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	CANbus, Ethernet, I²C, IrDA, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	81
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 16x12b; D/A 4x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3c1256c-aut

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

The media-independent interface (MII) and reduced MII (RMII) 10/100 Ethernet MAC module provides on-chip solutions for network-connected devices.

The Peripheral Event Controller (PEVC) allows to redirect events from one peripheral or from input pins to another peripheral. It can then trigger, in a deterministic time, an action inside a peripheral without the need of CPU. For instance a PWM waveform can directly trigger an ADC capture, hence avoiding delays due to software interrupt processing.

The AT32UC3C features analog functions like ADC, DAC, Analog comparators. The ADC interface is built around a 12-bit pipelined ADC core and is able to control two independent 8-channel or one 16-channel. The ADC block is able to measure two different voltages sampled at the same time. The analog comparators can be paired to detect when the sensing voltage is within or outside the defined reference window.

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.

AT32UC3C 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 aWire- or JTAG-based debuggers. The single-pin aWire interface allows all features available through the JTAG interface to be accessed through the RESET pin, allowing the JTAG pins to be used for GPIO or peripherals.

2.2 Configuration Summary

Table 2-1. Configuration Summary

Feature	AT32UC3C0512C/ AT32UC3C0256C/ AT32UC3C0128C/ AT32UC3C064C	AT32UC3C1512C/ AT32UC3C1256C/ AT32UC3C1128C/ AT32UC3C164C	AT32UC3C2512C/ AT32UC3C2256C/ AT32UC3C2128C/ AT32UC3C264C
Flash	512/256/128/64 KB	512/256/128/64 KB	512/256/128/64 KB
SRAM	64/64/32/16 KB	64/64/32/16 KB	64/64/32/16 KB
HSB RAM	4 KB		
EBI	1	0	0
GPIO	123	81	45
External Interrupts	8	8	8
TWI	3	3	2
USART	5	5	4
Peripheral DMA Channels	16	16	16
Peripheral Event System	1	1	1
SPI	2	2	1
CAN channels	2	2	2
USB	1	1	1
Ethernet MAC 10/100	1 RMII/MII	1 RMII/MII	1 RMII only
I2S	1	1	1
Asynchronous Timers	1	1	1
Timer/Counter Channels	6	6	3
PWM channels	4x2		
QDEC	2	2	1
Frequency Meter	1		
Watchdog Timer	1		
Power Manager	1		
Oscillators	PLL 80-240 MHz (PLL0/PLL1) Crystal Oscillator 0.4-20 MHz (OSC0) Crystal Oscillator 32 KHz (OSC32K) RC Oscillator 115 kHz (RCSYS) RC Oscillator 8 MHz (RC8M) RC Oscillator 120 MHz (RC120M)		
	0.4-20 MHz (OSC1)		-
12-bit ADC number of channels	1 16	1 16	1 11
12-bit DAC number of channels	1 4	1 4	1 2

