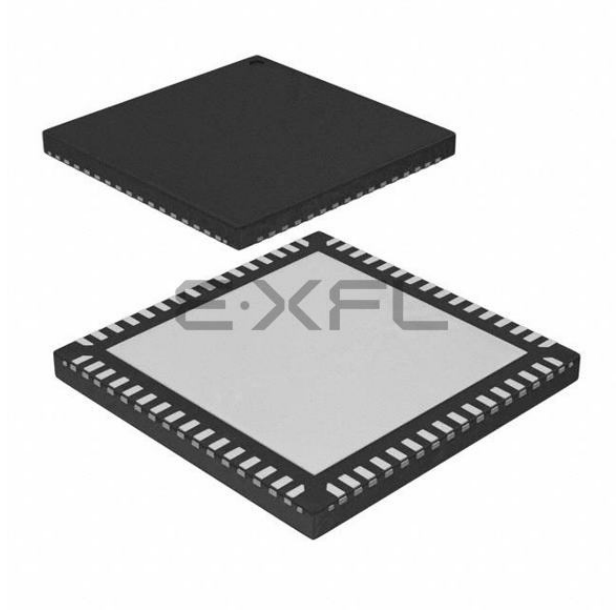


Welcome to E-XFL.COM

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	45
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 11x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°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/at32uc3c2128c-z2zt

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]		

3.2.2 Peripheral Functions

Each GPIO line can be assigned to one of several peripheral functions. The following table describes how the various peripheral functions are selected. The last listed function has priority in case multiple functions are enabled on the same pin.

Table 3-2. Peripheral Functions

Function	Description
GPIO Controller Function multiplexing	GPIO and GPIO peripheral selection A to F
Nexus OCD AUX port connections	OCD trace system
aWire DATAOUT	aWire output in two-pin mode
JTAG port connections	JTAG debug port
Oscillators	OSC0, OSC32

3.2.3 Oscillator Pinout

The oscillators are not mapped to the normal GPIO functions and their muxings are controlled by registers in the System Control Interface (SCIF). Please refer to the SCIF chapter for more information about this.

Table 3-3. Oscillator pinout

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pad	Oscillator pin
31	47	69	PB30	xin0
	99	143	PB02	xin1
62	96	140	PB00	xin32
32	48	70	PB31	xout0
	100	144	PB03	xout1
63	97	141	PB01	xout32

3.2.4 JTAG port connections

If the JTAG is enabled, the JTAG will take control over a number of pins, irrespectively of the I/O Controller configuration.

Table 3-4. JTAG pinout

QFN64/ TQFP64 pin	TQFP100 pin	LQFP144 pin	Pin name	JTAG pin
2	2	2	PA01	TDI
3	3	3	PA02	TDO
4	4	4	PA03	TMS
1	1	1	PA00	TCK

3.2.5 Nexus OCD AUX port connections

If the OCD trace system is enabled, the trace system will take control over a number of pins, irrespectively of the GPIO configuration. Three different OCD trace pin mappings are possible,

Table 3-7. Signal Description List

Signal Name	Function	Type	Active Level	Comments
DP	USB Device Port Data +	Analog		
VBUS	USB VBUS Monitor and OTG Negociation	Analog Input		
ID	ID Pin of the USB Bus	Input		
VBOF	USB VBUS On/off: bus power control port	output		

3.4 I/O Line Considerations

3.4.1 JTAG pins

The JTAG is enabled if TCK is low while the RESET_N pin is released. The TCK, TMS, and TDI pins have pull-up resistors when JTAG is enabled. The TCK pin always have pull-up enabled during reset. The TDO pin is an output, driven at VDDIO1, and has no pull-up resistor. The JTAG pins can be used as GPIO pins and muxed with peripherals when the JTAG is disabled. Please refer to Section 3.2.4 for the JTAG port connections.

3.4.2 RESET_N pin

The RESET_N pin integrates a pull-up resistor to VDDIO1. 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.

The RESET_N pin is also used for the aWire debug protocol. When the pin is used for debugging, it must not be driven by external circuitry.

3.4.3 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.

3.4.4 GPIO pins

All I/O lines integrate programmable pull-up and pull-down resistors. Most I/O lines integrate drive strength control, see Table 3-1. Programming of this pull-up and pull-down resistor or this drive strength is performed independently for each I/O line through the GPIO Controllers.

After reset, I/O lines default as inputs with pull-up/pull-down resistors disabled. After reset, output drive strength is configured to the lowest value to reduce global EMI of the device.

