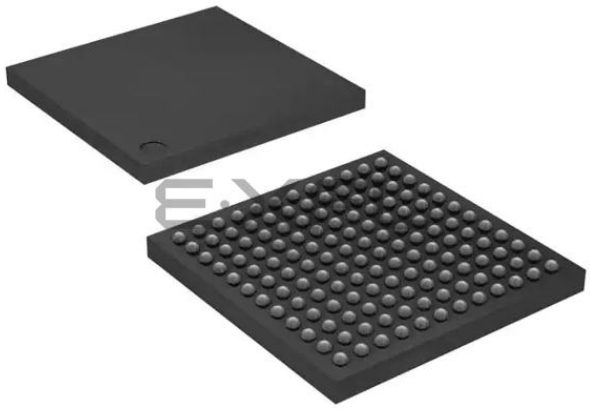


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	32-Bit Single-Core
Speed	66MHz
Connectivity	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB OTG
Peripherals	Brown-out Detect/Reset, DMA, POR, WDT
Number of I/O	110
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.75V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	144-TFBGA
Supplier Device Package	144-FFBGA (11x11)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32uc3a3256-ctut

2. Overview

2.1 Block Diagram

Figure 2-1. Block Diagram

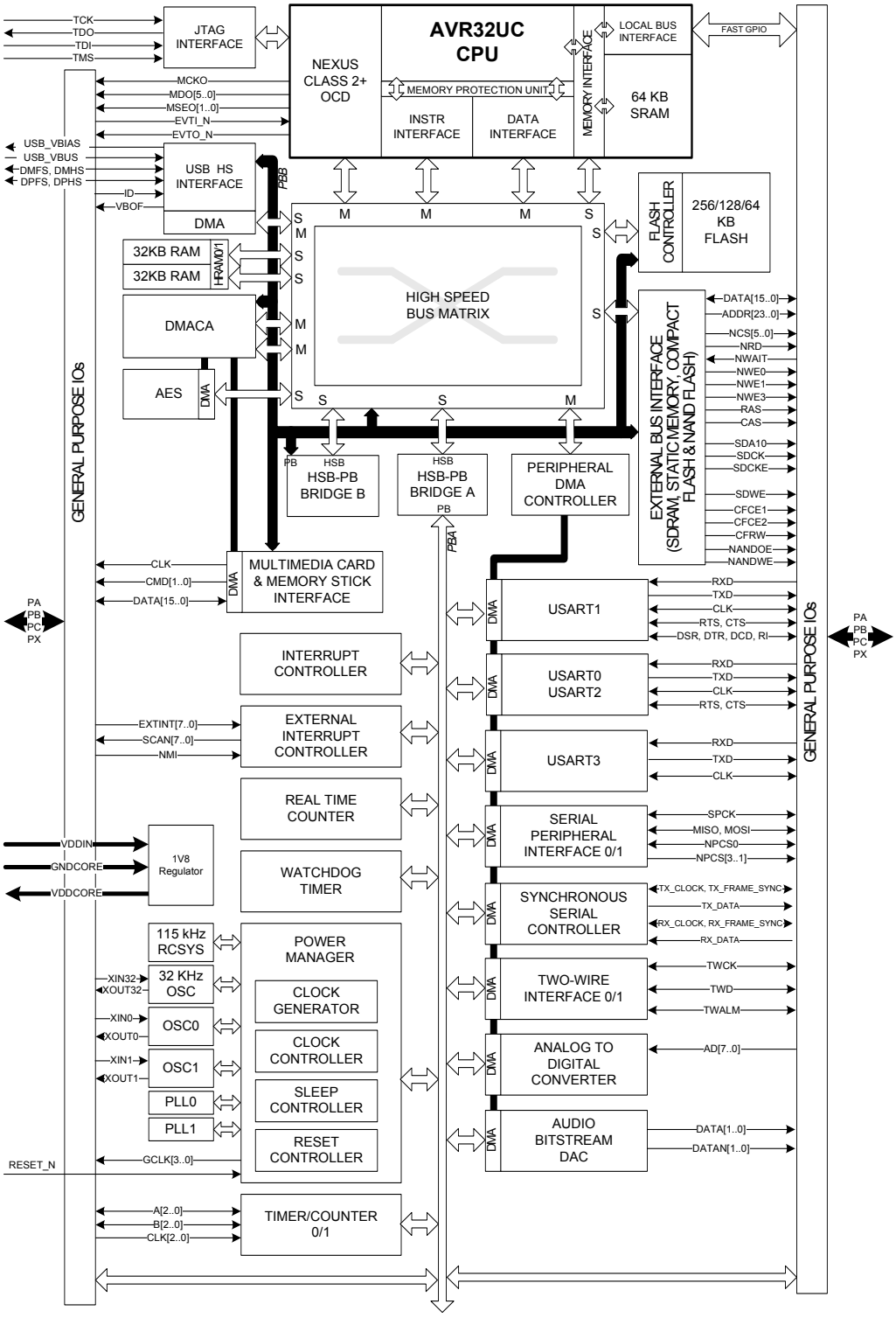
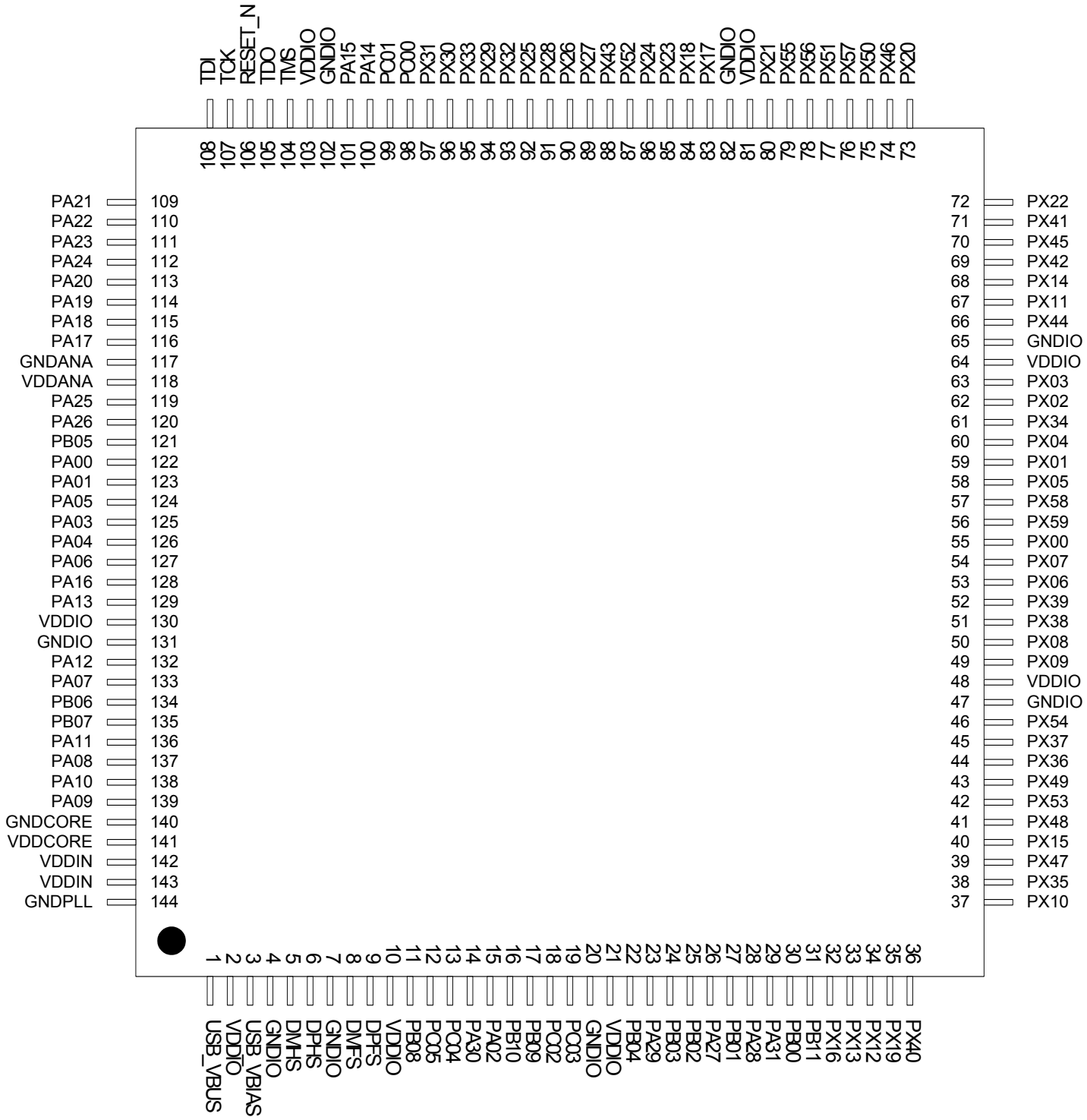