Figure 3-2. TQFP100 Pinout

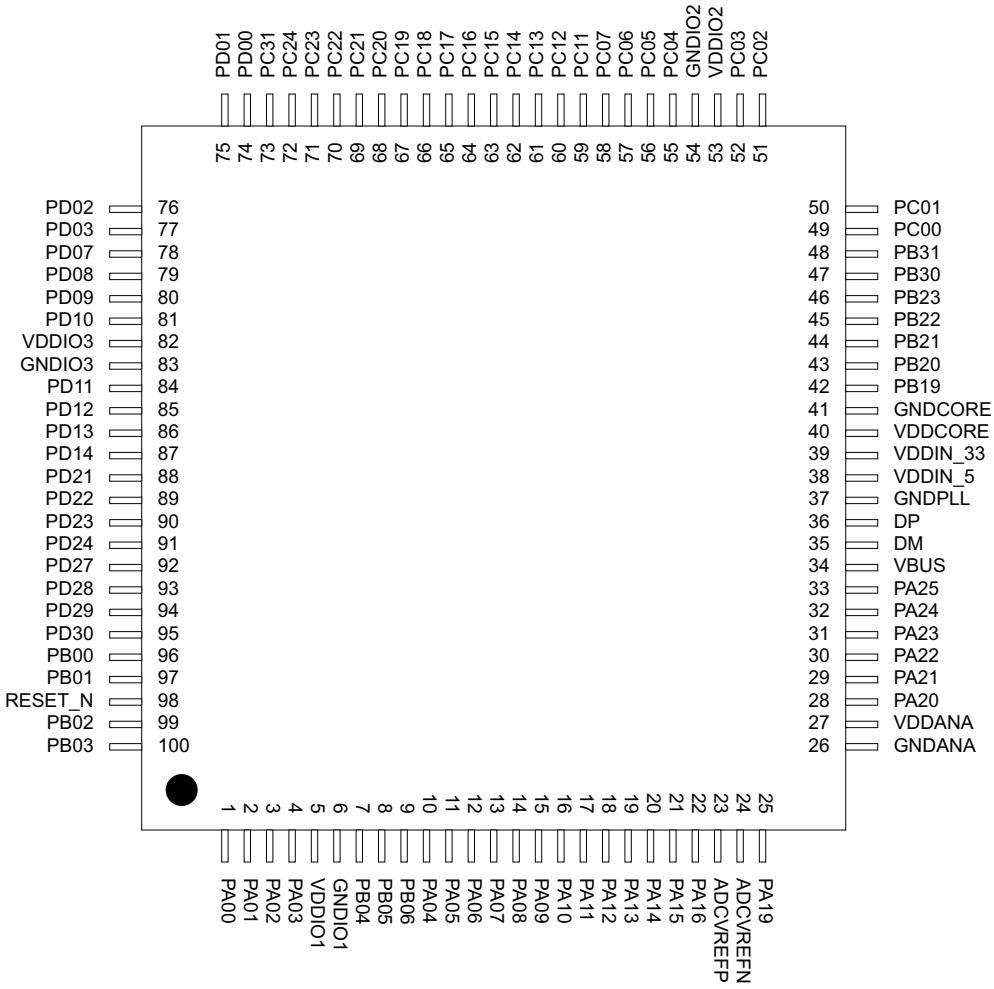


Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
16	25	36	PA19	19	VDDANA	x1/x2	ADCIN8	EIC - EXTINT[1]				
19	28	39	PA20	20	VDDANA	x1/x2	ADCIN9	AC0AP0	AC0AP0 or DAC0A			
20	29	40	PA21	21	VDDANA	x1/x2	ADCIN10	AC0BN0	AC0BN0 or DAC0B			
21	30	41	PA22	22	VDDANA	x1/x2	ADCIN11	AC0AN0	PEVC - PAD_EVT [4]		MACB - SPEED	
22	31	42	PA23	23	VDDANA	x1/x2	ADCIN12	AC0BP0	PEVC - PAD_EVT [5]		MACB - WOL	
	32	43	PA24	24	VDDANA	x1/x2	ADCIN13	SPI1 - NPCS[2]				
	33	44	PA25	25	VDDANA	x1/x2	ADCIN14	SPI1 - NPCS[3]	EIC - EXTINT[0]			
		45	PA26	26	VDDANA	x1/x2	AC0AP1	EIC - EXTINT[1]				
		46	PA27	27	VDDANA	x1/x2	AC0AN1	EIC - EXTINT[2]				
		47	PA28	28	VDDANA	x1/x2	AC0BP1	EIC - EXTINT[3]				
		48	PA29	29	VDDANA	x1/x2	AC0BN1	EIC - EXTINT[0]				
62	96	140	PB00	32	VDDIO1	x1	USART0 - CLK	CANIF - RXLINE[1]	EIC - EXTINT[8]	PEVC - PAD_EVT [10]		
63	97	141	PB01	33	VDDIO1	x1		CANIF - TXLINE[1]		PEVC - PAD_EVT [11]		
	99	143	PB02	34	VDDIO1	x1		USBC - ID	PEVC - PAD_EVT [6]	TC1 - A1		
	100	144	PB03	35	VDDIO1	x1		USBC - VBOF	PEVC - PAD_EVT [7]			
	7	7	PB04	36	VDDIO1	x1/x2	SPI1 - MOSI	CANIF - RXLINE[0]	QDEC1 - QEPI		MACB - TXD[2]	
	8	8	PB05	37	VDDIO1	x1/x2	SPI1 - MISO	CANIF - TXLINE[0]	PEVC - PAD_EVT [12]	USART3 - CLK	MACB - TXD[3]	
	9	9	PB06	38	VDDIO1	x2/x4	SPI1 - SCK		QDEC1 - QEPA	USART1 - CLK	MACB - TX_ER	
		10	PB07	39	VDDIO1	x1/x2	SPI1 - NPCS[0]	EIC - EXTINT[2]	QDEC1 - QEPB		MACB - RX_DV	
		11	PB08	40	VDDIO1	x1/x2	SPI1 - NPCS[1]	PEVC - PAD_EVT [1]	PWM - PWML[0]		MACB - RXD[0]	
		12	PB09	41	VDDIO1	x1/x2	SPI1 - NPCS[2]		PWM - PWMH[0]		MACB - RXD[1]	
		13	PB10	42	VDDIO1	x1/x2	USART1 - DTR	SPI0 - MOSI	PWM - PWML[1]			

