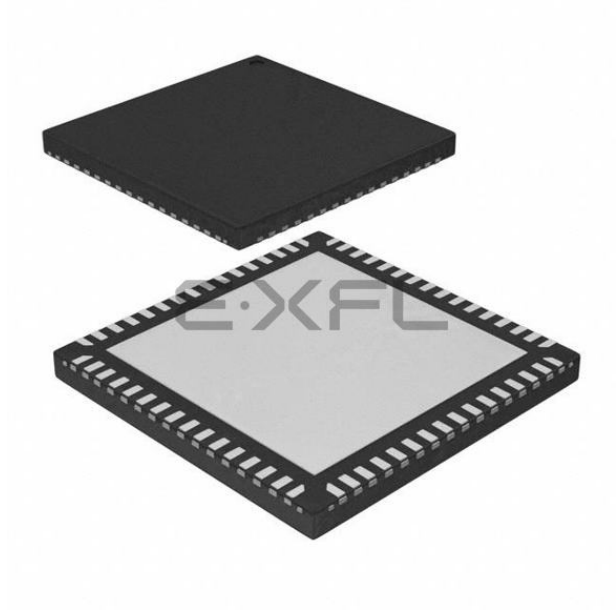


Welcome to E-XFL.COM

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	44
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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3b0512-z2ut

1. Description

The AT32UC3B is a complete System-On-Chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 60 MHz. AVR32 UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density and high performance.

The processor implements a Memory Protection Unit (MPU) and a fast and flexible interrupt controller for supporting modern operating systems and real-time operating systems.

Higher computation capability is achieved using a rich set of DSP instructions.

The AT32UC3B incorporates on-chip Flash and SRAM memories for secure and fast access.

The Peripheral Direct Memory Access controller enables data transfers between peripherals and memories without processor involvement. PDCA drastically reduces processing overhead when transferring continuous and large data streams between modules within the MCU.

The Power Manager improves design flexibility and security: the on-chip Brown-Out Detector monitors the power supply, the CPU runs from the on-chip RC oscillator or from one of external oscillator sources, a Real-Time Clock and its associated timer keeps track of the time.

The Timer/Counter includes three identical 16-bit timer/counter channels. Each channel can be independently programmed to perform frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

The PWM modules provides seven independent channels with many configuration options including polarity, edge alignment and waveform non overlap control. One PWM channel can trigger ADC conversions for more accurate close loop control implementations.

The AT32UC3B also features many communication interfaces for communication intensive applications. In addition to standard serial interfaces like USART, SPI or TWI, other interfaces like flexible Synchronous Serial Controller and USB are available. The USART supports different communication modes, like SPI mode.

The Synchronous Serial Controller provides easy access to serial communication protocols and audio standards like I^2S, UART or SPI.

The Full-Speed USB 2.0 Device interface supports several USB Classes at the same time thanks to the rich End-Point configuration. The Embedded Host interface allows device like a USB Flash disk or a USB printer to be directly connected to the processor.

Atmel offers the QTouch library for embedding capacitive touch buttons, sliders, and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and included fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS®) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop, and debug your own touch applications.

AT32UC3B integrates a class 2+ Nexus 2.0 On-Chip Debug (OCD) System, with non-intrusive real-time trace, full-speed read/write memory access in addition to basic runtime control. The Nanotrace interface enables trace feature for JTAG-based debuggers.

5.2 RESET_N pin

The RESET_N pin is a schmitt input and integrates a permanent pull-up resistor to VDDIO. As the product integrates a power-on reset cell, the RESET_N pin can be left unconnected in case no reset from the system needs to be applied to the product.

5.3 TWI pins

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with inputs with spike-filtering. When used as GPIO-pins or used for other peripherals, the pins have the same characteristics as GPIO pins.

5.4 GPIO pins

All the I/O lines integrate a pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the GPIO Controllers. After reset, I/O lines default as inputs with pull-up resistors disabled, except when indicated otherwise in the column “Reset Value” of the GPIO Controller user interface table.

5.5 High drive pins

The four pins PA20, PA21, PA22, PA23 have high drive output capabilities.