Figure 3-2. LQFP144 Pinout



3.2 Peripheral Multiplexing on I/O lines

3.2.1 Multiplexed Signals

Each GPIO line can be assigned to one of the peripheral functions. The following table describes the peripheral signals multiplexed to the GPIO lines.

Note that GPIO 44 is physically implemented in silicon but it must be kept unused and configured in input mode.

Table 3-1. GPIO Controller Function Multiplexing

BGA 144	QFP 144	BGA 100	PIN	GPIO	Supply	PIN Type [2]	GPIO function			
							A	B	C	D
G11	122	G8[1]	PA00	0	VDDIO	x3	USART0 - RTS	TC0 - CLK1	SPI1 - NPCS[3]	
G12	123	G10[1]	PA01	1	VDDIO	x1	USART0 - CTS	TC0 - A1	USART2 - RTS	
D8	15	E1[1]	PA02	2	VDDIO	x1	USART0 - CLK	TC0 - B1	SPI0 - NPCS[0]	
G10	125	F9	PA03	3	VDDIO	x1	USART0 - RXD	EIC - EXTINT[4]	ABDAC - DATA[0]	
F9	126	E9	PA04	4	VDDIO	x1	USART0 - TXD	EIC - EXTINT[5]	ABDAC - DATAN[0]	
F10	124	G9	PA05	5	VDDIO	x1	USART1 - RXD	TC1 - CLK0	USB - ID	
F8	127	E8[1]	PA06	6	VDDIO	x1	USART1 - TXD	TC1 - CLK1	USB - VBOF	
E10	133	H10[1]	PA07	7	VDDIO	x1	SPI0 - NPCS[3]	ABDAC - DATAN[0]	USART1 - CLK	
C11	137	F8	PA08	8	VDDIO	x3	SPI0 - SPCK	ABDAC - DATA[0]	TC1 - B1	
B12	139	D8	PA09	9	VDDIO	x2	SPI0 - NPCS[0]	EIC - EXTINT[6]	TC1 - A1	
C12	138	C10	PA10	10	VDDIO	x2	SPI0 - MOSI	USB - VBOF	TC1 - B0	
D10	136	C9	PA11	11	VDDIO	x2	SPI0 - MISO	USB - ID	TC1 - A2	
E12	132	G7[1]	PA12	12	VDDIO	x1	USART1 - CTS	SPI0 - NPCS[2]	TC1 - A0	
F11	129	E8[1]	PA13	13	VDDIO	x1	USART1 - RTS	SPI0 - NPCS[1]	EIC - EXTINT[7]	
J6	100	K7[1]	PA14	14	VDDIO	x1	SPI0 - NPCS[1]	TWIMS0 - TWALM	TWIMS1 - TWCK	
J7	101	J7[1]	PA15	15	VDDIO	x1	MCI - CMD[1]	SPI1 - SPCK	TWIMS1 - TWD	
F12	128	E7	PA16	16	VDDIO	x1	MCI - DATA[11]	SPI1 - MOSI	TC1 - CLK2	
H7	116	G10[1]	PA17	17	VDDANA	x1	MCI - DATA[10]	SPI1 - NPCS[1]	ADC - AD[7]	
K8	115	G8[1]	PA18	18	VDDANA	x1	MCI - DATA[9]	SPI1 - NPCS[2]	ADC - AD[6]	
J8	114	H10[1]	PA19	19	VDDANA	x1	MCI - DATA[8]	SPI1 - MISO	ADC - AD[5]	
J9	113	H9[1]	PA20	20	VDDANA	x1	EIC - NMI	SSC - RX_FRAME_SYNC	ADC - AD[4]	
H9	109	K10[1]	PA21	21	VDDANA	x1	ADC - AD[0]	EIC - EXTINT[0]	USB - ID	
H10	110	H6[1]	PA22	22	VDDANA	x1	ADC - AD[1]	EIC - EXTINT[1]	USB - VBOF	
G8	111	G6[1]	PA23	23	VDDANA	x1	ADC - AD[2]	EIC - EXTINT[2]	ABDAC - DATA[1]	
G9	112	J10[1]	PA24	24	VDDANA	x1	ADC - AD[3]	EIC - EXTINT[3]	ABDAC - DATAN[1]	
E9	119	G7[1]	PA25	25	VDDIO	x1	TWIMS0 - TWD	TWIMS1 - TWALM	USART1 - DCD	
D9	120	F7[1])	PA26	26	VDDIO	x1	TWIMS0 - TWCK	USART2 - CTS	USART1 - DSR	
A4	26	A2	PA27	27	VDDIO	x2	MCI - CLK	SSC - RX_DATA	USART3 - RTS	MSI - SCLK
A3	28	A1	PA28	28	VDDIO	x1	MCI - CMD[0]	SSC - RX_CLOCK	USART3 - CTS	MSI - BS
A6	23	B4	PA29	29	VDDIO	x1	MCI - DATA[0]	USART3 - TXD	TC0 - CLK0	MSI - DATA[0]