When the I/O line is configured as analog function (ADC I/O, AC inputs, DAC I/O), the pull-up and pull-down resistors are automatically disabled.

single cycle. Load and store instructions have several different formats in order to reduce code size and speed up execution.

The register file is organized as sixteen 32-bit registers and includes the Program Counter, the Link Register, and the Stack Pointer. In addition, register R12 is designed to hold return values from function calls and is used implicitly by some instructions.

4.3 The AVR32UC CPU

The AVR32UC CPU targets low- and medium-performance applications, and provides an advanced On-Chip Debug (OCD) system, no caches, and a Memory Protection Unit (MPU). A hardware Floating Point Unit (FPU) is also provided through the coprocessor instruction space. Java acceleration hardware is not implemented.

AVR32UC provides three memory interfaces, one High Speed Bus master for instruction fetch, one High Speed Bus master for data access, and one High Speed Bus slave interface allowing other bus masters to access data RAMs internal to the CPU. Keeping data RAMs internal to the CPU allows fast access to the RAMs, reduces latency, and guarantees deterministic timing. Also, power consumption is reduced by not needing a full High Speed Bus access for memory accesses. A dedicated data RAM interface is provided for communicating with the internal data RAMs.

A local bus interface is provided for connecting the CPU to device-specific high-speed systems, such as floating-point units and I/O controller ports. This local bus has to be enabled by writing a one to the LOCEN bit in the CPUCR system register. The local bus is able to transfer data between the CPU and the local bus slave in a single clock cycle. The local bus has a dedicated memory range allocated to it, and data transfers are performed using regular load and store instructions. Details on which devices that are mapped into the local bus space is given in the CPU Local Bus section in the Memories chapter.

Figure 4-1 on page 27 displays the contents of AVR32UC.

4.4 Programming Model

4.4.1 Register File Configuration

The AVR32UC register file is shown below.

Figure 4-3. The AVR32UC Register File

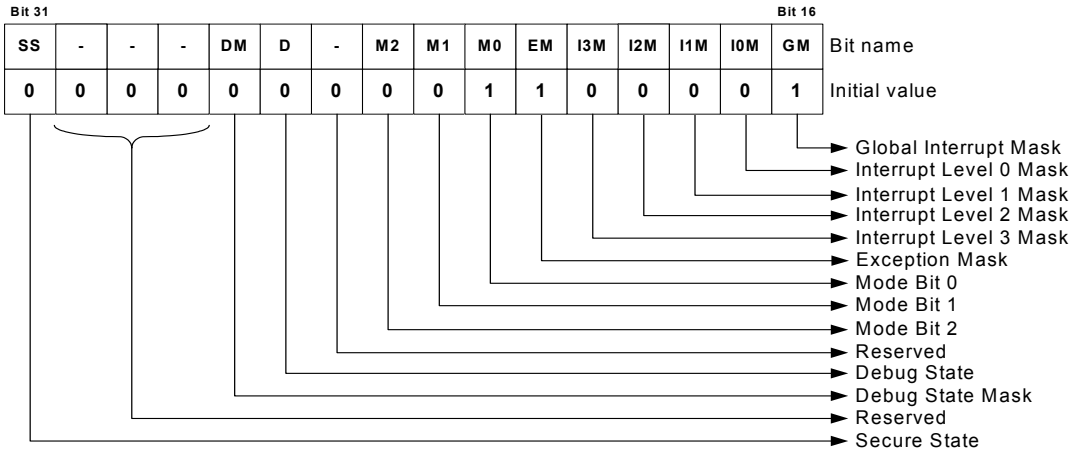
Application	Supervisor	INT0	INT1	INT2	INT3	Exception	NMI	Secure
Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0	Bit 31 Bit 0
PC	PC	PC	PC	PC	PC	PC	PC	PC
LR	LR	LR	LR	LR	LR	LR	LR	LR
SP_APP	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SYS	SP_SEC
R12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

SS_STATUS
SS_ADRF
SS_ADRR
SS_ADR0
SS_ADR1
SS_SP_SYS
SS_SP_APP
SS_RAR
SS_RSR

4.4.2 Status Register Configuration

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

Figure 4-4. The Status Register High Halfword



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 5-3. Peripheral Address Mapping