5.6 Power Considerations

5.6.1 Power Supplies

The AT32UC3B has several types of power supply pins:

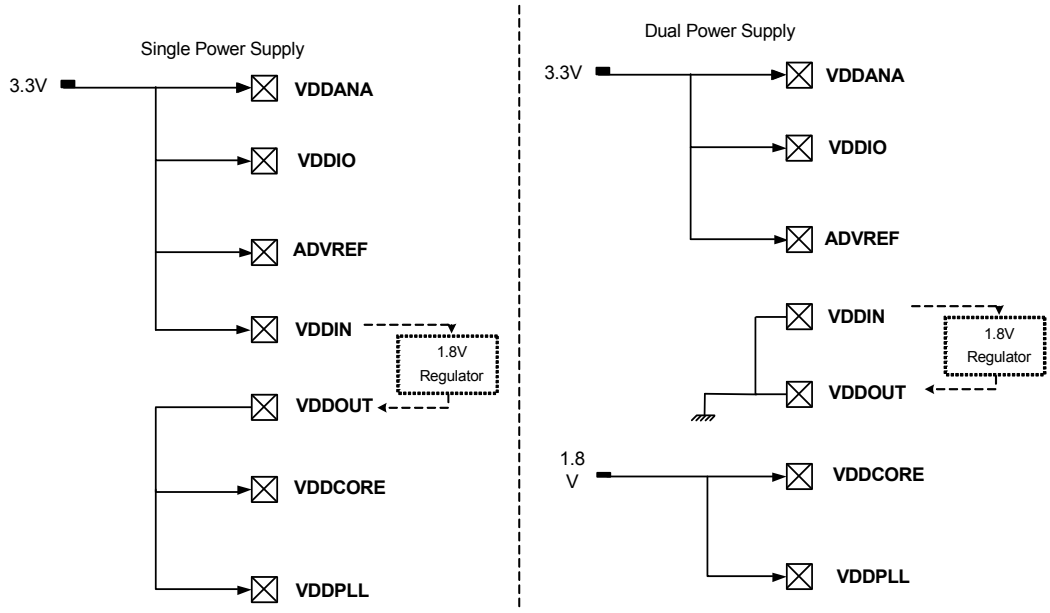
- **VDDIO: Powers I/O lines. Voltage is 3.3V nominal.**
- **VDDANA: Powers the ADC Voltage is 3.3V nominal.**
- **VDDIN: Input voltage for the voltage regulator. Voltage is 3.3V nominal.**
- **VDDCORE: Powers the core, memories, and peripherals. Voltage is 1.8V nominal.**
- **VDDPLL: Powers the PLL. Voltage is 1.8V nominal.**

The ground pins GND are common to VDDCORE, VDDIO and VDDPLL. The ground pin for VDDANA is GNDANA.

Refer to Electrical Characteristics section for power consumption on the various supply pins.

The main requirement for power supplies connection is to respect a star topology for all electrical connection.

Figure 5-1. Power Supply



5.6.2 Voltage Regulator

5.6.2.1 Single Power Supply

The AT32UC3B embeds a voltage regulator that converts from 3.3V to 1.8V. The regulator takes its input voltage from VDDIN, and supplies the output voltage on VDDOUT that should be externally connected to the 1.8V domains.

Adequate input supply decoupling is mandatory for VDDIN in order to improve startup stability and reduce source voltage drop. Two input decoupling capacitors must be placed close to the chip.

Adequate output supply decoupling is mandatory for VDDOUT to reduce ripple and avoid oscillations. The best way to achieve this is to use two capacitors in parallel between VDDOUT and GND as close to the chip as possible