Table 3-6. Signal Description List

Signal Name	Function	Type	Active Level	Comments
RESET_N	Reset Pin	Input	Low	
DMA Controller - DMACA (optional)				
DMAACK[1:0]	DMA Acknowledge	Output		
DMARQ[1:0]	DMA Requests	Input		
External Interrupt Controller - EIC				
EXTINT[7:0]	External Interrupt Pins	Input		
SCAN[7:0]	Keypad Scan Pins	Output		
NMI	Non-Maskable Interrupt Pin	Input	Low	
General Purpose Input/Output pin - GPIOA, GPIOB, GPIOC, GPIOX				
PA[31:0]	Parallel I/O Controller GPIO port A	I/O		
PB[11:0]	Parallel I/O Controller GPIO port B	I/O		
PC[5:0]	Parallel I/O Controller GPIO port C	I/O		
PX[59:0]	Parallel I/O Controller GPIO port X	I/O		
External Bus Interface - EBI				
ADDR[23:0]	Address Bus	Output		
CAS	Column Signal	Output	Low	
CFCE1	Compact Flash 1 Chip Enable	Output	Low	
CFCE2	Compact Flash 2 Chip Enable	Output	Low	
CFRNW	Compact Flash Read Not Write	Output		
DATA[15:0]	Data Bus	I/O		
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
NCS[5:0]	Chip Select	Output	Low	
NRD	Read Signal	Output	Low	
NWAIT	External Wait Signal	Input	Low	
NWE0	Write Enable 0	Output	Low	
NWE1	Write Enable 1	Output	Low	
RAS	Row Signal	Output	Low	

Table 3-6. Signal Description List

Signal Name	Function	Type	Active Level	Comments
DMHS	USB High Speed Data -	Analog		
DPHS	USB High Speed Data +	Analog		
USB_VBIAS	USB VBIAS reference	Analog		Connect to the ground through a 6810 ohms (+/- 1%) resistor in parallel with a 10pf capacitor. If USB hi-speed feature is not required, leave this pin unconnected to save power
USB_VBUS	USB VBUS signal	Output		
VBOF	USB VBUS on/off bus power control port	Output		
ID	ID Pin fo the USB bus	Input		

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

4.3 The AVR32UC CPU

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

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

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

Figure 4-1 on page 23 displays the contents of AVR32UC.

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

Table 4-1. Instructions with Unaligned Reference Support

Instruction	Supported alignment
ld.d	Word
st.d	Word

4.3.6 Unimplemented Instructions

The following instructions are unimplemented in AVR32UC, and will cause an Unimplemented Instruction Exception if executed:

- All SIMD instructions
- All coprocessor instructions if no coprocessors are present
- retj, incjosp, popjc, pushjc
- tlbr, tlbs, tlbw
- cache

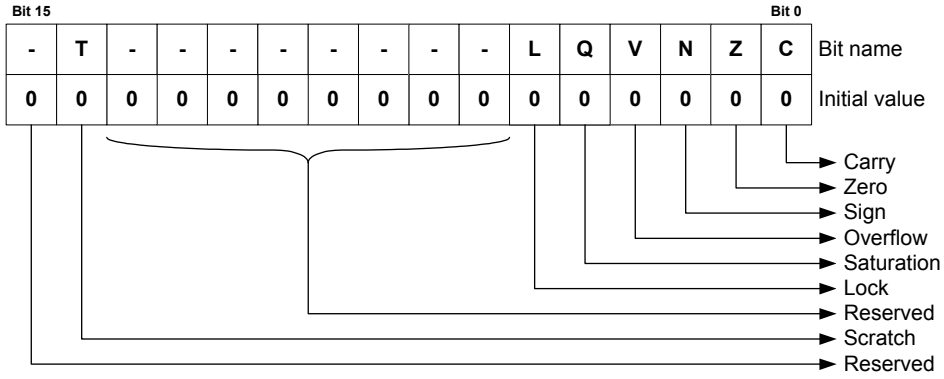
4.3.7 CPU and Architecture Revision

Three major revisions of the AVR32UC CPU currently exist.