Table 3-1. GPIO Controller Function Multiplexing

TQFP / QFN 64	TQFP 100	LQFP 144	PIN	GPIO	Supply	Pin Type (1)	GPIO function					
							A	B	C	D	E	F
33	51	73	PC02	66	VDDIO2	x1	TWIMS0 - TWD	SPI0 - NPCS[3]	USART2 - RXD	TC1 - CLK1	MACB - MDC	
34	52	74	PC03	67	VDDIO2	x1	TWIMS0 - TWCK	EIC - EXTINT[1]	USART2 - TXD	TC1 - B1	MACB - MDIO	
37	55	77	PC04	68	VDDIO2	x1	TWIMS1 - TWD	EIC - EXTINT[3]	USART2 - TXD	TC0 - B1		
38	56	78	PC05	69	VDDIO2	x1	TWIMS1 - TWCK	EIC - EXTINT[4]	USART2 - RXD	TC0 - A2		
	57	79	PC06	70	VDDIO2	x1	PEVC - PAD_EVT [15]	USART2 - CLK	USART2 - CTS	TC0 - CLK2	TWIMS2 - TWD	TWIMS0 - TWALM
	58	80	PC07	71	VDDIO2	x1	PEVC - PAD_EVT [2]	EBI - NCS[3]	USART2 - RTS	TC0 - B2	TWIMS2 - TWCK	TWIMS1 - TWALM
		81	PC08	72	VDDIO2	x1/x2	PEVC - PAD_EVT [13]	SPI1 - NPCS[1]	EBI - NCS[0]		USART4 - TXD	
		82	PC09	73	VDDIO2	x1/x2	PEVC - PAD_EVT [14]	SPI1 - NPCS[2]	EBI - ADDR[23]		USART4 - RXD	
		83	PC10	74	VDDIO2	x1/x2	PEVC - PAD_EVT [15]	SPI1 - NPCS[3]	EBI - ADDR[22]			
	59	84	PC11	75	VDDIO2	x1/x2	PWM - PWMH[3]	CANIF - RXLINE[1]	EBI - ADDR[21]	TC0 - CLK0		
	60	85	PC12	76	VDDIO2	x1/x2	PWM - PWML[3]	CANIF - TXLINE[1]	EBI - ADDR[20]	USART2 - CLK		
	61	86	PC13	77	VDDIO2	x1/x2	PWM - PWMH[2]	EIC - EXTINT[7]		USART0 - RTS		
	62	87	PC14	78	VDDIO2	x1/x2	PWM - PWML[2]	USART0 - CLK	EBI - SDCKE	USART0 - CTS		
39	63	88	PC15	79	VDDIO2	x1/x2	PWM - PWMH[1]	SPI0 - NPCS[0]	EBI - SDWE	USART0 - RXD	CANIF - RXLINE[1]	
40	64	89	PC16	80	VDDIO2	x1/x2	PWM - PWML[1]	SPI0 - NPCS[1]	EBI - CAS	USART0 - TXD	CANIF - TXLINE[1]	
41	65	90	PC17	81	VDDIO2	x1/x2	PWM - PWMH[0]	SPI0 - NPCS[2]	EBI - RAS	IISC - ISDO		USART3 - TXD
42	66	91	PC18	82	VDDIO2	x1/x2	PWM - PWML[0]	EIC - EXTINT[5]	EBI - SDA10	IISC - ISDI		USART3 - RXD
43	67	92	PC19	83	VDDIO3	x1/x2	PWM - PWML[2]	SCIF - GCLK[0]	EBI - DATA[0]	IISC - IMCK		USART3 - CTS
44	68	93	PC20	84	VDDIO3	x1/x2	PWM - PWMH[2]	SCIF - GCLK[1]	EBI - DATA[1]	IISC - ISCK		USART3 - RTS
45	69	94	PC21	85	VDDIO3	x1/x2	PWM - EXT_ FAULTS[0]	CANIF - RXLINE[0]	EBI - DATA[2]	IISC - IWS		
46	70	95	PC22	86	VDDIO3	x1/x2	PWM - EXT_ FAULTS[1]	CANIF - TXLINE[0]	EBI - DATA[3]		USART3 - CLK	
	71	96	PC23	87	VDDIO3	x1/x2	QDEC1 - QEPB	CANIF - RXLINE[1]	EBI - DATA[4]	PEVC - PAD_EVT [3]		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
VDDIN_5	1.8V Voltage Regulator Input	Power Input		Power Supply: 4.5V to 5.5V or 3.0V to 3.6 V
VDDIN_33	USB I/O power supply	Power Output/ Input		Capacitor Connection for the 3.3V voltage regulator or power supply: 3.0V to 3.6 V
VDDCORE	1.8V Voltage Regulator Output	Power output		Capacitor Connection for the 1.8V voltage regulator
GNDIO1 GNDIO2 GNDIO3	I/O Ground	Ground		
GNDANA	Analog Ground	Ground		
GNDCORE	Ground of the core	Ground		
GNDPLL	Ground of the PLLs	Ground		
Analog Comparator Interface - ACIFA0/1				
AC0AN1/AC0AN0	Negative inputs for comparator AC0A	Analog		
AC0AP1/AC0AP0	Positive inputs for comparator AC0A	Analog		
AC0BN1/AC0BN0	Negative inputs for comparator AC0B	Analog		
AC0BP1/AC0BP0	Positive inputs for comparator AC0B	Analog		
AC1AN1/AC1AN0	Negative inputs for comparator AC1A	Analog		
AC1AP1/AC1AP0	Positive inputs for comparator AC1A	Analog		
AC1BN1/AC1BN0	Negative inputs for comparator AC1B	Analog		
AC1BP1/AC1BP0	Positive inputs for comparator AC1B	Analog		
ACAOUT/ACBOUT	analog comparator outputs	output		
ADC Interface - ADCIFA				
ADCIN[15:0]	ADC input pins	Analog		
ADCREF0	Analog positive reference 0 voltage input	Analog		
ADCREF1	Analog positive reference 1 voltage input	Analog		
ADCVREFP	Analog positive reference connected to external capacitor	Analog		