Figure 5-2. Supply Decoupling

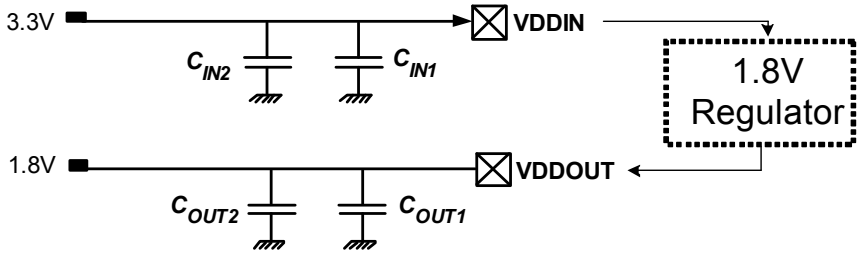
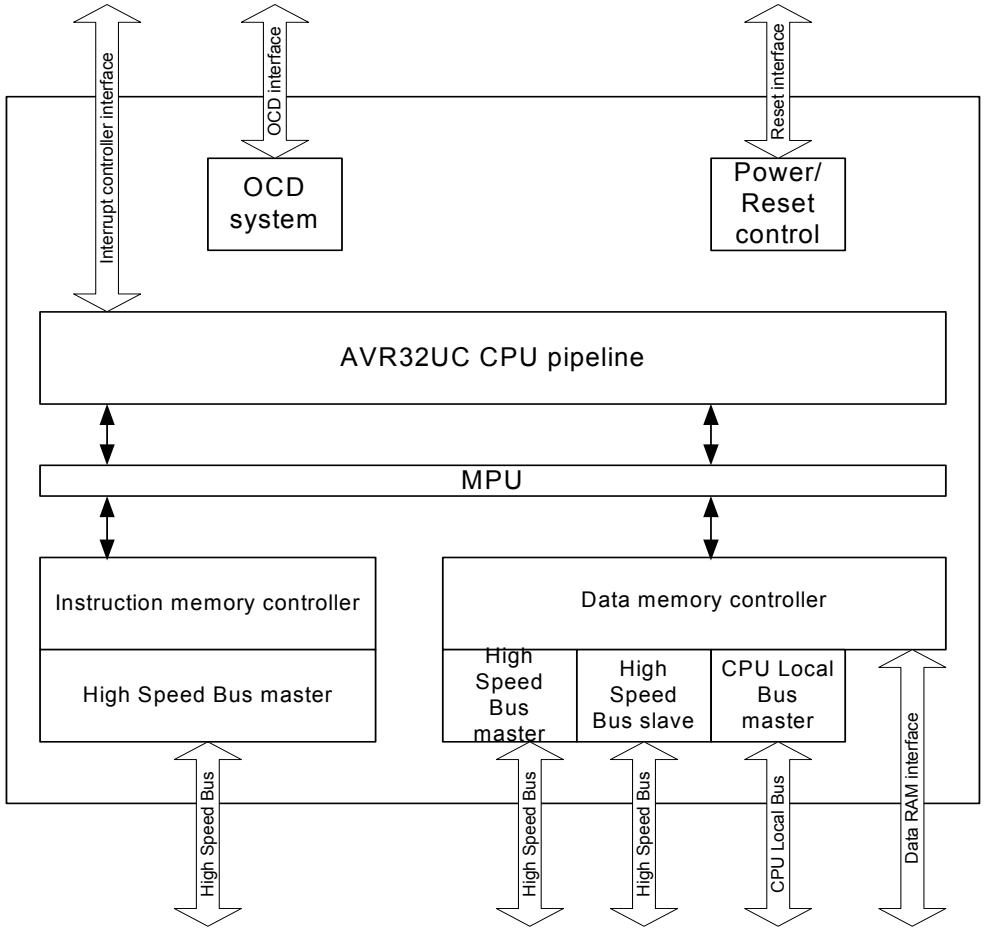


Figure 6-1. Overview of the AVR32UC CPU



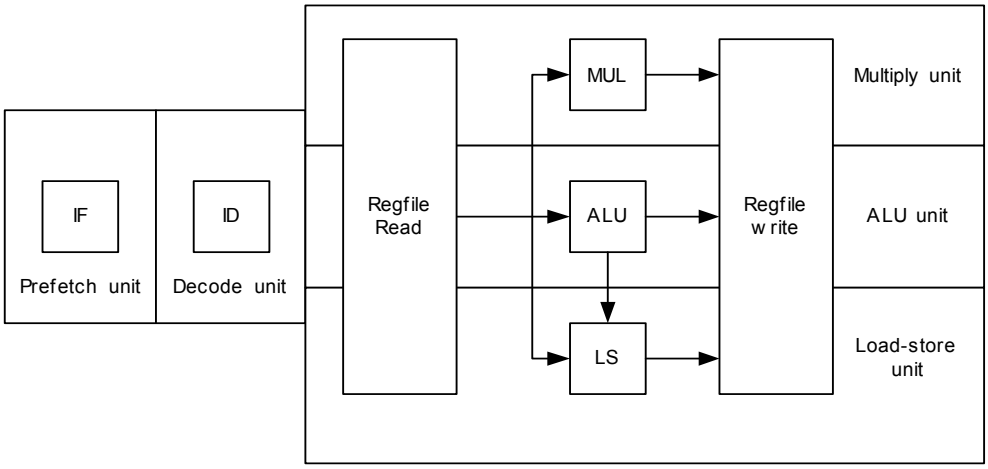
6.3.1 Pipeline Overview

AVR32UC has three pipeline stages, Instruction Fetch (IF), Instruction Decode (ID), and Instruction Execute (EX). The EX stage is split into three parallel subsections, one arithmetic/logic (ALU) section, one multiply (MUL) section, and one load/store (LS) section.