The Architecture Revision field in the CONFIG0 system register identifies which architecture revision is implemented in a specific device.

AVR32UC CPU revision 3 is fully backward-compatible with revisions 1 and 2, ie. code compiled for revision 1 or 2 is binary-compatible with revision 3 CPUs.

Figure 4-5. The Status Register Low Halfword



4.4.3 Processor States

4.4.3.1 Normal RISC State

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

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

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

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

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

4.4.3.2 Debug State

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

Table 4-3. System Registers (Continued)

Reg #	Address	Name	Function
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
90	360	MPUPSR2	MPU Privilege Select Register region 2
91	364	MPUPSR3	MPU Privilege Select Register region 3

7.7 System Clock Characteristics

These parameters are given in the following conditions:

- V_{DDCORE} = 1.8V

7.7.1 CPU/HSB Clock Characteristics

Table 7-14. Core Clock Waveform Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$1/(t_{CPCPU})$	CPU Clock Frequency	-40°C < Ambient Temperature < 70°C			84	MHz
$1/(t_{CPCPU})$	CPU Clock Frequency	-40°C < Ambient Temperature < 85°C			66	MHz

7.7.2 PBA Clock Characteristics

Table 7-15. PBA Clock Waveform Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$1/(t_{CPPBA})$	PBA Clock Frequency	-40°C < Ambient Temperature < 70°C			84	MHz
$1/(t_{CPPBA})$	PBA Clock Frequency	-40°C < Ambient Temperature < 85°C			66	MHz

7.7.3 PBB Clock Characteristics

Table 7-16. PBB Clock Waveform Parameters

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$1/(t_{CPPBB})$	PBB Clock Frequency	-40°C < Ambient Temperature < 70°C			84	MHz
$1/(t_{CPPBB})$	PBB Clock Frequency	-40°C < Ambient Temperature < 85°C			66	MHz

7.8.3 Main Oscillators

Table 7-19. Main Oscillators Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
$1/(t_{CPMAIN})$	Oscillator Frequency	External clock on XIN			50	MHz
		Crystal	0.4		20	MHz
C_{L1}, C_{L2}	Internal Load Capacitance ($C_{L1} = C_{L2}$)			7		pF
ESR	Crystal Equivalent Series Resistance				75	Ω
	Duty Cycle		40	50	60	%
t_{ST}	Startup Time	f = 400 KHz f = 8 MHz f = 16 MHz f = 20 MHz		25 4 1.4 1		ms
t_{CH}	XIN Clock High Half-period		0.4 t_{CP}		0.6 t_{CP}	
t_{CL}	XIN Clock Low Half-period		0.4 t_{CP}		0.6 t_{CP}	
C_{IN}	XIN Input Capacitance			7		pF
I_{OSC}	Current Consumption	Active mode at 400 KHz. Gain = G0 Active mode at 8 MHz. Gain = G1 Active mode at 16 MHz. Gain = G2 Active mode at 20 MHz. Gain = G3		30 45 95 205		µA

7.8.4 Phase Lock Loop (PLL0, PLL1)

Table 7-20. PLL Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
F_{OUT}	VCO Output Frequency		80		240	MHz
F_{IN}	Input Frequency (after input divider)		4		16	MHz
I_{PLL}	Current Consumption	Active mode (Fout=80 MHz)		250		µA
		Active mode (Fout=240 MHz)		600		µA

7.8.5 USB Hi-Speed Phase Lock Loop

Table 7-21. PLL Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
F_{OUT}	VCO Output Frequency			480		MHz
F_{IN}	Input Frequency			12		MHz
Delta F_{IN}	Input Frequency Accuracy (applicable to Clock signal on XIN or to Quartz tolerance)		-500		+500	ppm
I_{PLL}	Current Consumption	Active mode @480MHz @1.8V		2.5		mA

7.9 ADC Characteristics

Table 7-22. Channel Conversion Time and ADC Clock

Parameter	Conditions	Min.	Typ.	Max.	Unit
ADC Clock Frequency	10-bit resolution mode			5	MHz
	8-bit resolution mode			8	MHz
Startup Time	Return from Idle Mode			20	µs
Track and Hold Acquisition Time		600			ns
Conversion Time	ADC Clock = 5 MHz			2	µs
	ADC Clock = 8 MHz			1.25	µs
Throughput Rate	ADC Clock = 5 MHz			384 (1)	kSPS
	ADC Clock = 8 MHz			533 (2)	kSPS

1. Corresponds to 13 clock cycles: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.
2. Corresponds to 15 clock cycles: 5 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

Table 7-23. ADC Power Consumption

Parameter	Conditions	Min.	Typ.	Max.	Unit
Current Consumption on VDDANA (1)	On 13 samples with ADC clock = 5 MHz			1.25	mA