0xFFFD1000	MDMA	Memory DMA - MDMA
0xFFFD1400	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1
0xFFFD1800	SPI0	Serial Peripheral Interface - SPI0
0xFFFD1C00	CANIF	Control Area Network interface - CANIF
0xFFFD2000	TC0	Timer/Counter - TC0
0xFFFD2400	ADCIFA	ADC controller interface with Touch Screen functionality - ADCIFA
0xFFFD2800	USART4	Universal Synchronous/Asynchronous Receiver/Transmitter - USART4
0xFFFD2C00	TWIM2	Two-wire Master Interface - TWIM2
0xFFFD3000	TWIS2	Two-wire Slave Interface - TWIS2
0xFFFE0000	HFLASHC	Flash Controller - HFLASHC
0xFFFE1000	USBC	USB 2.0 OTG Interface - USBC
0xFFFE2000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE2400	SAU	Secure Access Unit - SAU
0xFFFE2800	SMC	Static Memory Controller - SMC
0xFFFE2C00	SDRAMC	SDRAM Controller - SDRAMC
0xFFFE3000	MACB	Ethernet MAC - MACB
0xFFFF0000	INTC	Interrupt controller - INTC
0xFFFF0400	PM	Power Manager - PM
0xFFFF0800	SCIF	System Control Interface - SCIF

  - Internal 3.3V regulator is off
- T_A = 25°C
- I/Os are configured as inputs, with internal pull-up enabled.
- Oscillators
  - OSC0/1 (crystal oscillator) stopped
  - OSC32K (32KHz crystal oscillator) stopped
  - PLL0 running
  - PLL1 stopped
- Clocks
  - External clock on XIN0 as main clock source **(**10MHz**)**
  - CPU, HSB, and PBB clocks undivided
  - PBA, PBC clock divided by 4
  - All peripheral clocks running

Table 7-4. Power Consumption for Different Operating Modes

Mode	Conditions	Measured on	Consumption Typ	Unit
Active[1]	CPU running a recursive Fibonacci algorithm	Amp	512	µA/MHz
Idle[1]			258	
Frozen[1]			106	
Standby[1]			48	
Stop			73	µA
DeepStop			43	
Static	OSC32K and AST running		32	
	AST and OSC32K stopped		31	

Note: 1. These numbers are valid for the measured condition only and must not be extrapolated to other frequencies.

7.5 I/O Pin Characteristics

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

Symbol	Parameter	Condition		Min	Typ	Max	Units
R_{PULLUP}	Pull-up resistance	V_{VDD} = 3V		5		26	kOhm
		V_{VDD} = 5V		5		16	kOhm
$R_{PULLDOWN}$	Pull-down resistance			2		16	kOhm
V_{IL}	Input low-level voltage	V_{VDD} = 3V				$0.3*V_{VDDIO}$	V
		V_{VDD} = 4.5V				$0.3*V_{VDDIO}$	
V_{IH}	Input high-level voltage	V_{VDD} = 3.6V		$0.7*V_{VDDIO}$			V
		V_{VDD} = 5.5V		$0.7*V_{VDDIO}$			
V_{OL}	Output low-level voltage	I_{OL} = -3.5mA, pin drive x1[2]				0.5	V
		I_{OL} = -7mA, pin drive x2[2]					
		I_{OL} = -14mA, pin drive x4[2]					
V_{OH}	Output high-level voltage	I_{OH} = 3.5mA, pin drive x1[2]		V_{VDD} - 0.8			V
		I_{OH} = 7mA, pin drive x2[2]					
		I_{OH} = 14mA, pin drive x4[2]					
f_{MAX}	Output frequency[3]	V_{VDD} = 3.0V	load = 10pF, pin drive x1[2]			30	MHz
			load = 10pF, pin drive x2[2]			50	
			load = 10pF, pin drive x4[2]			60	
			load = 30pF, pin drive x1[2]			15	
			load = 30pF, pin drive x2[2]			25	
			load = 30pF, pin drive x4[2]			40	
		V_{VDD} =4.5V	load = 10pF, pin drive x1[2]			45	
			load = 10pF, pin drive x2[2]			65	
			load = 10pF, pin drive x4[2]			85	
			load = 30pF, pin drive x1[2]			20	
			load = 30pF, pin drive x2[2]			40	
			load = 30pF, pin drive x4[2]			60	

7.7 Flash Characteristics

Table 7-15 gives the device maximum operating frequency depending on the number of flash wait states. The FSW bit in the FLASHC FSR register controls the number of wait states used when accessing the flash memory.