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output		
SDWE	SDRAM Write Enable	Output	Low	
External Interrupt Controller - EIC				
EXTINT[8:1]	External Interrupt Pins	Input		
NMI_N = EXTINT[0]	Non-Maskable Interrupt Pin	Input	Low	
General Purpose Input/Output - GPIOA, GPIOB, GPIOC, GPIOD				
PA[29:19] - PA[16:0]	Parallel I/O Controller GPIOA	I/O		
PB[31:0]	Parallel I/O Controller GPIOB	I/O		
PC[31:0]	Parallel I/O Controller GPIOC	I/O		
PD[30:0]	Parallel I/O Controller GPIOD	I/O		
Inter-IC Sound (I2S) Controller - IISC				
IMCK	I2S Master Clock	Output		
ISCK	I2S Serial Clock	I/O		
ISDI	I2S Serial Data In	Input		
ISDO	I2S Serial Data Out	Output		
IWS	I2S Word Select	I/O		
JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		
Ethernet MAC - MACB				
COL	Collision Detect	Input		
CRS	Carrier Sense and Data Valid	Input		
MDC	Management Data Clock	Output		
MDIO	Management Data Input/Output	I/O		
RXD[3:0]	Receive Data	Input		

Table 4-3. System Registers (Continued)

Reg #	Address	Name	Function
24	96	JAVA_LV1	Unused in AVR32UC
25	100	JAVA_LV2	Unused in AVR32UC
26	104	JAVA_LV3	Unused in AVR32UC
27	108	JAVA_LV4	Unused in AVR32UC
28	112	JAVA_LV5	Unused in AVR32UC
29	116	JAVA_LV6	Unused in AVR32UC
30	120	JAVA_LV7	Unused in AVR32UC
31	124	JTBA	Unused in AVR32UC
32	128	JBCR	Unused in AVR32UC
33-63	132-252	Reserved	Reserved for future use
64	256	CONFIG0	Configuration register 0
65	260	CONFIG1	Configuration register 1
66	264	COUNT	Cycle Counter register
67	268	COMPARE	Compare register
68	272	TLBEHI	Unused in AVR32UC
69	276	TLBELO	Unused in AVR32UC
70	280	PTBR	Unused in AVR32UC
71	284	TLBEAR	Unused in AVR32UC
72	288	MMUCR	Unused in AVR32UC
73	292	TLBARLO	Unused in AVR32UC
74	296	TLBARHI	Unused in AVR32UC
75	300	PCCNT	Unused in AVR32UC
76	304	PCNT0	Unused in AVR32UC
77	308	PCNT1	Unused in AVR32UC
78	312	PCCR	Unused in AVR32UC
79	316	BEAR	Bus Error Address Register
80	320	MPUAR0	MPU Address Register region 0
81	324	MPUAR1	MPU Address Register region 1
82	328	MPUAR2	MPU Address Register region 2
83	332	MPUAR3	MPU Address Register region 3
84	336	MPUAR4	MPU Address Register region 4
85	340	MPUAR5	MPU Address Register region 5
86	344	MPUAR6	MPU Address Register region 6
87	348	MPUAR7	MPU Address Register region 7
88	352	MPUPSR0	MPU Privilege Select Register region 0
89	356	MPUPSR1	MPU Privilege Select Register region 1

Table 4-3. System Registers (Continued)

Reg #	Address	Name	Function
90	360	MPUPSR2	MPU Privilege Select Register region 2
91	364	MPUPSR3	MPU Privilege Select Register region 3
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	412	SS_STATUS	Secure State Status Register
104	416	SS_ADRF	Secure State Address Flash Register
105	420	SS_ADRR	Secure State Address RAM Register
106	424	SS_ADR0	Secure State Address 0 Register
107	428	SS_ADR1	Secure State Address 1 Register
108	432	SS_SP_SYS	Secure State Stack Pointer System Register
109	436	SS_SP_APP	Secure State Stack Pointer Application Register
110	440	SS_RAR	Secure State Return Address Register
111	444	SS_RSR	Secure State Return Status Register
112-191	448-764	Reserved	Reserved for future use
192-255	768-1020	IMPL	IMPLEMENTATION DEFINED

4.5 Exceptions and Interrupts

In the AVR32 architecture, events are used as a common term for exceptions and interrupts. AVR32UC incorporates a powerful event handling scheme. The different event sources, like Illegal Op-code and interrupt requests, have different priority levels, ensuring a well-defined behavior when multiple events are received simultaneously. Additionally, pending events of a higher priority class may preempt handling of ongoing events of a lower priority class.

When an event occurs, the execution of the instruction stream is halted, and execution is passed to an event handler at an address specified in Table 4-4 on page 38. 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 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 interrupt requests, yielding a uniform event handling scheme.

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

4.5.1 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.

The user must also make sure that the system stack is large enough so that any event is able to push the required registers to stack. If the system stack is full, and an event occurs, the system will enter an UNDEFINED state.