1. Including internal reference input current

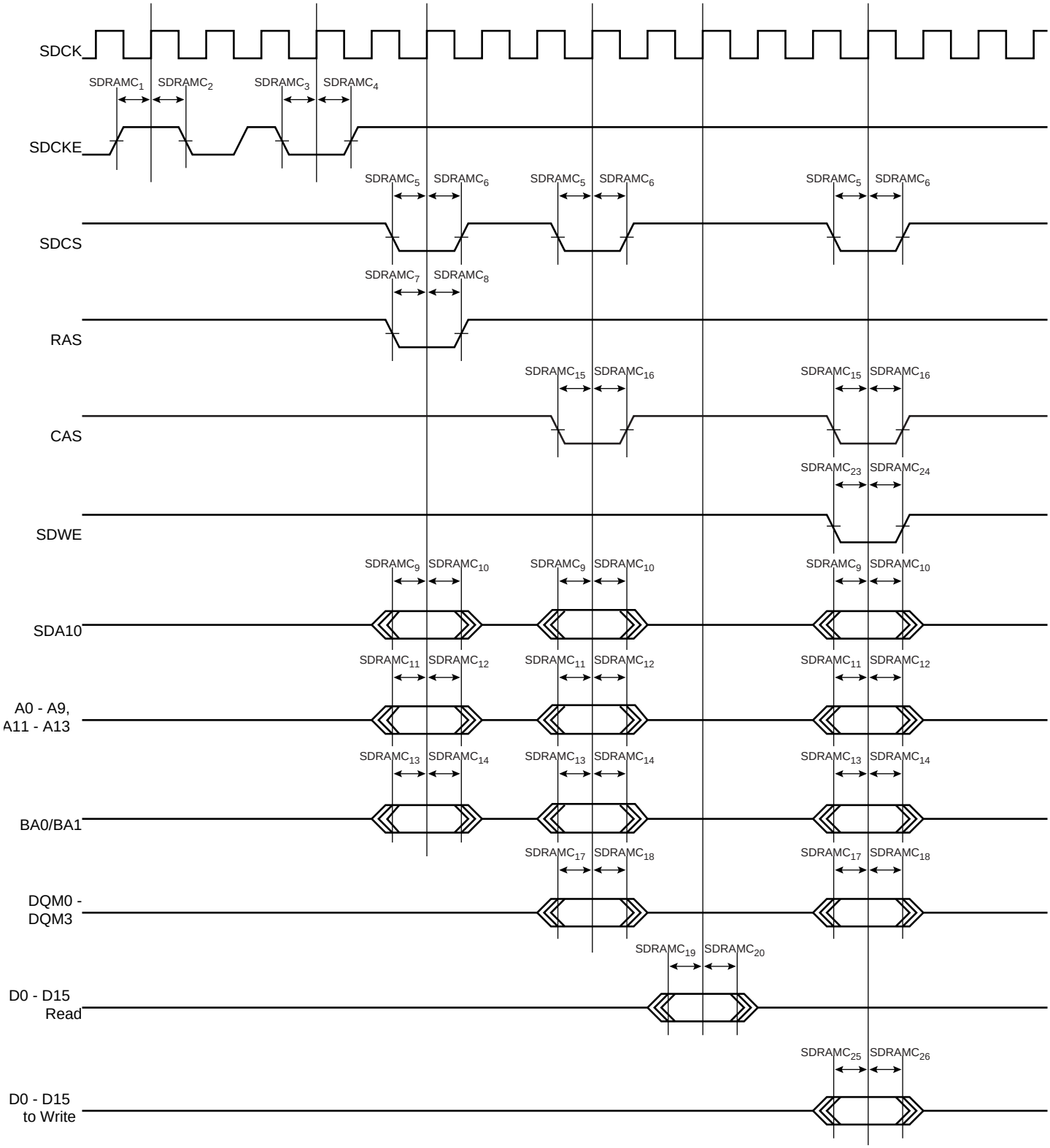
Table 7-24. Analog Inputs

Parameter	Conditions	Min.	Typ.	Max.	Unit
Input Voltage Range		0		VDDANA	V
Input Leakage Current				1	µA
Input Capacitance			7		pF
Input Resistance			350	850	Ohm

Table 7-25. Transfer Characteristics in 8-bit mode

Parameter	Conditions	Min.	Typ.	Max.	Unit
Resolution			8		Bit
Absolute Accuracy	ADC Clock = 5 MHz			0.8	LSB
	ADC Clock = 8 MHz			1.5	LSB
Integral Non-linearity	ADC Clock = 5 MHz		0.35	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.5	LSB
Differential Non-linearity	ADC Clock = 5 MHz		0.3	0.5	LSB
	ADC Clock = 8 MHz		0.5	1.5	LSB
Offset Error	ADC Clock = 5 MHz	-1.5		1.5	LSB
Gain Error	ADC Clock = 5 MHz	-0.5		0.5	LSB

Figure 7-9. SDRAMC Signals relative to SDCK.