Table 7-15. Maximum Operating Frequency

Flash Wait States	Read Mode	Maximum Operating Frequency
0	1 cycle	25 MHz
1	2 cycles	50 MHz

Table 7-16. Flash Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FPP}	Page programming time	f_{CLK_HSB} = 50 MHz		17		ms
t_{FPE}	Page erase time			17		
t_{FFP}	Fuse programming time			1.3		
t_{FEA}	Full chip erase time (EA)			18.3		
t_{FCE}	JTAG chip erase time (CHIP_ERASE)	f_{CLK_HSB} = 115 kHz		640		

Table 7-17. Flash Endurance and Data Retention

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
N_{FARRAY}	Array endurance (write/page)		10 k			cycles
N_{FFUSE}	General Purpose fuses endurance (write/bit)		500			cycles
t_{RET}	Data retention		15			years

Table 7-29. ADC Decoupling requirements

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$C_{ADCREFPN}$	ADCREFP/ADCREFN capacitance	No voltage reference appplied on ADCREFP/ADCREFN		100		nF

Table 7-30. ADC Inputs

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{ADCINn}	ADC input voltage range		0		V_{VDDANA}	V
C_{ONCHIP}	Internal Capacitance	ADC used without S/H			5	pF
		ADC used with S/H			4	
R_{ONCHIP}	Switch resistance	ADC used without S/H			5.1	kΩ
		ADC used with S/H			4.6	

Figure 7-3. ADC input

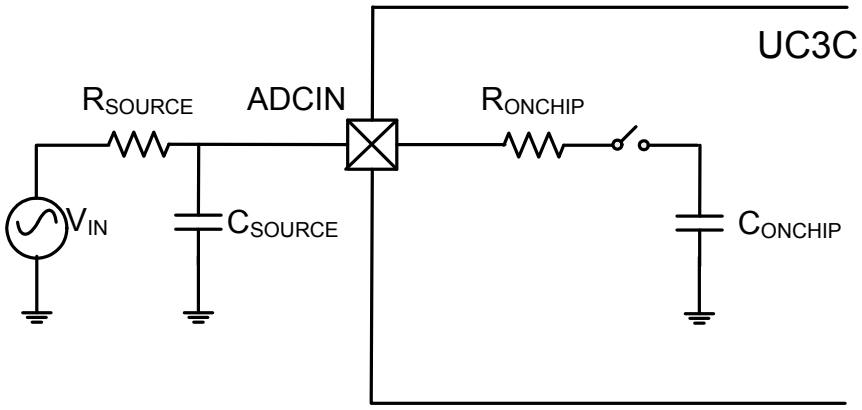


Table 7-31. ADC Transfer Characteristics 12-bit Resolution Mode[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Units
RES	Resolution	Differential mode, V_{VDDANA} = 3V, $V_{ADCREF0}$ = 1V, ADCFIA.SEQCFGn.SRES = 0 (F_{adc} = 1.2MHz)			12	Bit
INL	Integral Non-Linearity				6	LSB
DNL	Differential Non-Linearity				5	LSB
	Offset error		-10		10	mV
	Gain error		-30		30	mV

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				6	LSB
DNL	Differential Non-Linearity				5	LSB
	Offset error		-10		10	mV
	Gain error		-30		30	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				6	LSB
DNL	Differential Non-Linearity				4	LSB
	Offset error		-15		15	mV
	Gain error		-30		30	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				30	LSB
DNL	Differential Non-Linearity				30	LSB
	Offset error		-10		10	mV
	Gain error		-25		25	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				10	LSB
DNL	Differential Non-Linearity				15	LSB
	Offset error		-20		20	mV
	Gain error		-30		30	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				4	LSB
DNL	Differential Non-Linearity				4	LSB
	Offset error		-15		15	mV
	Gain error		-25		25	mV

7.9.4.2 Slave mode

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

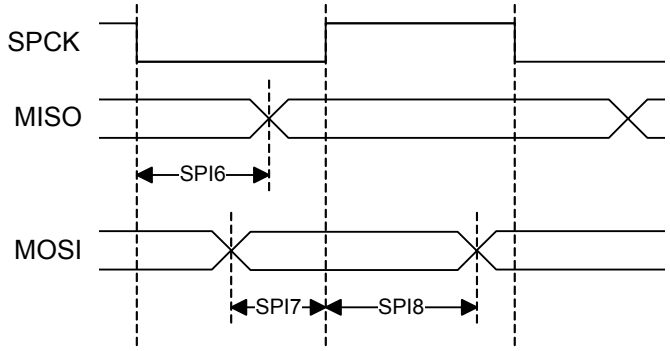


Figure 7-14. SPI Slave Mode With (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)

