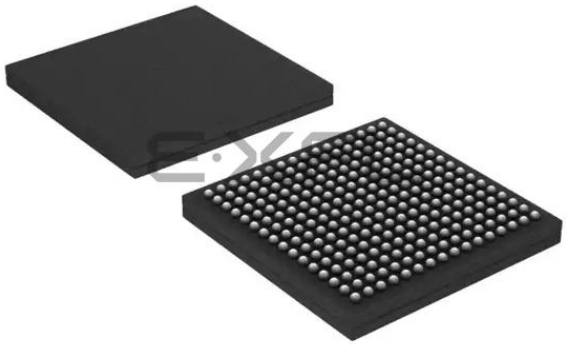


Welcome to E-XFL.COM

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	Obsolete
Core Processor	AVR
Core Size	32-Bit Single-Core
Speed	150MHz
Connectivity	EBI/EMI, Ethernet, I²C, Memory Card, PS/2, SPI, SSC, UART/USART, USB
Peripherals	AC'97, DMA, I²S, LCD, POR, PWM, WDT
Number of I/O	160
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 1.95V
Data Converters	D/A 2x16b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	256-LBGA
Supplier Device Package	256-CTBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at32ap7000-ctur

1. Part Description

The AT32AP7000 is a complete System-on-chip application processor with an AVR32 RISC processor achieving 210 DMIPS running at 150 MHz. AVR32 is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density and high application performance.

AT32AP7000 implements a Memory Management Unit (MMU) and a flexible interrupt controller supporting modern operating systems and real-time operating systems. The processor also includes a rich set of DSP and SIMD instructions, specially designed for multimedia and telecom applications.

AT32AP7000 incorporates SRAM memories on-chip for fast and secure access. For applications requiring additional memory, external 16-bit SRAM is accessible. Additionally, an SDRAM controller provides off-chip volatile memory access as well as controllers for all industry standard off-chip non-volatile memories, like Compact Flash, MultiMedia Card (MMC), Secure Digital (SD)-card, SmartCard, NAND Flash and Atmel DataFlash™.

The Direct Memory Access controller for all the serial peripherals enables data transfer between memories without processor intervention. This reduces the processor overhead when transferring continuous and large data streams between modules in the MCU.

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

AT32AP7000 also features an onboard LCD Controller, supporting single and double scan monochrome and color passive STN LCD modules and single scan active TFT LCD modules. On monochrome STN displays, up to 16 gray shades are supported using a time-based dithering algorithm and Frame Rate Control (FRC) method. This method is also used in color STN displays to generate up to 4096 colors.

The LCD Controller is programmable for supporting resolutions up to 2048 x 2048 with a pixel depth from 1 to 24 bits per pixel.

A pixel co-processor provides color space conversions for images and video, in addition to a wide variety of hardware filter support

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

Synchronous Serial Controllers provide easy access to serial communication protocols, audio standards like I2S and frame-based protocols.

The Java hardware acceleration implementation in AVR32 allows for a very high-speed Java byte-code execution. AVR32 implements Java instructions in hardware, reusing the existing RISC data path, which allows for a near-zero hardware overhead and cost with a very high performance.

The Image Sensor Interface supports cameras with up to 12-bit data buses.

PS2 connectivity is provided for standard input devices like mice and keyboards.

- Configurable coefficients with flexible fixed-point representation.

2.0.3 Debug and Test system

- IEEE1149.1 compliant JTAG and boundary scan
- Direct memory access and programming capabilities through JTAG interface
- Extensive On-Chip Debug features in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 3
- Auxiliary port for high-speed trace information
- Hardware support for 6 Program and 2 data breakpoints
- Unlimited number of software breakpoints supported
- Advanced Program, Data, Ownership, and Watchpoint trace supported

2.0.4 DMA Controller

- 2 HSB Master Interfaces
- 3 Channels
- Software and Hardware Handshaking Interfaces
  - 11 Hardware Handshaking Interfaces
- Memory/Non-Memory Peripherals to Memory/Non-Memory Peripherals Transfer
- Single-block DMA Transfer
- Multi-block DMA Transfer
  - Linked Lists
  - Auto-Reloading
  - Contiguous Blocks
- DMA Controller is Always the Flow Controller
- Additional Features
  - Scatter and Gather Operations
  - Channel Locking
  - Bus Locking
  - FIFO Mode
  - Pseudo Fly-by Operation

2.0.5 Peripheral DMA Controller

- Transfers from/to peripheral to/from any memory space without intervention of the processor.
- Next Pointer Support, forbids strong real-time constraints on buffer management.
- Eighteen channels
  - Two for each USART
  - Two for each Serial Synchronous Controller
  - Two for each Serial Peripheral Interface

2.0.6 Bus system

- HSB bus matrix with 10 Masters and 8 Slaves handled
  - Handles Requests from the CPU Icache, CPU Dcache, HSB bridge, HISI, USB 2.0 Controller, LCD Controller, Ethernet Controller 0, Ethernet Controller 1, DMA Controller 0, DMA Controller 1, and to internal SRAM 0, internal SRAM 1, PB A, PB B, EBI and, USB.

