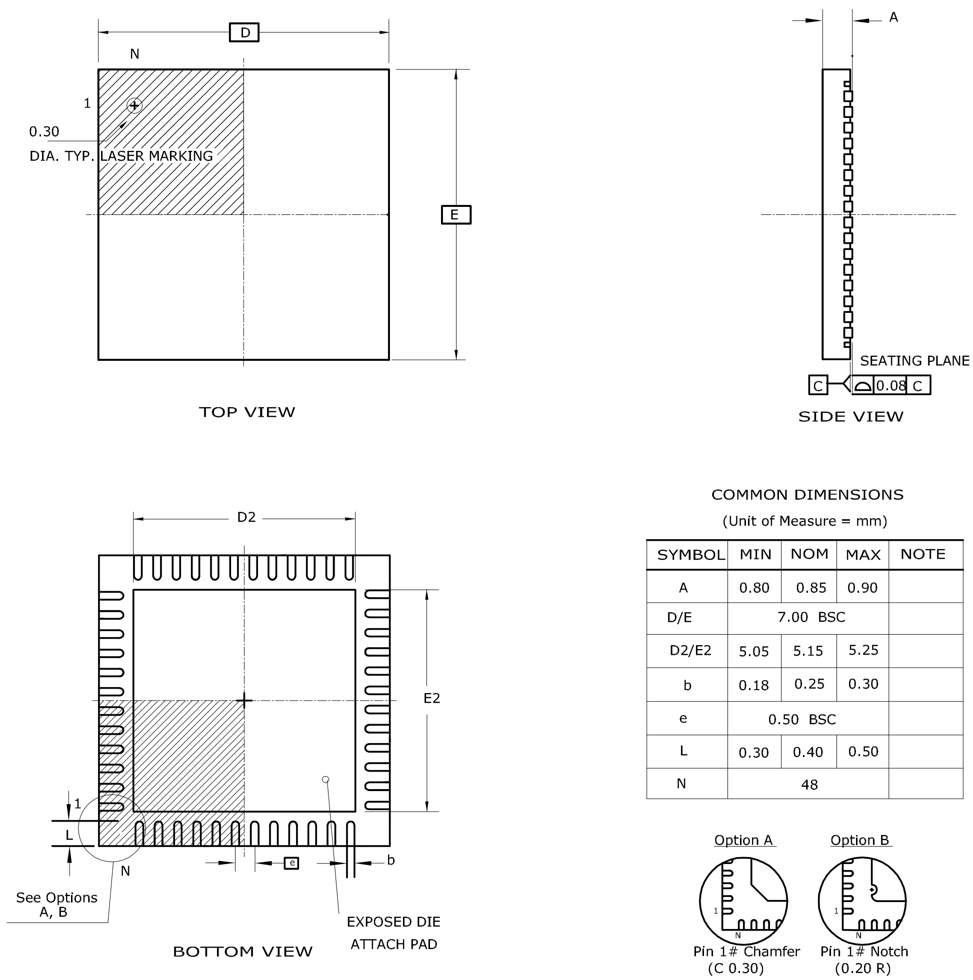


Table 9-27. SPI Timings

Symbol	Parameter	Conditions	Min.	Max.	Unit
SPI_0	MISO Setup time before SPCK rises (master)	3.3V domain[(1)]	22 + (t_{CPMCK})/2[(2)]		ns
SPI_1	MISO Hold time after SPCK rises (master)	3.3V domain[(1)]	0		ns
SPI_2	SPCK rising to MOSI Delay (master)	3.3V domain[(1)]		7	ns
SPI_3	MISO Setup time before SPCK falls (master)	3.3V domain[(1)]	22 + (t_{CPMCK})/2[(2)]		ns
SPI_4	MISO Hold time after SPCK falls (master)	3.3V domain[(1)]	0		ns
SPI_5	SPCK falling to MOSI Delay master)	3.3V domain[(1)]		7	ns
SPI_6	SPCK falling to MISO Delay (slave)	3.3V domain[(1)]		26.5	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	3.3V domain[(1)]	0		ns
SPI_8	MOSI Hold time after SPCK rises (slave)	3.3V domain[(1)]	1.5		ns
SPI_9	SPCK rising to MISO Delay (slave)	3.3V domain[(1)]		27	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	3.3V domain[(1)]	0		ns
SPI_{11}	MOSI Hold time after SPCK falls (slave)	3.3V domain[(1)]	1		ns

Notes: 1. 3.3V domain: V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40 pF.
2. t_{CPMCK}: Master Clock period in ns.