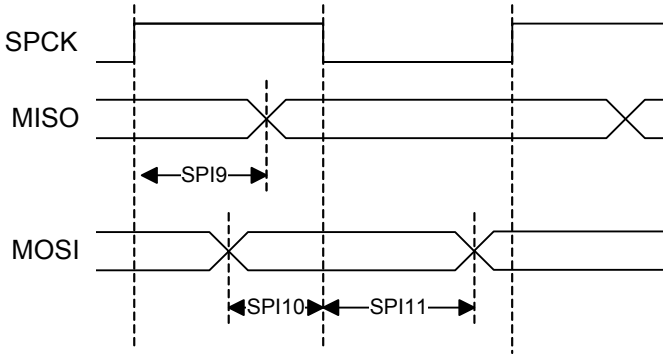


Figure 7-15. SPI Slave Mode NPCS Timing

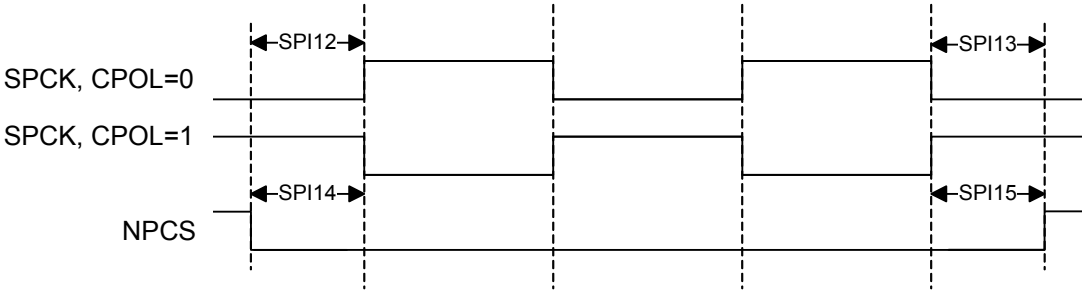


Table 7-56. SMC Write Signals with No Hold Settings (NWE Controlled only)[1]

Symbol	Parameter	Conditions	Min	Units
SMC_{37}	NWE rising to A2-A25 valid	V_{VDD} = 3.0V, drive strength of the pads set to the lowest, external capacitor = 40pF	9.1	ns
SMC_{38}	NWE rising to NBS0/A0 valid		7.9	
SMC_{40}	NWE rising to A1/NBS2 change		9.1	
SMC_{42}	NWE rising to NCS rising		8.7	
SMC_{43}	Data Out valid before NWE rising		(nwe pulse length - 1) * t_{CPSMC} - 1.5	
SMC_{44}	Data Out valid after NWE rising		8.7	
SMC_{45}	NWE pulse width		nwe pulse length * t_{CPSMC} - 0.1	

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.