2.1 Package and PinoutAVR32AP7000

Figure 2-2. 256 CTBGA Pinout

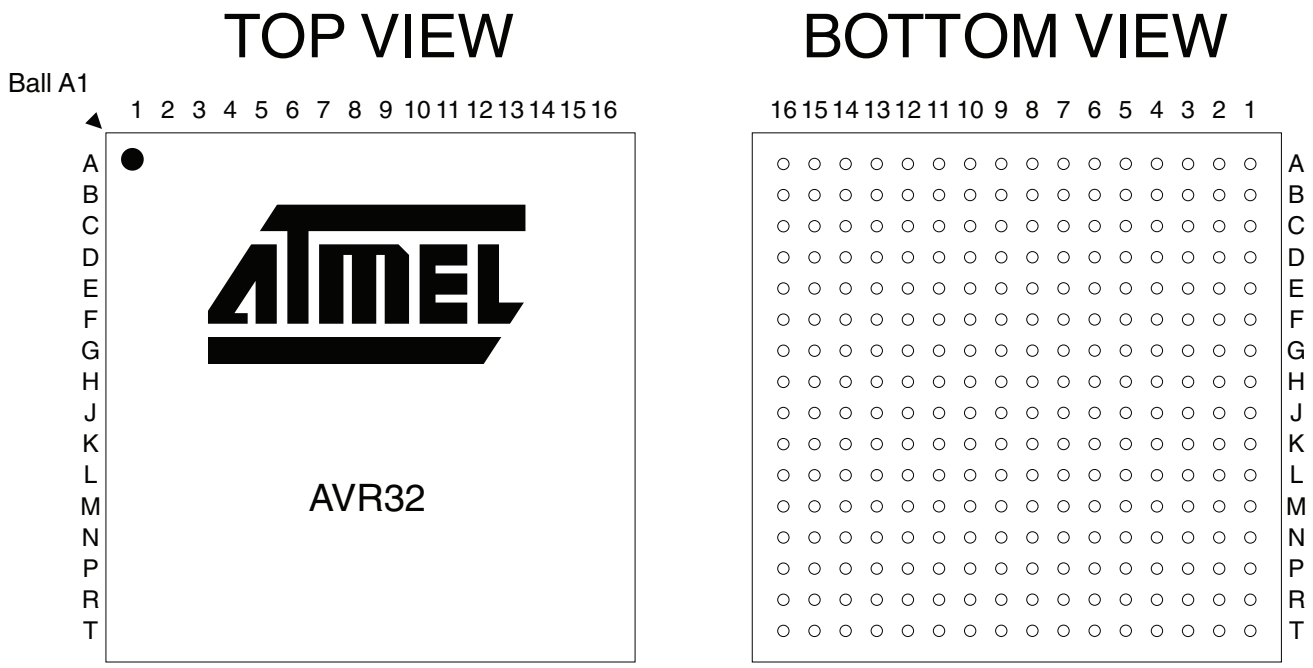


Table 2-1. CTBGA256 Package Pinout A1..T8

	1	2	3	4	5	6	7	8
A	VDDIO	PE15	PE13	PE11	PE07	PE02	AGNDPLL	OSCEN_N
B	GNDIO	PE16	PE12	PE09	PE04	PLL0	AVDDOSC	PC30
C	PD01	PD00	PE14	PE10	PE06	PE00	PLL1	PC31
D	PE17	PE18	PD02	PE08	PE03	GND	AGNDOSC	PC29
E	PX48	PX50	PX49	PX47	PE05	PE01	XOUT32	PC28
F	PX32	PX00	PX33	VDDIO	PX51	AVDDPLL	XIN0	PC27
G	PX04	VDDCORE	PX05	PX03	PX02	PX01	XOUT0	PC26
H	PD06	VDDIO	PD07	PD05	PD04	PD03	GND	XIN32
J	TRST_N	TMS	TDI	TCK	TDO	PD09	PD08	EVTI_N
K	PA05	PA01	PA02	PA00	RESET_N	PA03	PA04	HSDP
L	PA09	PB25	VDDIO	PA08	GND	PB24	AGNDUSB	VDDCORE
M	PA14	PA11	PA13	PA10	PA12	VDDIO	VDDIO	GND
N	PA18	PA16	PA17	PA15	PD14	GND	FSDM	VBG
P	PA20	PA19	PA21	PD11	PD16	XOUT1	GND	PA25
R	PA22	PD10	PA23	PD13	PD17	AVDDUSB	HSDM	PA26
T	VDDIO	GND	PA24	PD12	PD15	XIN1	FSDP	VDDIO