Instructions are issued and complete in order. Certain operations require several clock cycles to complete, and in this case, the instruction resides in the ID and EX stages for the required number of clock cycles. Since there is only three pipeline stages, no internal data forwarding is required, and no data dependencies can arise in the pipeline.

Figure 6-2 on page 20 shows an overview of the AVR32UC pipeline stages.

Figure 6-2. The AVR32UC Pipeline



6.3.2 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.

6.3.3 Java Support

AVR32UC does not provide Java hardware acceleration.

6.3.4 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.

6.3.5 Unaligned Reference Handling

AVR32UC does not support unaligned accesses, except for doubleword accesses. AVR32UC is able to perform word-aligned *st.d* and *ld.d*. Any other unaligned memory access will cause an address exception. Doubleword-sized accesses with word-aligned pointers will automatically be performed as two word-sized accesses.

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

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

Debug state is exited by the *retd* instruction.

6.4.4 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 6-3. 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 7-2. Peripheral Address Mapping

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

Table 9-1. DC Characteristics

Symbol	Parameter	Conditions			Min.	Typ.	Max.	Unit
C_{IN}	Input Capacitance	QFP64					7	pF
		QFP48					7	pF
		QFN64					7	pF
		QFN48					7	pF
R_{PULLUP}	Pull-up Resistance	AT32UC3B064 AT32UC3B0128 AT32UC3B0256 AT32UC3B164 AT32UC3B1128 AT32UC3B1256	All I/O pins except RESET_N, TCK, TDI, TMS pins		13	19	25	KΩ
			RESET_N pin, TCK, TDI, TMS pins		5	12	25	KΩ
		AT32UC3B0512 AT32UC3B1512	All I/O pins except PA20, PA21, PA22, PA23, RESET_N, TCK, TDI, TMS pins		10	15	20	KΩ
			PA20, PA21, PA22, PA23		5	7.5	12	KΩ
			RESET_N pin, TCK, TDI, TMS pins		5	10	25	KΩ
I_{SC}	Static Current	AT32UC3B064 AT32UC3B0128 AT32UC3B0256 AT32UC3B164 AT32UC3B1128 AT32UC3B1256	On $V_{VDDCORE}$ = 1.8V, device in static mode	T_A = 25°C		6		µA
			All inputs driven including JTAG; RESET_N=1	T_A = 85°C		42.5		µA
		AT32UC3B0512 AT32UC3B1512	On $V_{VDDCORE}$ = 1.8V, device in static mode	T_A = 25°C		7.5		µA
			All inputs driven including JTAG; RESET_N=1	T_A = 85°C		39		µA

9.3 Regulator Characteristics

Table 9-2. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{VDDIN}	Supply voltage (input)		3	3.3	3.6	V
V_{VDDOUT}	Supply voltage (output)		1.70	1.8	1.85	V
I_{OUT}	Maximum DC output current	V_{VDDIN} = 3.3V			100	mA
I_{SCR}	Static Current of internal regulator	Low Power mode (stop, deep stop or static) at T_A = 25°C		10		µA

Table 9-3. 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

9.4 Analog Characteristics

9.4.1 ADC Reference

Table 9-4. Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{ADVREF}	Analog voltage reference (input)		2.6		3.6	V

Table 9-5. Decoupling Requirements

Symbol	Parameter	Conditions	Typ.	Technology	Unit
C_{VREF1}	Voltage reference Capacitor 1		10	NPO	nF
C_{VREF2}	Voltage reference Capacitor 2		1	NPO	uF

9.4.2 BOD

Table 9-6. BOD Level Values

Symbol	Parameter Value	Conditions	Min.	Typ.	Max.	Unit
BODLEVEL	00 0000b			1.44		V
	01 0111b			1.52		V
	01 1111b			1.61		V
	10 0111b			1.71		V

Table 9-6 describes the values of the BODLEVEL field in the flash FGPFR register.