Figure 10-4. QFN-48 package drawing



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.85	0.90	
D/E		7.00 BSC		
D2/E2	5.05	5.15	5.25	
b	0.18	0.25	0.30	
e		0.50 BSC		
L	0.30	0.40	0.50	
N		48		

Notes : 1. This drawing is for general information only. Refer to JEDEC Drawing MO-220, Variation VKKD-4, for proper dimensions, tolerances, datums, etc.
2. Dimension b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip.
If the terminal has the optical radius on the other end of the terminal, the dimension should not be measured in that radius area.

Table 10-11. Device and Package Maximum Weight

Weight	100 mg

Table 10-12. Package Characteristics

Moisture Sensitivity Level	Jedec J-STD-20D-MSL3

Table 10-13. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	e3

7. SPI Glitch on RXREADY flag in slave mode when enabling the SPI or during the first transfer

In slave mode, the SPI can generate a false RXREADY signal during enabling of the SPI or during the first transfer.

Fix/Workaround

1. Set slave mode, set required CPOL/CPHA.
2. Enable SPI.
3. Set the polarity CPOL of the line in the opposite value of the required one.
4. Set the polarity CPOL to the required one.
5. Read the RXHOLDING register.

Transfers can now begin and RXREADY will now behave as expected.

8. 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).

9. 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.

10. 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.

11. Power Manager

12. If the BOD level is higher than VDDCORE, the part is constantly reset

If the BOD level is set to a value higher than VDDCORE and enabled by fuses, the part will be in constant reset.

Fix/Workaround

Apply an external voltage on VDDCORE that is higher than the BOD level and is lower than VDDCORE max and disable the BOD.

13. 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.

14. 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

8. **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).

9. **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.

10. **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.

11. **Power Manager**

12. **If the BOD level is higher than VDDCORE, the part is constantly reset**
    If the BOD level is set to a value higher than VDDCORE and enabled by fuses, the part will be in constant reset.
    Fix/Workaround
    Apply an external voltage on VDDCORE that is higher than the BOD level and is lower than VDDCORE max and disable the BOD.

13. **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.

14. **VDDCORE power supply input needs to be 1.95V**
    When used in dual power supply, VDDCORE needs to be 1.95V.
    Fix/Workaround
    When used in single power supply, VDDCORE needs to be connected to VDDOUT, which is configured on revision C at 1.95V (typ.).

15. **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.

12.2 AT32UC3B0256, AT32UC3B0128, AT32UC3B064, AT32UC3B1256, AT32UC3B1128, AT32UC3B164

All industrial parts labelled with -UES (for engineering samples) are revision B parts.

12.2.1 Rev I, J, K

- *PWM*

1. **PWM channel interrupt enabling triggers an interrupt**
   When enabling a PWM channel that is configured with center aligned period (CALG=1), an interrupt is signalled.
   Fix/Workaround
   When using center aligned mode, enable the channel and read the status before channel interrupt is enabled.

2. **PWN counter restarts at 0x0001**
   The PWM counter restarts at 0x0001 and not 0x0000 as specified. Because of this the first PWM period has one more clock cycle.
   Fix/Workaround
   - The first period is 0x0000, 0x0001, ..., period.
   - Consecutive periods are 0x0001, 0x0002, ..., period.

3. **PWM update period to a 0 value does not work**
   It is impossible to update a period equal to 0 by the using the PWM update register (PWM_CUPD).
   Fix/Workaround
   Do not update the PWM_CUPD register with a value equal to 0.

4. **SPI**

5. **SPI Slave / PDCA transfer: no TX UNDERRUN flag**
   There is no TX UNDERRUN flag available, therefore in SPI slave mode, there is no way to be informed of a character lost in transmission.
   Fix/Workaround
   For PDCA transfer: none.

6. **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.

7. **SPI Glitch on RXREADY flag in slave mode when enabling the SPI or during the first transfer**
   In slave mode, the SPI can generate a false RXREADY signal during enabling of the SPI or during the first transfer.
   Fix/Workaround
   1. Set slave mode, set required CPOL/CPHA.
   2. Enable SPI.