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
TX_CLK	Transmit Clock or Reference Clock	Input		
TX_EN	Transmit Enable	Output		
TX_ER	Transmit Coding Error	Output		
External Bus Interface - EBI				
PX0 - PX53	I/O Controlled by EBI	I/O		
ADDR0 - ADDR25	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		
DATA0 - DATA31	Data Bus	I/O		
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
NCS0 - NCS5	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	
NWE3	Write Enable 3	Output	Low	
RAS	Row Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output		
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output		
SDWE	SDRAM Write Enable	Output	Low	
Image Sensor Interface - ISI				
DATA0 - DATA11	Image Sensor Data	Input		
HSYNC	Horizontal Synchronization	Input		
PCLK	Image Sensor Data Clock	Input		

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
TXD	Transmit Data	Output		
Pulse Width Modulator - PWM				
PWM0 - PWM3	PWM Output Pins	Output		
USB Interface - USBA				
HSDM	High Speed USB Interface Data -	Analog		
FSDM	Full Speed USB Interface Data -	Analog		
HSDP	High Speed USB Interface Data +	Analog		
FSDP	Full Speed USB Interface Data +	Analog		
VBG	USB bandgap	Analog		Connected to a 6810 Ohm ± 0.5% resistor to gound and a 10 pF capacitor to ground.

4. Power Considerations

4.1 Power Supplies

The AT32AP7000 has several types of power supply pins:

- **VDDCORE pins: Power the core, memories, and peripherals. Voltage is 1.8V nominal.**
- **VDDIO pins: Power I/O lines. Voltage is 3.3V nominal.**
- **VDDPLL pin: Powers the PLL. Voltage is 1.8V nominal.**
- **VDDUSB pin: Powers the USB. Voltage is 1.8V nominal.**
- **VDDOSC pin: Powers the oscillators. Voltage is 1.8V nominal.**

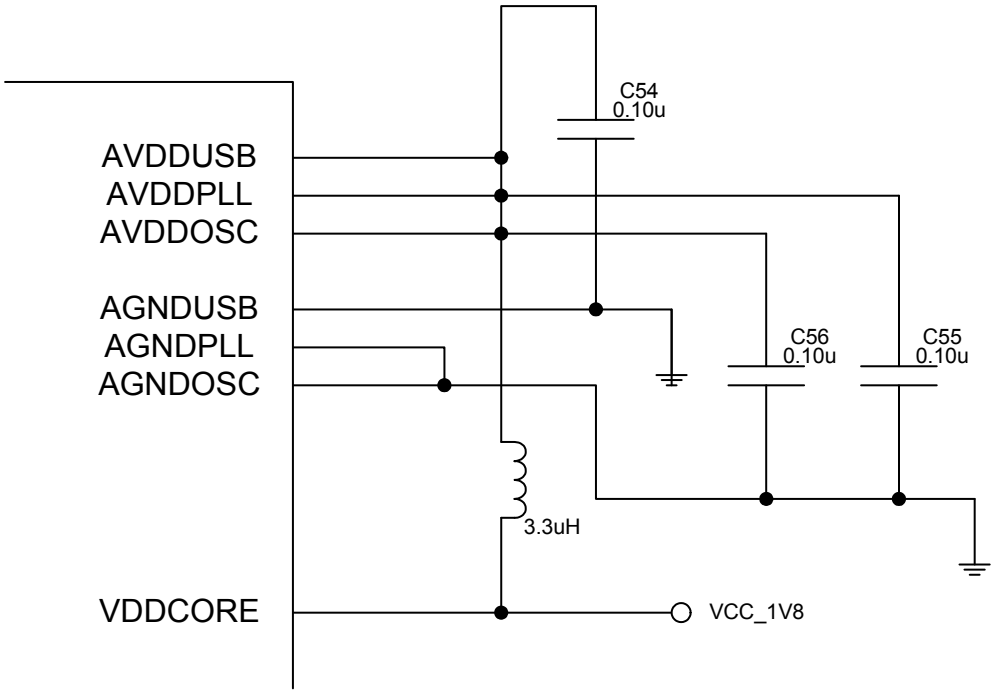
The ground pins GND are common to VDDCORE and VDDIO. The ground pin for VDDPLL is GNDPLL, and the GND pin for VDDOSC is GNDOSC.