Figure 7-17. SMC Signals for NCS Controlled Accesses

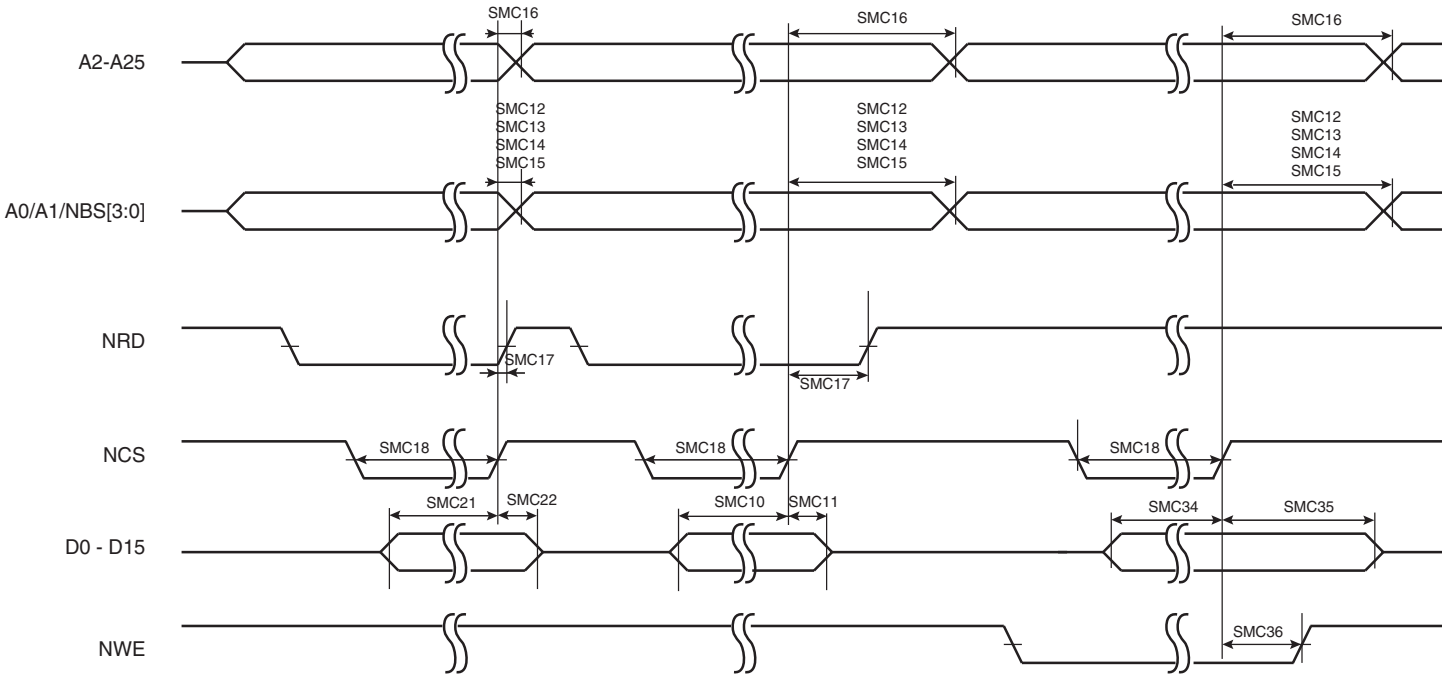
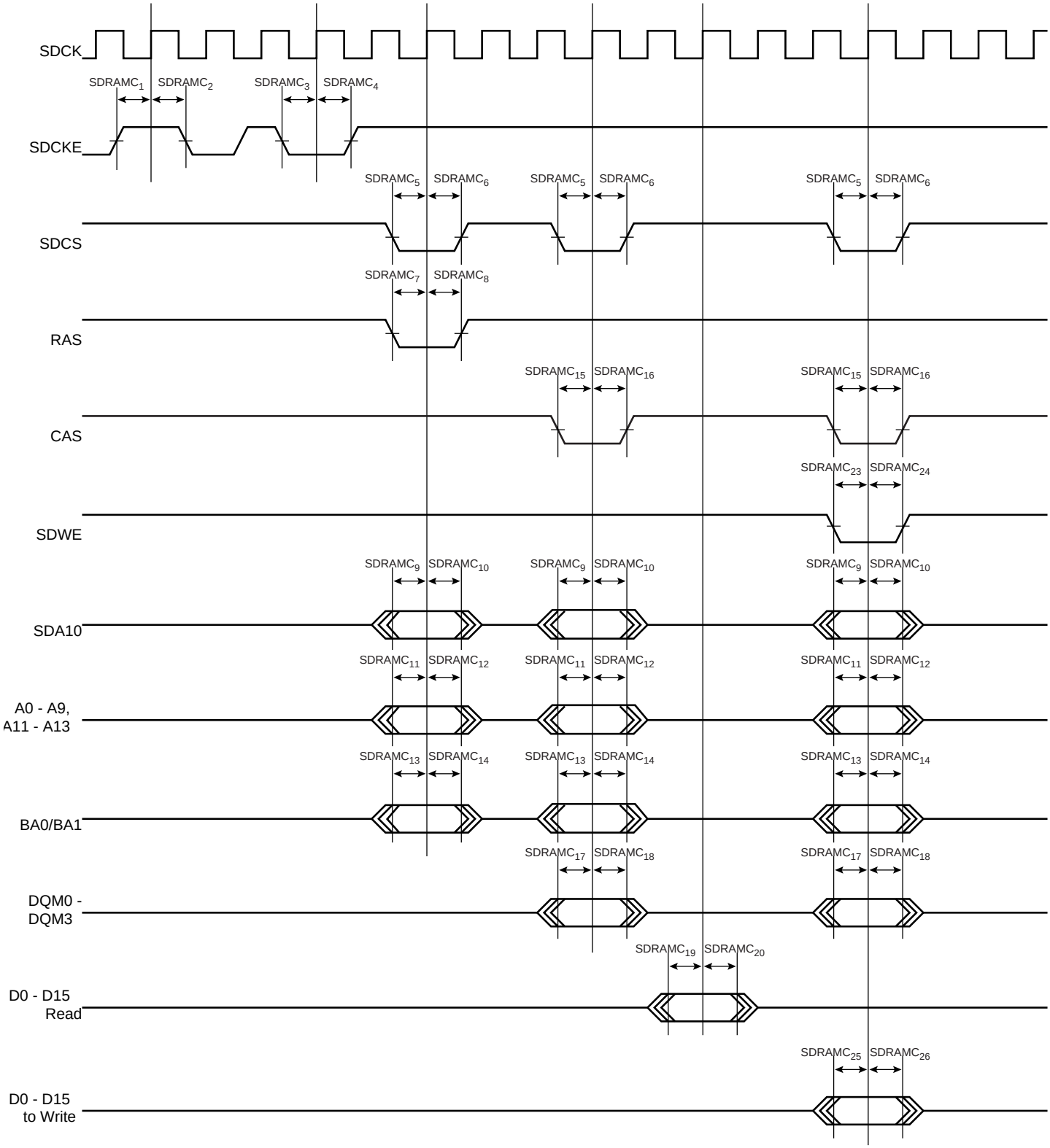