12.2.2 Rev. G

- PWM

1. **PWM channel interrupt enabling triggers an interrupt**
   When enabling a PWM channel that is configured with center aligned period (CALG=1), an interrupt is signalled.
   Fix/Workaround
   When using center aligned mode, enable the channel and read the status before channel interrupt is enabled.

2. **PWN counter restarts at 0x0001**
   The PWM counter restarts at 0x0001 and not 0x0000 as specified. Because of this the first PWM period has one more clock cycle.
   Fix/Workaround
   - The first period is 0x0000, 0x0001, ..., period.
   - Consecutive periods are 0x0001, 0x0002, ..., period.

3. **PWM update period to a 0 value does not work**
   It is impossible to update a period equal to 0 by the using the PWM update register (PWM_CUPD).
   Fix/Workaround
   Do not update the PWM_CUPD register with a value equal to 0.

4. **SPI**

5. **SPI Slave / PDCA transfer: no TX UNDERRUN flag**
   There is no TX UNDERRUN flag available, therefore in SPI slave mode, there is no way to be informed of a character lost in transmission.
   Fix/Workaround
   For PDCA transfer: none.

6. **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.

7. **SPI Glitch on RXREADY flag in slave mode when enabling the SPI or during the first transfer**
   In slave mode, the SPI can generate a false RXREADY signal during enabling of the SPI or during the first transfer.
   Fix/Workaround
   1. Set slave mode, set required CPOL/CPHA.
   2. Enable SPI.
   3. Set the polarity CPOL of the line in the opposite value of the required one.
   4. Set the polarity CPOL to the required one.
   5. Read the RXHOLDING register.
   Transfers can now begin and RXREADY will now behave as expected.

2. **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.

3. **TWI**

4. **The TWI RXRDY flag in SR register is not reset when a software reset is performed**
The TWI RXRDY flag in SR register is not reset when a software reset is performed.
Fix/Workaround
After a Software Reset, the register TWI RHR must be read.

5. **TWI in master mode will continue to read data**
TWI in master mode will continue to read data on the line even if the shift register and the RHR register are full. This will generate an overrun error.
Fix/Workaround
To prevent this, read the RHR register as soon as a new RX data is ready.

6. **TWI slave behaves improperly if master acknowledges the last transmitted data byte before a STOP condition**
In I2C slave transmitter mode, if the master acknowledges the last data byte before a STOP condition (what the master is not supposed to do), the following TWI slave receiver mode frame may contain an inappropriate clock stretch. This clock stretch can only be stopped by resetting the TWI.
Fix/Workaround
If the TWI is used as a slave transmitter with a master that acknowledges the last data byte before a STOP condition, it is necessary to reset the TWI before entering slave receiver mode.

7. **GPIO**

8. **PA29 (TWI SDA) and PA30 (TWI SCL) GPIO VIH (input high voltage) is 3.6V max instead of 5V tolerant**
The following GPIOs are not 5V tolerant: PA29 and PA30.
Fix/Workaround
None.

- *TC*

1. **Channel chaining skips first pulse for upper channel**
When chaining two channels using the Block Mode Register, the first pulse of the clock between the channels is skipped.
Fix/Workaround
Configure the lower channel with RA = 0x1 and RC = 0x2 to produce a dummy clock cycle for the upper channel. After the dummy cycle has been generated, indicated by the SR.CPCS bit, reconfigure the RA and RC registers for the lower channel with the real values.

it is done atomically. Even if this step is described in general as not safe in the UC technical reference guide, it is safe in this very specific case.
2. Execute the RETE instruction.

13. 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.

13.1 Rev. L– 01/2012

1. Updated Mechanical Characteristics section.

13.2 Rev. K– 02/2011

1. Updated USB section.
2. Updated Configuration Summary section.
3. Updated Electrical Characteristics section.
4. Updated Errata section.

13.3 Rev. J– 12/2010

1. Updated USB section.
2. Updated USART section.
3. Updated TWI section.
4. Updated PWM section.
5. Updated Electrical Characteristics section.

13.4 Rev. I – 06/2010

1. Updated SPI section.
2. Updated Electrical Characteristics section.

13.5 Rev. H – 10/2009

1. Update datasheet architecture.
2. Add AT32UC3B0512 and AT32UC3B1512 devices description.

13.12 Rev. A – 05/2007

1. Initial revision.