See ”Electrical Characteristics” on page 928 for power consumption on the various supply pins.

4.2 Power Supply Connections

Special considerations should be made when connecting the power and ground pins on a PCB. Figure 4-1 shows how this should be done.

Figure 4-1. Connecting analog power supplies



6. Memories

6.1 Embedded Memories

- **32 Kbyte SRAM**
  - **Implemented as two 16Kbyte blocks**
  - **Single cycle access at full bus speed**

6.2 Physical Memory Map

The system bus is implemented as an HSB bus matrix. All system bus addresses are fixed, and they are never remapped in any way, not even in boot. Note that AT32AP7000 by default uses segment translation, as described in the AVR32 Architecture Manual. The 32 bit physical address space is mapped as follows:

Table 6-1. AT32AP7000 Physical Memory Map

Start Address	Size	Device
0x0000_0000	64 Mbyte	EBI SRAM CS0
0x0400_0000	64 Mbyte	EBI SRAM CS4
0x0800_0000	64 Mbyte	EBI SRAM CS2
0x0C00_0000	64 Mbyte	EBI SRAM CS3
0x1000_0000	256 Mbyte	EBI SRAM/SDRAM CS1
0x2000_0000	64 Mbyte	EBI SRAM CS5
0x2400_0000	16 Kbyte	Internal SRAM 0
0x2400_4000	16 Kbyte	Internal SRAM1
0xFF00_0000	4 Kbyte	LCDC configuration
0xFF20_0000	1 KByte	DMACA configuration
0xFF30_0000	1 MByte	USBA Data
0xFFE0_0000	1 MByte	PBA
0xFFF0_0000	1 MByte	PBB

Accesses to unused areas returns an error result to the master requesting such an access.

The bus matrix has the several masters and slaves. Each master has its own bus and its own decoder, thus allowing a different memory mapping per master. The master number in the table below can be used to index the HMATRIX control registers. For example, MCFG2 is associated with the HSB-HSB bridge.

Table 7-1. Peripheral Address Mapping (Continued)

Address		Peripheral Name	Bus
0xFFE03800	PIOE	Parallel Input/Output 2 - PIOE	PB A
0xFFE03C00	PSIF	PS2 Interface - PSIF	PB A
0xFFF00000	PM	Power Manager - PM	PB B
0xFFF00080	RTC	Real Time Counter- RTC	PB B
0xFFF000B0	WDT	WatchDog Timer- WDT	PB B
0xFFF00100	EIC	External Interrupt Controller - EIC	PB B
0xFFF00400	INTC	Interrupt Controller - INTC	PB B
0xFFF00800	HMATRIX	HSB Matrix - HMATRIX	PB B
0xFFF00C00	TC0	Timer/Counter - TC0	PB B
0xFFF01000	TC1	Timer/Counter - TC1	PB B
0xFFF01400	PWM	Pulse Width Modulation Controller - PWM	PB B
0xFFF01800	MACB0	Ethernet MAC - MACB0	PB B
0xFFF01C00	MACB1	Ethernet MAC - MACB1	PB B
0xFFF02000	ABDAC	Audio Bitstream DAC - ABDAC	PB B
0xFFF02400	MCI	MultiMedia Card Interface - MCI	PB B
0xFFF02800	AC97C	AC97 Controller - AC97C	PB B
0xFFF02C00	ISI	Image Sensor Interface - ISI	PB B
0xFFF03000	USBA	USB Configuration Interface - USBA	PB B
0xFFF03400	SMC	Static Memory Controller - SMC	PB B

Table 7-2. Interrupt Request Signal Map

Group	Line	Signal
12	0	SSC2
13	0	PIOA
14	0	PIOB
15	0	PIOC
16	0	PIOD
17	0	PIOE
18	0	PSIF
19	0	EIC0
	1	EIC1
	2	EIC2
	3	EIC3
20	0	PM
21	0	RTC
22	0	TC00
	1	TC01
	2	TC02
23	0	TC10
	1	TC11
	2	TC12
24	0	PWM
25	0	MACB0
26	0	MACB1
27	0	ABDAC
28	0	MCI
29	0	AC97C
30	0	ISI
31	0	USBA
32	0	EBI