7.12 JTAG Characteristics

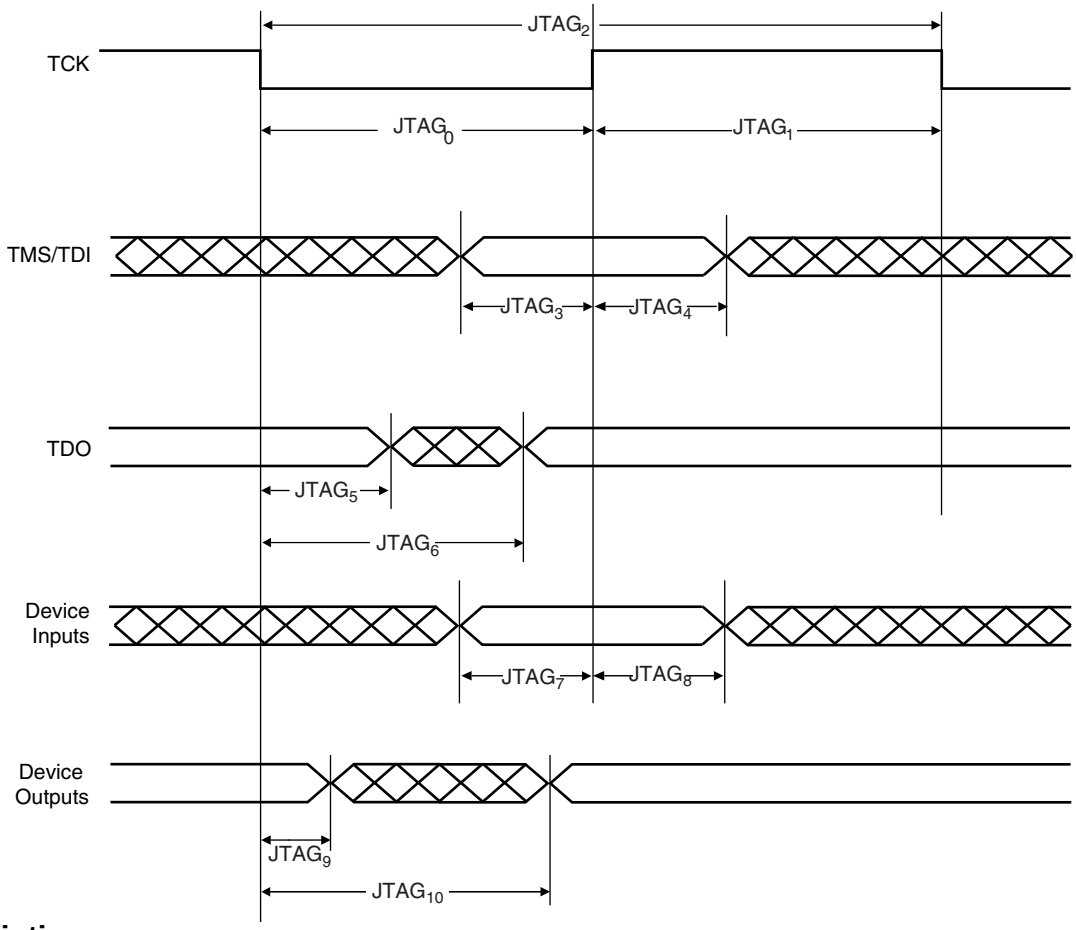
7.12.1 JTAG Interface Signals

Table 7-37. JTAG Interface Timing Specification

Symbol	Parameter	Conditions [1]	Min.	Max.	Unit
$JTAG_0$	TCK Low Half-period		6		ns
$JTAG_1$	TCK High Half-period		3		ns
$JTAG_2$	TCK Period		9		ns
$JTAG_3$	TDI, TMS Setup before TCK High		1		ns
$JTAG_4$	TDI, TMS Hold after TCK High		0		ns
$JTAG_5$	TDO Hold Time		4		ns
$JTAG_6$	TCK Low to TDO Valid			6	ns
$JTAG_7$	Device Inputs Setup Time				ns
$JTAG_8$	Device Inputs Hold Time				ns
$JTAG_9$	Device Outputs Hold Time				ns
$JTAG_{10}$	TCK to Device Outputs Valid				ns

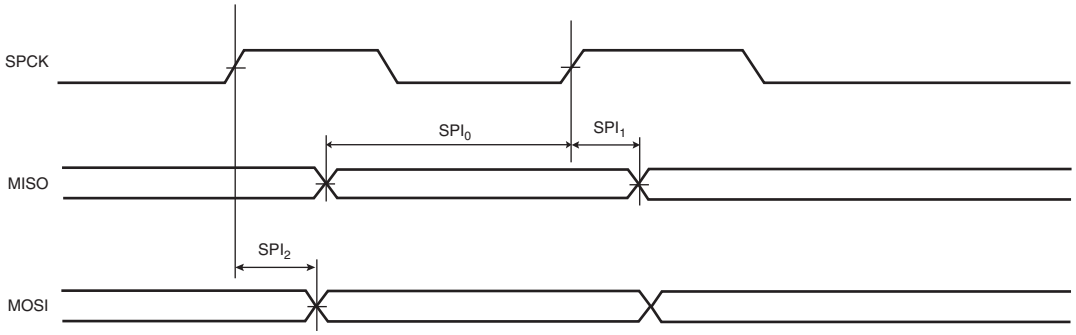
1. V_{VDDIO} from 3.0V to 3.6V, maximum external capacitor = 40pF