Table 9-23. Transfer Characteristics in 8-bit Mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Differential Non-linearity	ADC Clock = 5 MHz		0.3	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.0	LSB
Offset Error	ADC Clock = 5 MHz	-0.5		0.5	LSB
Gain Error	ADC Clock = 5 MHz	-0.5		0.5	LSB

Table 9-24. Transfer Characteristics in 10-bit Mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Resolution			10		Bit
Absolute Accuracy	ADC Clock = 5 MHz			3	LSB
Integral Non-linearity	ADC Clock = 5 MHz		1.5	2	LSB
Differential Non-linearity	ADC Clock = 5 MHz		1	2	LSB
	ADC Clock = 2.5 MHz		0.6	1	LSB
Offset Error	ADC Clock = 5 MHz	-2		2	LSB
Gain Error	ADC Clock = 5MHz	-2		2	LSB

9.11 SPI Characteristics

Figure 9-7. SPI Master mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

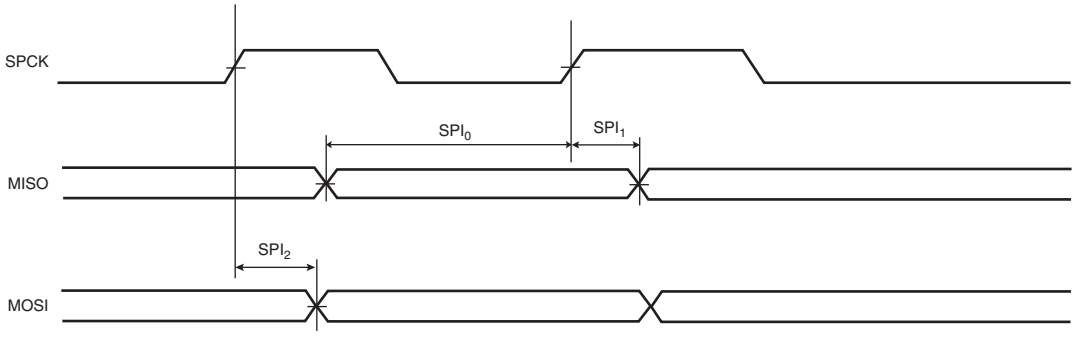


Figure 9-8. SPI Master mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

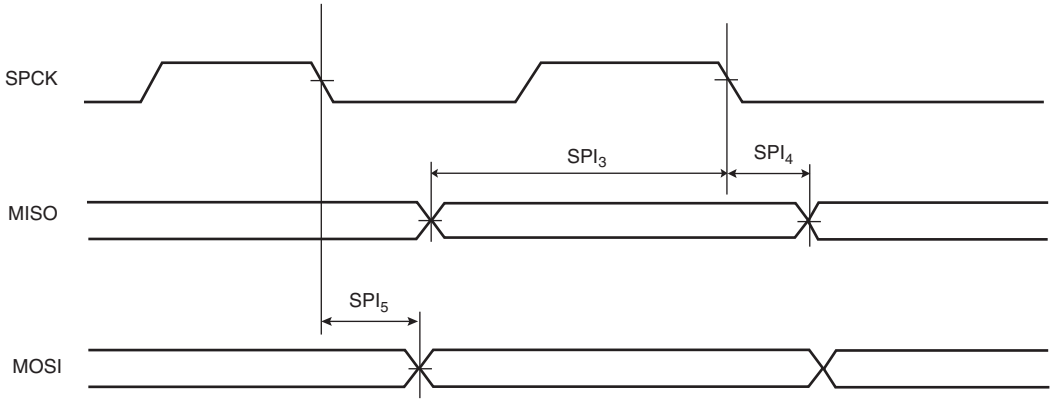
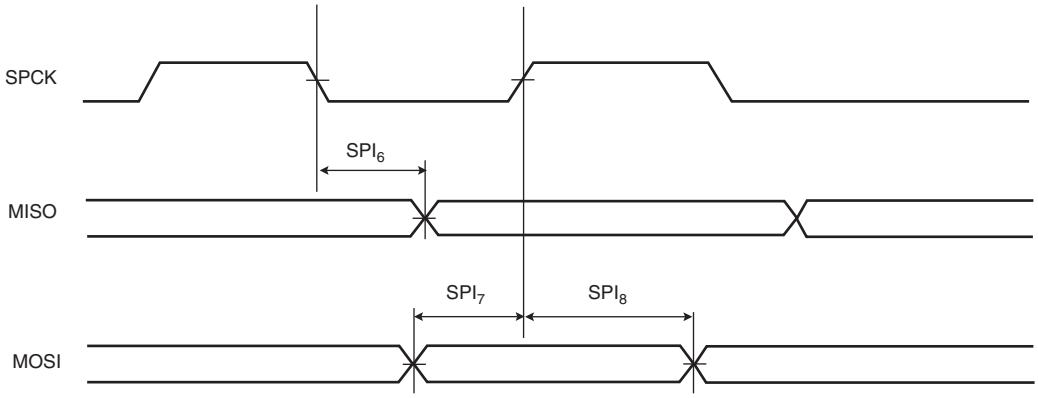