7.4 Clock Connections

7.4.1 Timer/Counters

Each Timer/Counter channel can independently select an internal or external clock source for its counter:

Table 7-4. Timer/Counter clock connections

Timer/Counter	Source	Name	Connection
0	Internal	TIMER_CLOCK1	clk_osc32
		TIMER_CLOCK2	clk_pbb / 4
		TIMER_CLOCK3	clk_pbb / 8
		TIMER_CLOCK4	clk_pbb / 16
		TIMER_CLOCK5	clk_pbb / 32
	External	XC0	See Section 7.7
		XC1	
		XC2	
1	Internal	TIMER_CLOCK1	clk_osc32
		TIMER_CLOCK2	clk_pbb / 4
		TIMER_CLOCK3	clk_pbb / 8
		TIMER_CLOCK4	clk_pbb / 16
		TIMER_CLOCK5	clk_pbb / 32
	External	XC0	See Section 7.7
		XC1	
		XC2	

7.4.2 USARTs

Each USART can be connected to an internally divided clock:

Table 7-5. USART clock connections

USART	Source	Name	Connection
0	Internal	CLK_DIV	clk_pba / 8
1			
2			
3			

Table 7-9. PIO Controller A Multiplexing

M9	PA29	PWM - PWM[1]	TC1 - B2
N9	PA30	PM - GCLK[0]	TC1 - CLK1
R9	PA31	PM - GCLK[1]	TC1 - CLK2

7.7.2 PIO Controller B Multiplexing

Table 7-10. PIO Controller B Multiplexing

CTBGA256	I/O Line	Peripheral A	Peripheral B
E12	PB00	ISI - DATA[0]	SPI1 - MISO
E14	PB01	ISI - DATA[1]	SPI1 - MOSI
E16	PB02	ISI - DATA[2]	SPI1 - NPCS[0]
D13	PB03	ISI - DATA[3]	SPI1 - NPCS[1]
D15	PB04	ISI - DATA[4]	SPI1 - NPCS[2]
D14	PB05	ISI - DATA[5]	SPI1 - SCK
D16	PB06	ISI - DATA[6]	MCI - CMD[1]
C15	PB07	ISI - DATA[7]	MCI - DATA[4]
C16	PB08	ISI - HSYNC	MCI - DATA[5]
C14	PB09	ISI - VSYNC	MCI - DATA[6]
B14	PB10	ISI - PCLK	MCI - DATA[7]
A14	PB11	PSIF - CLOCK[1]	ISI - DATA[8]
C13	PB12	PSIF - DATA[1]	ISI - DATA[9]
A13	PB13	SSC2 - TX_DATA	ISI - DATA[10]
B13	PB14	SSC2 - RX_DATA	ISI - DATA[11]
D12	PB15	SSC2 - TX_CLOCK	USART3 - CTS
A12	PB16	SSC2 - TX_FRAME_SYNC	USART3 - RTS
C12	PB17	SSC2 - RX_FRAME_SYNC	USART3 - TXD
B12	PB18	SSC2 - RX_CLOCK	USART3 - RXD
E11	PB19	PM - GCLK[2]	USART3 - CLK
D11	PB20	ABDAC - DATA[1]	AC97C - SDO
A11	PB21	ABDAC - DATA[0]	AC97C - SYNC
C11	PB22	ABDAC - DATAN[1]	AC97C - SCLK
B11	PB23	ABDAC - DATAN[0]	AC97C - SDI
L6	PB24	NMI_N	DMACA - DMARQ[0]
L2	PB25	EXTINT0	DMACA - DMARQ[1]
T9	PB26	EXTINT1	USART2 - RXD
J9	PB27	EXTINT2	USART2 - TXD
M10	PB28	EXTINT3	USART2 - CLK
R13	PB29	PM - GCLK[3]	USART2 - CTS
P13	PB30	PM - GCLK[4]	USART2 - RTS

Table 7-14. IO Pins without multiplexing (Continued)