Figure 7-19. SDRAMC Signals relative to SDCK.



8.3 Soldering Profile

Table 8-14 gives the recommended soldering profile from J-STD-20.

Table 8-14. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec
Preheat Temperature 175°C ±25°C	Min. 150 °C, Max. 200 °C
Temperature Maintained Above 217°C	60-150 sec
Time within 5·C of Actual Peak Temperature	30 sec
Peak Temperature Range	260 °C
Ramp-down Rate	6 °C/sec
Time 25·C to Peak Temperature	Max. 8 minutes

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

10. Errata

10.1 rev E

10.1.1 ADCIFA

1 **ADCREFP/ADCREFN can not be selected as an external ADC reference by setting the ADCIFA.CFG.EXREF bit to one**
Fix/Workaround
A voltage reference can be applied on ADCREFP/ADCREFN pins if the ADCIFA.CFG.EXREF bit is set to zero, the ADCIFA.CFG.RS bit is set to zero and the voltage reference applied on ADCREFP/ADCREFN pins is higher than the internal 1V reference.

10.1.2 AST

1 **AST wake signal is released one AST clock cycle after the BUSY bit is cleared**
After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
Fix/Workaround
Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

10.1.3 aWire

1 **aWire MEMORY_SPEED_REQUEST command does not return correct CV**
The aWire MEMORY_SPEED_REQUEST command does not return a CV corresponding to the formula in the aWire Debug Interface chapter.
Fix/Workaround
Issue a dummy read to address 0x100000000 before issuing the MEMORY_SPEED_REQUEST command and use this formula instead:

$$f_{sab} = \frac{7 f_{aw}}{CV - 3}$$

10.1.4 Power Manager

1 **TWIS may not wake the device from sleep mode**
If the CPU is put to a sleep mode (except Idle and Frozen) directly after a TWI Start condition, the CPU may not wake upon a TWIS address match. The request is NACKed.
Fix/Workaround
When using the TWI address match to wake the device from sleep, do not switch to sleep modes deeper than Frozen. Another solution is to enable asynchronous EIC wake on the TWIS clock (TWCK) or TWIS data (TWD) pins, in order to wake the system up on bus events.

10.2 rev D

10.2.1 ADCIFA

1. **ADCREFP/ADCREFN can not be selected as an external ADC reference by setting the ADCIFA.CFG.EXREF bit to one**
   Fix/Workaround
   A voltage reference can be applied on ADCREFP/ADCREFN pins if the ADCIFA.CFG.EXREF bit is set to zero, the ADCIFA.CFG.RS bit is set to zero and the voltage reference applied on ADCREFP/ADCREFN pins is higher than the internal 1V reference.

10.2.2 AST

1. **AST wake signal is released one AST clock cycle after the BUSY bit is cleared**
   After writing to the Status Clear Register (SCR) the wake signal is released one AST clock cycle after the BUSY bit in the Status Register (SR.BUSY) is cleared. If entering sleep mode directly after the BUSY bit is cleared the part will wake up immediately.
   Fix/Workaround
   Read the Wake Enable Register (WER) and write this value back to the same register. Wait for BUSY to clear before entering sleep mode.