4.5.2 Exceptions and Interrupt Requests

When an event other than *scall* or debug request is received by the core, the following actions are performed atomically:

1. The pending event will not be accepted if it is masked. The I3M, I2M, I1M, I0M, EM, and GM bits in the Status Register are used to mask different events. Not all events can be masked. A few critical events (NMI, Unrecoverable Exception, TLB Multiple Hit, and Bus Error) can not be masked. When an event is accepted, hardware automatically sets the mask bits corresponding to all sources with equal or lower priority. This inhibits acceptance of other events of the same or lower priority, except for the critical events listed above. Software may choose to clear some or all of these bits after saving the necessary state if other priority schemes are desired. It is the event source's responsability to ensure that their events are left pending until accepted by the CPU.
2. When a request is accepted, the Status Register and Program Counter of the current context is stored to the system stack. If the event is an INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also automatically stored to stack. Storing the Status Register ensures that the core is returned to the previous execution mode when the current event handling is completed. When exceptions occur, both the EM and GM bits are set, and the application may manually enable nested exceptions if desired by clearing the appropriate bit. Each exception handler has a dedicated handler address, and this address uniquely identifies the exception source.
3. The Mode bits are set to reflect the priority of the accepted event, and the correct register file bank is selected. The address of the event handler, as shown in Table 4-4 on page 38, is loaded into the Program Counter.

The execution of the event handler routine then continues from the effective address calculated.

The *rete* instruction signals the end of the event. When encountered, the Return Status Register and Return Address Register are popped from the system stack and restored to the Status Register and Program Counter. If the *rete* instruction returns from INT0, INT1, INT2, or INT3, registers R8-R12 and LR are also popped from the system stack. The restored Status Register contains information allowing the core to resume operation in the previous execution mode. This concludes the event handling.

Table 7-6. Normal I/O Pin Characteristics[1]

Symbol	Parameter	Condition		Min	Typ	Max	Units
t_{RISE}	Rise time[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			7.7	ns
			load = 10pF, pin drive x2[2]			3.4	
			load = 10pF, pin drive x4[2]			1.9	
			load = 30pF, pin drive x1[2]			16	
			load = 30pF, pin drive x2[2]			7.5	
			load = 30pF, pin drive x4[2]			3.8	
		V_{VDD} = 4.5V	load = 10pF, pin drive x1[2]			5.3	
			load = 10pF, pin drive x2[2]			2.4	
			load = 10pF, pin drive x4[2]			1.3	
			load = 30pF, pin drive x1[2]			11.1	
			load = 30pF, pin drive x2[2]			5.2	
			load = 30pF, pin drive x4[2]			2.7	
t_{FALL}	Fall time[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			7.6	ns
			load = 10pF, pin drive x2[2]			3.5	
			load = 10pF, pin drive x4[2]			1.9	
			load = 30pF, pin drive x1[2]			15.8	
			load = 30pF, pin drive x2[2]			7.3	
			load = 30pF, pin drive x4[2]			3.8	
		V_{VDD} = 4.5V	load = 10pF, pin drive x1[2]			5.2	
			load = 10pF, pin drive x2[2]			2.4	
			load = 10pF, pin drive x4[2]			1.4	
			load = 30pF, pin drive x1[2]			10.9	
			load = 30pF, pin drive x2[2]			5.1	
			load = 30pF, pin drive x4[2]			2.7	
I_{LEAK}	Input leakage current	Pull-up resistors disabled				1.0	µA
C_{IN}	Input capacitance	PA00-PA29, PB00-PB31, PC00-PC01, PC08-PC31, PD00-PD30			7.5		pF
		PC02, PC03, PC04, PC05, PC06, PC07			2		

Note:
1. V_{VDD} corresponds to either V_{VDDIO1}, V_{VDDIO2}, V_{VDDIO3}, or V_{VDDANA}, depending on the supply for the pin. Refer to Section 3-1 on page 11 for details.
2. drive x1 capability pins are: PB00, PB01, PB02, PB03, PB30, PB31, PC02, PC03, PC04, PC05, PC06, PC07 - drive x2 /x4 capability pins are: PB06, PB21, PB26, PD02, PD06, PD13 - drive x1/x2 capability pins are the remaining PA, PB, PC, PD pins. The drive strength is programmable through ODCR0, ODCR0S, ODCR0C, ODCR0T registers of GPIO.
3. These values are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are not covered by test limits in production.

7.6.3 Phase Lock Loop (PLL0 and PLL1) Characteristics

Table 7-11. PLL Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{VCO}	Output frequency		80		240	MHz
f_{IN}	Input frequency		4		16	MHz
I_{PLL}	Current consumption	Active mode, f_{VCO} = 80MHz		250		µA
		Active mode, f_{VCO} = 240MHz		600		
$t_{STARTUP}$	Startup time, from enabling the PLL until the PLL is locked	Wide Bandwidth mode disabled		15		µs
		Wide Bandwidth mode enabled		45		

7.6.4 120MHz RC Oscillator (RC120M) Characteristics

Table 7-12. Internal 120MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency[1]		88	120	152	MHz
I_{RC120M}	Current consumption			1.85		mA
$t_{STARTUP}$	Startup time			3		µs

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

7.6.5 System RC Oscillator (RCSYS) Characteristics

Table 7-13. System RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	Calibrated at T_A = 85°C	110	115.2	120	kHz
		T_A = 25°C	105	109	115	
		T_A = -40°C	100	104	108	

7.6.6 8MHz/1MHz RC Oscillator (RC8M) Characteristics

Table 7-14. 8MHz/1MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OUT}	Output frequency	SCIF.RCCR8.FREQMODE = 0[1]	7.6	8	8.4	MHz
		SCIF.RCCR8.FREQMODE = 1[1]	0.955	1	1.045	
$t_{STARTUP}$	Startup time				20	µs

Notes: 1. Please refer to the SCIF chapter for details.