Figure 7-10. JTAG Interface Signals



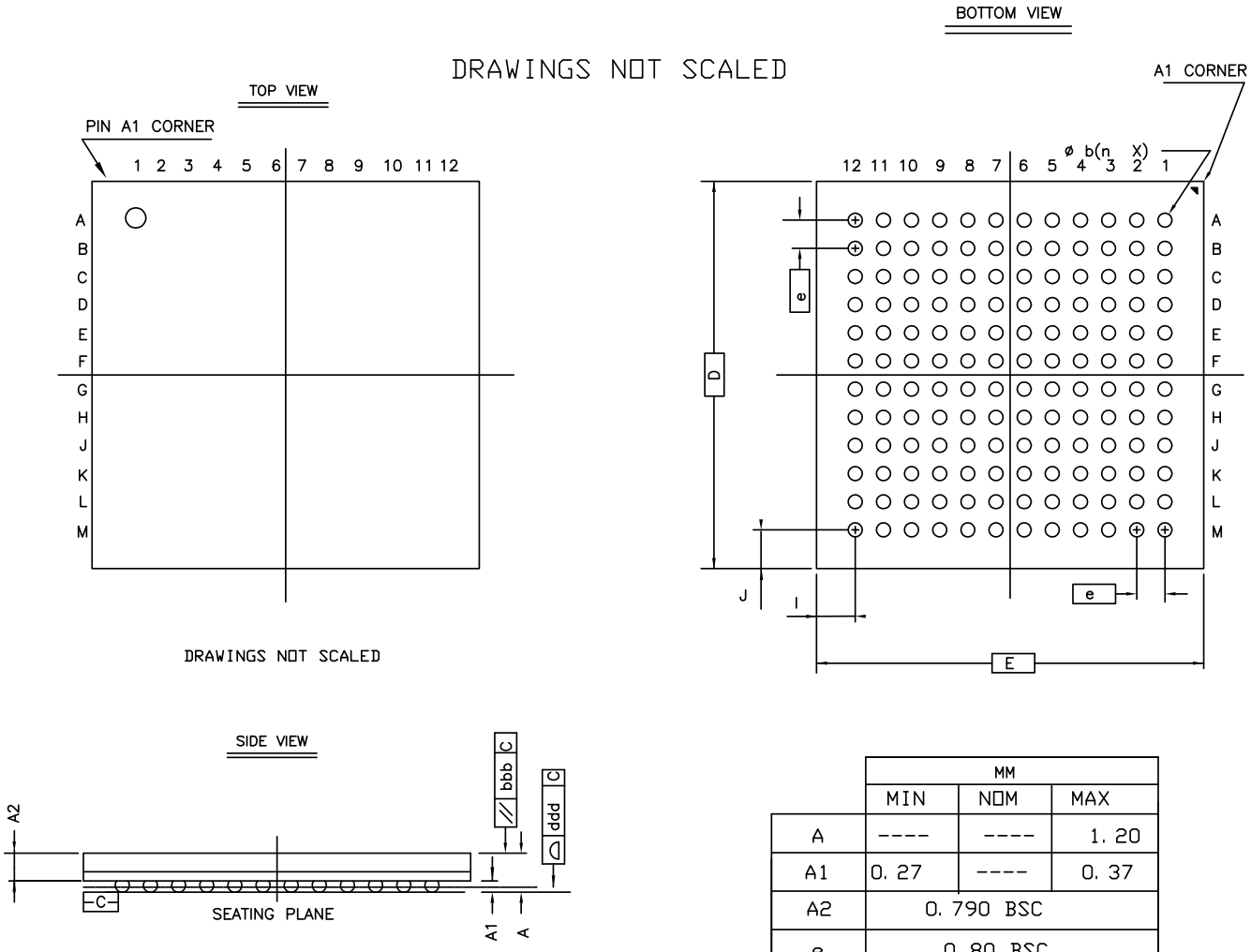
7.13 SPI Characteristics

Figure 7-11. SPI Master mode with (CPOL= NCPHA= 0) or (CPOL= NCPHA= 1)



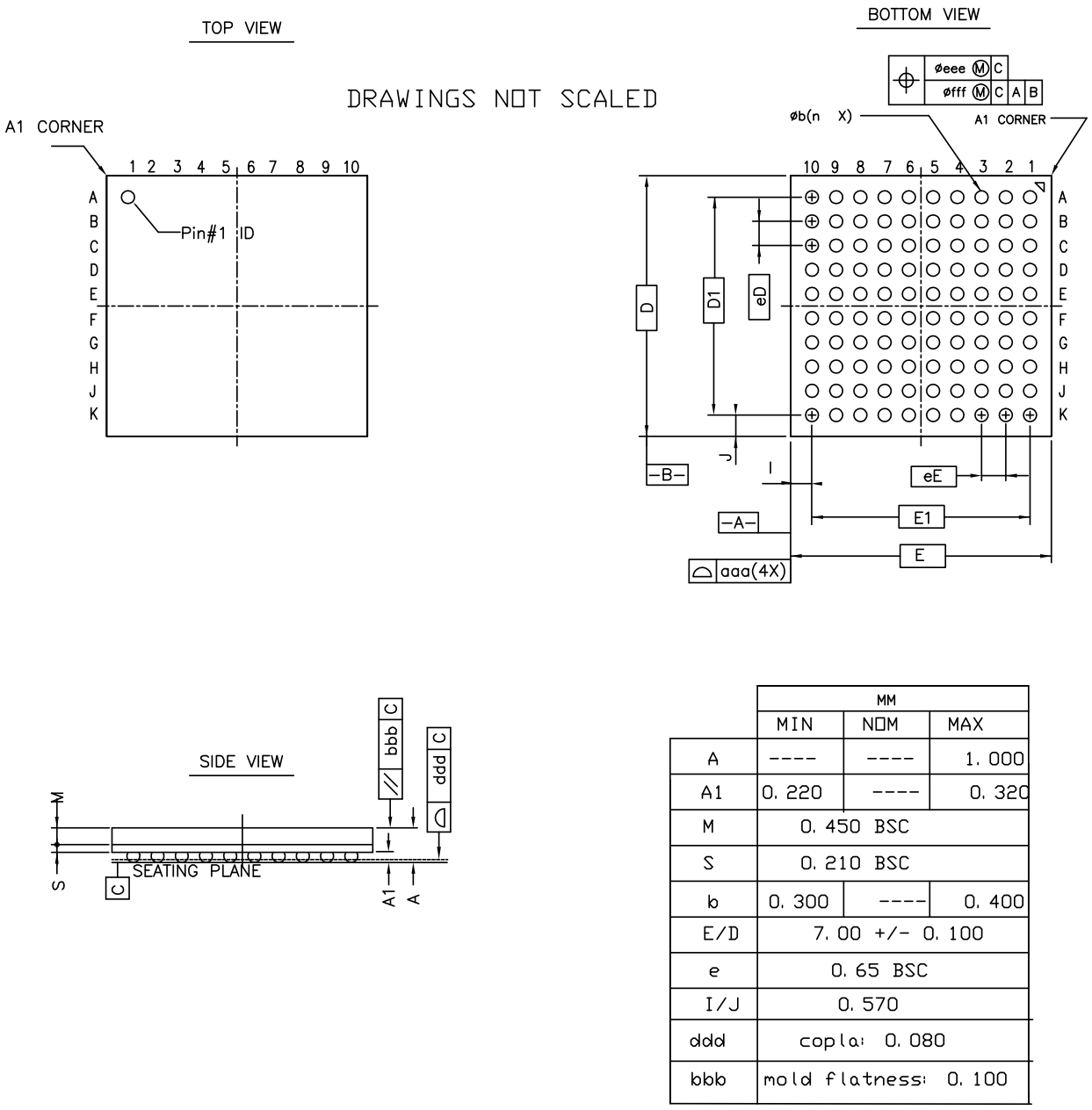
8.2 Package Drawings

Figure 8-1. TFBGA 144 package drawing



	MM		
	MIN	NOM	MAX
A	----	----	1.20
A1	0.27	----	0.37
A2	0.790 BSC		
e	0.80 BSC		
b	0.38	----	0.48
E/D	11.00 BSC		
I/J	1.100 REF		
ddd	coplar 0.120		
bbb	coplar 0.200		

Figure 8-3. VFBGA-100 package drawing



	MM		
	MIN	NOM	MAX
A	----	----	1.000
A1	0.220	----	0.320
M	0.450 BSC		
S	0.210 BSC		
b	0.300	----	0.400
E/D	7.00 +/- 0.100		
e	0.65 BSC		
I/J	0.570		
ddd	copla: 0.080		
bbb	mold flatness: 0.100		

For higher polling time, the software must freeze the pipe for the desired period in order to prevent any "extra" token.