Figure 9-9. SPI Slave mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)



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.

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

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.

8. **Corruption after receiving too many bits in SPI slave mode**
   If the USART is in SPI slave mode and receives too much data bits (ex: 9bitsinstead of 8 bits) by the SPI master, an error occurs. After that, the next reception may be corrupted even if the frame is correct and the USART has been disabled, reset by a soft reset and re-enabled.
   Fix/Workaround
   None.

9. **USART slave synchronous mode external clock must be at least 9 times lower in frequency than CLK_USART**
   When the USART is operating in slave synchronous mode with an external clock, the frequency of the signal provided on CLK must be at least 9 times lower than CLK_USART.
   Fix/Workaround
   When the USART is operating in slave synchronous mode with an external clock, provide a signal on CLK that has a frequency at least 9 times lower than CLK_USART.

10. **HMATRIX**

11. **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.

- *DSP Operations*

1. **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.

12.1.2 Rev C

- *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.

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.

1. **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.

13. **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.

12.2.2 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.

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.

9. **Some GPIO VIH (input high voltage) are 3.6V max instead of 5V tolerant**
   Only 11 GPIOs remain 5V tolerant (VIHmax=5V):PB01, PB02, PB03, PB10, PB19, PB20, PB21, PB22, PB23, PB27, PB28.
   Fix/Workaround
   None.

10. **TC**

11. **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

2. **The command Quick Page Read User Page(QPRUP) is not functional**
   The command Quick Page Read User Page(QPRUP) is not functional.
   Fix/Workaround
   None.

3. **PAGEN Semantic Field for Program GP Fuse Byte is WriteData[7:0], ByteAddress[1:0] on revision B instead of WriteData[7:0], ByteAddress[2:0]**
   PAGEN Semantic Field for Program GP Fuse Byte is WriteData[7:0], ByteAddress[1:0] on revision B instead of WriteData[7:0], ByteAddress[2:0].
   Fix/Workaround
   None.

4. **Reading from on-chip flash may fail after a flash fuse write operation (FLASHC LP, UP, WGPB, EGPB, SSB, PGPFB, EAGPF commands).**
   After a flash fuse write operation (FLASHC LP, UP, WGPB, EGPB, SSB, PGPFB, EAGPF commands), the following flash read access may return corrupted data. This erratum does not affect write operations to regular flash memory.
   Fix/Workaround
   The flash fuse write operation (FLASHC LP, UP, WGPB, EGPB, SSB, PGPFB, EAGPF commands) must be issued from internal RAM. After the write operation, perform a dummy flash page write operation (FLASHC WP). Content and location of this page is not important and filling the write buffer with all one (FFh) will leave the current flash content unchanged. It is then safe to read and fetch code from the flash.

5.

- *RTC*

1. **Writes to control (CTRL), top (TOP) and value (VAL) in the RTC are discarded if the RTC peripheral bus clock (PBA) is divided by a factor of four or more relative to the HSB clock**
   Writes to control (CTRL), top (TOP) and value (VAL) in the RTC are discarded if the RTC peripheral bus clock (PBA) is divided by a factor of four or more relative to the HSB clock.
   Fix/Workaround
   Do not write to the RTC registers using the peripheral bus clock (PBA) divided by a factor of four or more relative to the HSB clock.

2. **The RTC CLKEN bit (bit number 16) of CTRL register is not available**
   The RTC CLKEN bit (bit number 16) of CTRL register is not available.
   Fix/Workaround
   Do not use the CLKEN bit of the RTC on Rev B.

Table of Contents