Table 7-34. ADC and S/H Transfer Characteristics 12-bit Resolution Mode and S/H gain = 1[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 0, S/H gain = 1 (F_{adc} = 1.2MHz)			12	Bit
INL	Integral Non-Linearity				5	LSB
DNL	Differential Non-Linearity				4	LSB
	Offset error		-5		5	mV
	Gain error		-20		20	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 0, S/H gain = 1 (F_{adc} = 1.5MHz)			12	Bit
INL	Integral Non-Linearity				5	LSB
DNL	Differential Non-Linearity				3	LSB
	Offset error		-10		10	mV
	Gain error		-20		20	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain.

Table 7-35. ADC and S/H Transfer Characteristics 12-bit Resolution Mode and S/H gain from 1 to 8[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 0, S/H gain from 1 to 8 (F_{adc} = 1.2MHz)			12	Bit
INL	Integral Non-Linearity				25	LSB
DNL	Differential Non-Linearity				25	LSB
	Offset error		-10		10	mV
	Gain error		-20		20	mV
RES	Resolution	Differential mode, V_{VDDANA} = 5V, $V_{ADCREF0}$ = 3V, ADCFIA.SEQCFGn.SRES = 0, S/H gain from 1 to 8 (F_{adc} = 1.5MHz)			12	Bit
INL	Integral Non-Linearity				9	LSB
DNL	Differential Non-Linearity				10	LSB
	Offset error		-15		15	mV
	Gain error		-20		20	mV

Note: 1. The measures are done without any I/O activity on VDDANA/GNDANA power domain

Table 7-36. ADC and S/H Transfer Characteristics 10-bit Resolution Mode and S/H gain from 1 to 16[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 1, S/H gain from 1 to 16 (F_{adc} = 1.5MHz)			10	Bit
INL	Integral Non-Linearity				3	LSB
DNL	Differential Non-Linearity				3	LSB
	Offset error		-15		15	mV
	Gain error		-20		20	mV

7.9.3 USART in SPI Mode Timing

7.9.3.1 Master mode

Figure 7-6. USART in SPI Master Mode With (CPOL= CPHA= 0) or (CPOL= CPHA= 1)

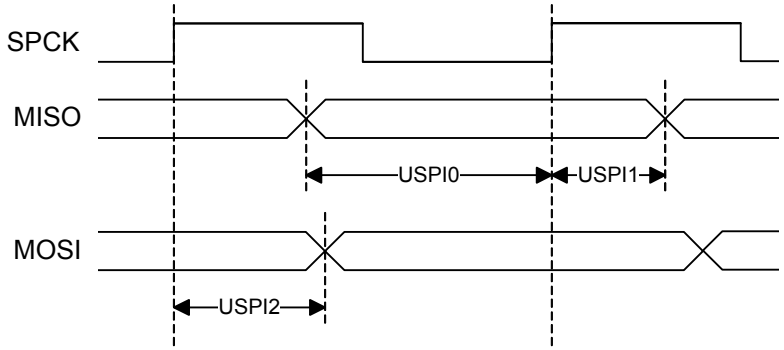


Figure 7-7. USART in SPI Master Mode With (CPOL= 0 and CPHA= 1) or (CPOL= 1 and CPHA= 0)

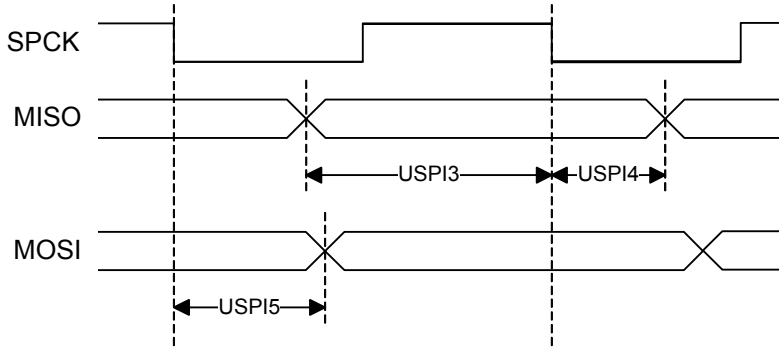


Table 7-46. USART in SPI Mode Timing, Master Mode[1]

Symbol	Parameter	Conditions	Min	Max	Units
USPI0	MISO setup time before SPCK rises	external capacitor = 40pF	26+ t_{SAMPLE}[2]		ns
USPI1	MISO hold time after SPCK rises		0		ns
USPI2	SPCK rising to MOSI delay			11	ns
USPI3	MISO setup time before SPCK falls		26+ t_{SAMPLE}[2]		ns
USPI4	MISO hold time after SPCK falls		0		ns
USPI5	SPCK falling to MOSI delay			11.5	ns