10.2.3 aWire

1. **aWire MEMORY_SPEED_REQUEST command does not return correct CV**
   The aWire MEMORY_SPEED_REQUEST command does not return a CV corresponding to the formula in the aWire Debug Interface chapter.
   Fix/Workaround
   Issue a dummy read to address 0x100000000 before issuing the MEMORY_SPEED_REQUEST command and use this formula instead:

$$f_{sab} = \frac{7 f_{aw}}{CV - 3}$$

10.2.4 GPIO

1. **Clearing Interrupt flags can mask other interrupts**
   When clearing interrupt flags in a GPIO port, interrupts on other pins of that port, happening in the same clock cycle will not be registered.
   Fix/Workaround
   Read the PVR register of the port before and after clearing the interrupt to see if any pin change has happened while clearing the interrupt. If any change occurred in the PVR between the reads, they must be treated as an interrupt.

10.2.5 Power Manager

1. **Clock Failure Detector (CFD) can be issued while turning off the CFD**
   While turning off the CFD, the CFD bit in the Status Register (SR) can be set. This will change the main clock source to RCSYS.
   Fix/Workaround
   Solution 1: Enable CFD interrupt. If CFD interrupt is issues after turning off the CFD, switch back to original main clock source.
   Solution 2: Only turn off the CFD while running the main clock on RCSYS.

2 **Requesting clocks in idle sleep modes will mask all other PB clocks than the requested**
In idle or frozen sleep mode, all the PB clocks will be frozen if the TWIS or the AST need to wake the cpu up.
Fix/Workaround
Disable the TWIS or the AST before entering idle or frozen sleep mode.

3 **TWIS may not wake the device from sleep mode**
If the CPU is put to a sleep mode (except Idle and Frozen) directly after a TWI Start condition, the CPU may not wake upon a TWIS address match. The request is NACKed.
Fix/Workaround
When using the TWI address match to wake the device from sleep, do not switch to sleep modes deeper than Frozen. Another solution is to enable asynchronous EIC wake on the TWIS clock (TWCK) or TWIS data (TWD) pins, in order to wake the system up on bus events.

10.2.6 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.

10.2.7 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.

10.2.10 TWIS

1. **Clearing the NAK bit before the BTF bit is set locks up the TWI bus**
When the TWIS is in transmit mode, clearing the NAK Received (NAK) bit of the Status Register (SR) before the end of the Acknowledge/Not Acknowledge cycle will cause the TWIS to attempt to continue transmitting data, thus locking up the bus.
Fix/Workaround
Clear SR.NAK only after the Byte Transfer Finished (BTF) bit of the same register has been set.

2. **TWIS stretch on Address match error**
When the TWIS stretches TWCK due to a slave address match, it also holds TWD low for the same duration if it is to be receiving data. When TWIS releases TWCK, it releases TWD at the same time. This can cause a TWI timing violation.
Fix/Workaround
None.

3. **TWALM forced to GND**
The TWALM pin is forced to GND when the alternate function is selected and the TWIS module is enabled.
Fix/Workaround
None.

10.2.11 USBC

1. **UPINRQx.INRQ field is limited to 8-bits**
In Host mode, when using the UPINRQx.INRQ feature together with the multi-packet mode to launch a finite number of packet among multi-packet, the multi-packet size (located in the descriptor table) is limited to the UPINRQx.INRQ value multiply by the pipe size.
Fix/Workaround
UPINRQx.INRQ value shall be less than the number of configured multi-packet.

2. **In USB host mode, downstream resume feature does not work (UHCON.RESUME=1).**
Fix/Workaround
None.

3. **In host mode, the disconnection during OUT transition is not supported**
In USB host mode, a pipe can not work if the previous USB device disconnection has occurred during a USB transfer.
Fix/Workaround
Reset the USBC (USBCON.USB=0 and =1) after a device disconnection (UHINT.DDISCI).

4. **In USB host mode, entering suspend mode can fail**
In USB host mode, entering suspend mode can fail when UHCON.SOFE=0 is done just after a SOF reception (UHINT.HSOFI).
Fix/Workaround
Check that UHNUM.FLENHIGH is below 185 in Full speed and below 21 in Low speed before clearing UHCON.SOFE.

5. **In USB host mode, entering suspend mode for low speed device can fail when the USB freeze (USBCON.FRZCLK=1) is done just after UHCON.SOFE=0.**
Fix/Workaround
When entering suspend mode (UHCON.SOFE is cleared), check that USBFSM.DRDSTATE is not equal to three before freezing the clock (USBCON.FRZCLK=1).