10.1.5 ADC

Sleep Mode activation needs additional A to D conversion
If the ADC sleep mode is activated when the ADC is idle the ADC will not enter sleep mode before after the next AD conversion.
Fix/Workaround
Activate the sleep mode in the mode register and then perform an AD conversion.

10.1.6 USART

ISO7816 info register US_NER cannot be read
The NER register always returns zero.
Fix/Workaround
None.

The LIN ID is not transmitted in mode PDCM='0'
Fix/Workaround
Using USART in mode LIN master with the PDCM bit = '0', the LINID written at the first address of the transmit buffer is not used. The LINID must be written in the LINIR register, after the configuration and start of the PDCA transfer. Writing the LINID in the LINIR register will start the transfer whenever the PDCA transfer is ready.

The LINID interrupt is only available for the header reception and not available for the header transmission
Fix/Workaround
None.

USART LIN mode is not functional with the PDCA if PDCM bit in LINMR register is set to 1
If a PDCA transfer is initiated in USART LIN mode with PDCM bit set to 1, the transfer never starts.
Fix/Workaround
Only use PDCM=0 configuration with the PDCA transfer.

10.1.7 SPI

SPI disable does not work in SLAVE mode
SPI disable does not work in SLAVE mode.
Fix/Workaround
Read the last received data, then perform a software reset by writing a one to the Software Reset bit in the Control Register (CR.SWRST).

SPI bad serial clock generation on 2nd chip_select when SCBR=1, CPOL=1, and NCPHA=0
When multiple chip selects (CS) are in use, if one of the baudrates equal 1 while one (CSRn.SCBR=1) of the others do not equal 1, and CSRn.CPOL=1 and CSRn.NCPHA=0, then an additional pulse will be generated on SCK.
Fix/Workaround
When multiple CS are in use, if one of the baudrates equals 1, the others must also equal 1 if CSRn.CPOL=1 and CSRn.NCPHA=0.

10.1.10 AES

URAD (Unspecified Register Access Detection Status) does not detect read accesses to the write-only KEYW[5..8]R registers
Fix/Workaround
None.

10.1.11 HMATRIX

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.

10.1.12 TWIM

TWIM SR.IDLE goes high immediately when NAK is received
When a NAK is received and there is a non-zero number of bytes to be transmitted, SR.IDLE goes high immediately and does not wait for the STOP condition to be sent. This does not cause any problem just by itself, but can cause a problem if software waits for SR.IDLE to go high and then immediately disables the TWIM by writing a one to CR.MDIS. Disabling the TWIM causes the TWCK and TWD pins to go high immediately, so the STOP condition will not be transmitted correctly.
Fix/Workaround
If possible, do not disable the TWIM. If it is absolutely necessary to disable the TWIM, there must be a software delay of at least two TWCK periods between the detection of SR.IDLE==1 and the disabling of the TWIM.

TWIM TWALM polarity is wrong
The TWALM signal in the TWIM is active high instead of active low.
Fix/Workaround
Use an external inverter to invert the signal going into the TWIM. When using both TWIM and TWIS on the same pins, the TWALM cannot be used.

SMBALERT bit may be set after reset
The SMBus Alert (SMBALERT) bit in the Status Register (SR) might be erroneously set after system reset.
Fix/Workaround
After system reset, clear the SR.SMBALERT bit before commencing any TWI transfer.

10.1.13 TWIS

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.

10.3.5 ADC

Sleep Mode activation needs additional A to D conversion
If the ADC sleep mode is activated when the ADC is idle the ADC will not enter sleep mode before after the next AD conversion.
Fix/Workaround
Activate the sleep mode in the mode register and then perform an AD conversion.

10.3.6 USART

ISO7816 info register US_NER cannot be read
The NER register always returns zero.
Fix/Workaround
None.

The LIN ID is not transmitted in mode PDCM='0'
Fix/Workaround
Using USART in mode LIN master with the PDCM bit = '0', the LINID written at the first address of the transmit buffer is not used. The LINID must be written in the LINIR register, after the configuration and start of the PDCA transfer. Writing the LINID in the LINIR register will start the transfer whenever the PDCA transfer is ready.

The LINID interrupt is only available for the header reception and not available for the header transmission
Fix/Workaround
None.

USART LIN mode is not functional with the PDCA if PDCM bit in LINMR register is set to 1
If a PDCA transfer is initiated in USART LIN mode with PDCM bit set to 1, the transfer never starts.
Fix/Workaround
Only use PDCM=0 configuration with the PDCA transfer.

The RTS output does not function correctly in hardware handshaking mode
The RTS signal is not generated properly when the USART receives data in hardware handshaking mode. When the Peripheral DMA receive buffer becomes full, the RTS output should go high, but it will stay low.
Fix/Workaround
Do not use the hardware handshaking mode of the USART. If it is necessary to drive the RTS output high when the Peripheral DMA receive buffer becomes full, use the normal mode of the USART. Configure the Peripheral DMA Controller to signal an interrupt when 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.

ISO7816 Mode T1: RX impossible after any TX
RX impossible after any TX.
Fix/Workaround
SOFT_RESET on RX+ Config US_MR + Config_US_CR.