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

2. Where: $t_{SAMPLE} = t_{SPCK} - \left(\left\lfloor \frac{t_{SPCK}}{2 \times t_{CLKUSART}} \right\rfloor \frac{1}{2} \right) \times t_{CLKUSART}$

Maximum SPI Frequency, Master Output

The maximum SPI master output frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn}, \frac{f_{CLKSPI} \times 2}{9})$$

Where $SPIn$ is the MOSI delay, USPI2 or USPI5 depending on CPOL and NCPHA. f_{PINMAX} is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins. f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

Maximum SPI Frequency, Master Input

The maximum SPI master input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(\frac{1}{SPIn + t_{VALID}}, \frac{f_{CLKSPI} \times 2}{9})$$

Where $SPIn$ is the MISO setup and hold time, USPI0 + USPI1 or USPI3 + USPI4 depending on CPOL and NCPHA. T_{VALID} is the SPI slave response time. Please refer to the SPI slave datasheet for T_{VALID} . f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

7.9.3.2 Slave mode

Figure 7-8. USART in SPI Slave Mode With (CPOL= 0 and CPHA= 1) or (CPOL= 1 and CPHA= 0)

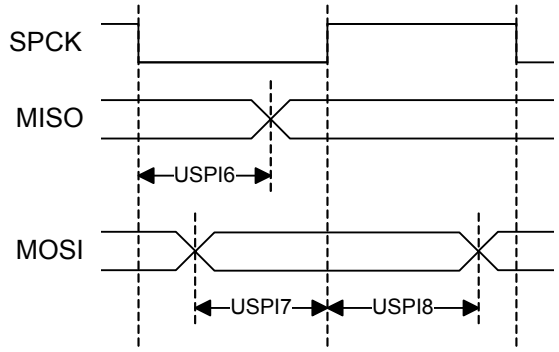


Table 7-49. SPI Timing, Slave Mode[1]

Symbol	Parameter	Conditions	Min	Max	Units
SPI6	SPCK falling to MISO delay	external capacitor = 40pF		29	ns
SPI7	MOSI setup time before SPCK rises		0		ns
SPI8	MOSI hold time after SPCK rises		6.5		ns
SPI9	SPCK rising to MISO delay			30	ns
SPI10	MOSI setup time before SPCK falls		0		ns
SPI11	MOSI hold time after SPCK falls		5		ns
SPI12	NPCS setup time before SPCK rises		0		ns
SPI13	NPCS hold time after SPCK falls		1.5		ns
SPI14	NPCS setup time before SPCK falls		0		ns
SPI15	NPCS hold time after SPCK rises		1.5		ns

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

Maximum SPI Frequency, Slave Input Mode

The maximum SPI slave input frequency is given by the following formula:

$$f_{SPCKMAX} = MIN(f_{CLKSPI}, \frac{1}{SPIn})$$

Where $SPIn$ is the MOSI setup and hold time, SPI7 + SPI8 or SPI10 + SPI11 depending on CPOL and NCPHA. f_{CLKSPI} is the maximum frequency of the CLK_SPI. Refer to the SPI chapter for a description of this clock.

Maximum SPI Frequency, Slave Output Mode

The maximum SPI slave output frequency is given by the following formula:

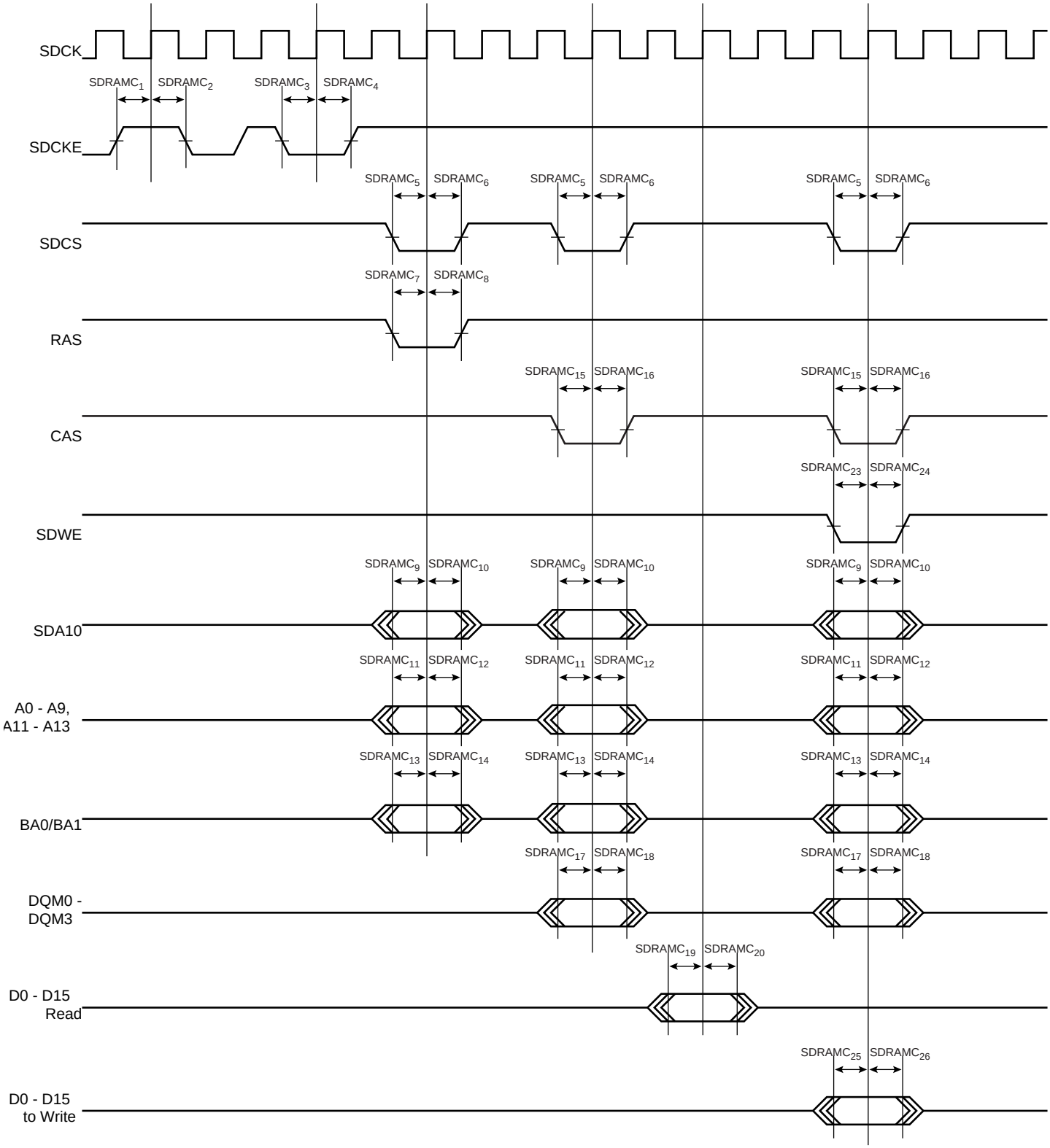
$$f_{SPCKMAX} = MIN(f_{PINMAX}, \frac{1}{SPIn + t_{SETUP}})$$

Where $SPIn$ is the MISO delay, SPI6 or SPI9 depending on CPOL and NCPHA. t_{SETUP} is the SPI master setup time. Please refer to the SPI masterdatasheet for t_{SETUP} .f_{PINMAX} is the maximum frequency of the SPI pins. Please refer to the I/O Pin Characteristics section for the maximum frequency of the pins.

7.9.5 TWIM/TWIS Timing

Figure 7-50 shows the TWI-bus timing requirements and the compliance of the device with them. Some of these requirements (t_r and t_f) are met by the device without requiring user intervention. Compliance with the other requirements (t_{HD-STA}, t_{SU-STA}, t_{SU-STO}, t_{HD-DAT}, $t_{SU-DAT-I2C}$, $t_{LOW-I2C}$, t_{HIGH}, and f_{TWCK}) requires user intervention through appropriate programming of the relevant

Figure 7-19. SDRAMC Signals relative to SDCK.



8. Mechanical Characteristics

8.1 Thermal Considerations

8.1.1 Thermal Data

Table 8-1 summarizes the thermal resistance data depending on the package.

Table 8-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	QFN64	20.0	°C/W
θ_{JC}	Junction-to-case thermal resistance		QFN64	0.8	
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	TQFP64	40.5	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP64	8.7	
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	TQFP100	39.3	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP100	8.5	
θ_{JA}	Junction-to-ambient thermal resistance	No air flow	LQFP144	38.1	°C/W
θ_{JC}	Junction-to-case thermal resistance		LQFP144	8.4	

8.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

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

where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 8-1 on page 90.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 8-1 on page 90.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section "Power Consumption" on page 51.
- T_A = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T_J in °C.

10.1.5 SCIF

1 **PLLCOUNT value larger than zero can cause PLLEN glitch**
Initializing the PLLCOUNT with a value greater than zero creates a glitch on the PLLEN signal during asynchronous wake up.
Fix/Workaround
The lock-masking mechanism for the PLL should not be used.
The PLLCOUNT field of the PLL Control Register should always be written to zero.

2 **PLL lock might not clear after disable**
Under certain circumstances, the lock signal from the Phase Locked Loop (PLL) oscillator may not go back to zero after the PLL oscillator has been disabled. This can cause the propagation of clock signals with the wrong frequency to parts of the system that use the PLL clock.
Fix/Workaround
PLL must be turned off before entering STOP, DEEPSTOP or STATIC sleep modes. If PLL has been turned off, a delay of 30us must be observed after the PLL has been enabled again before the SCIF.PLL0LOCK bit can be used as a valid indication that the PLL is locked.

3 **BOD33 reset locks the device**
If BOD33 is enabled as a reset source (SCIF.BOD33.CTRL=0x1) and when VDDIN_33 power supply voltage falls below the BOD33 voltage (SCIF.BOD33.LEVEL), the device is locked permanently under reset even if the power supply goes back above BOD33 reset level. In order to unlock the device, an external reset event should be applied on RESET_N.
Fix/Workaround
Use an external BOD on VDDIN_33 or an external reset source.

10.1.6 SPI

1 **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.

2 **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.

3 **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).

Table of Contents