PX32	EBI - ADDR[16]
PX33	EBI - ADDR[17]
PX34	EBI - ADDR[18]
PX35	EBI - ADDR[19]
PX36	EBI - ADDR[20]
PX37	EBI - ADDR[21]
PX38	EBI - ADDR[22]
PX39	EBI - NCS[0]
PX40	EBI - NCS[1]
PX41	EBI - NCS[3]
PX42	EBI - NRD
PX43	EBI - NWE0
PX44	EBI - NWE1
PX45	EBI - NWE3
PX46	EBI - SDCK
PX47	EBI - SDCKE
PX48	EBI - RAS
PX49	EBI - CAS
PX50	EBI - SDWE
PX51	EBI - SDA10
PX52	EBI - NANDOE
PX53	EBI - NANDWE

- Energy-saving Capabilities
  - Self-refresh, Power-down and Deep Power Modes Supported
  - Supports Mobile SDRAM Devices
- Error Detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by Software
- CAS Latency of 1, 2, 3 Supported
- Auto Precharge Command Not Used

7.8.4 Error Corrected Code Controller

- Hardware Error Corrected Code (ECC) Generation
  - Detection and Correction by Software
- Supports NAND Flash and SmartMedia™ Devices with 8- or 16-bit Data Path.
- Supports NAND Flash/SmartMedia with Page Sizes of 528, 1056, 2112 and 4224 Bytes, Specified by Software

7.8.5 Serial Peripheral Interface

- Supports communication with serial external devices
  - Four chip selects with external decoder support allow communication with up to 15 peripherals
  - Serial memories, such as DataFlash™ and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External co-processors
- Master or slave serial peripheral bus interface
  - 8- to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delays between consecutive transfers and between clock and data per chip select
  - Programmable delay between consecutive transfers
  - Selectable mode fault detection
- Very fast transfers supported
  - Transfers with baud rates up to MCK
  - The chip select line may be left active to speed up transfers on the same device

7.8.6 Two-wire Interface

- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations

8. Boot Sequence

This chapter summarizes the boot sequence of the AT32AP7000. The behaviour after power-up is controlled by the Power Manager.

8.1 Starting of clocks

After power-up, the device will be held in a reset state by the Power-On Reset (POR) circuitry until the voltage has reached the power-on reset rising threshold value (see Electrical Characteristics for details). This ensures that all critical parts of the device are properly reset.

Once the power-on reset is complete, the device will use the XIN0 pin as clock source. XIN0 can be connected either to an external clock, or a crystal. The OSCEN_N pin is connected either to VDD or GND to inform the Power Manager on how the XIN0 pin is connected. If XIN0 receives a signal from a crystal, dedicated circuitry in the Power Manager keeps the part in a reset state until the oscillator connected to XIN0 has settled. If XIN0 receives an external clock, no such settling delay is applied.

On system start-up, the PLLs are disabled. All clocks to all modules are running. No clocks have a divided frequency, all parts of the system recieves a clock with the same frequency as the XIN0 clock.

Note that the power-on reset will release reset at a lower voltage threshold than the minimum specified operating voltage. If the voltage is not guaranteed to be stable by the time the device starts executing, an external brown-out reset circuit should be used.

8.2 Fetching of initial instructions

After reset has been released, the AVR32AP CPU starts fetching instructions from the reset address, which is 0xA000_0000. This address lies in the P2 segment, which is non-translated, non-cacheable, and permanently mapped to the physical address range 0x0000_0000 to 0x2000_0000. This means that the instruction being fetched from virtual address 0xA000_0000 is being fetched from physical address 0x0000_0000. Physical address 0x0000_0000 is mapped to EBI SRAM CS0. This is the external memory the device boots from.

The code read from the SRAM CS0 memory is free to configure the system to use for example the PLLs, to divide the frequency of the clock routed to some of the peripherals, and to gate the clocks to unused peripherals.

9. Ordering Information

Table 9-1. Ordering Information

Ordering Code	Package	Package Type	Packing	Temperature Operating Range
AT32AP7000-CTUR	CTBGA256	Green	Reel	Industrial (-40°C to 85°C)
AT32AP7000-CTUT	CTBGA256	Green	Tray	Industrial (-40°C to 85°C)

Fix/Workaround

Before executing any code the user should enable the RTC with the smallest prescaler and poll that the RTC is counting before doing anything in your program. Another way to ensure that the osc32 is valid is to use interrupts with TOP=1.

Example:

```
//reset the counter register
  AVR32_RTC.val = 0x0;
//enable the RTC with the smallest prescaler
  AVR32_RTC.ctrl = 0x1;
//wait until the value increases
  while(AVR32_RTC.val == 0);
```

26. SPI can generate a false RXREADY signal in SLAVE mode

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

Fix/Workaround

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

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

27. EBI address lines 23, 24, and 25 are pulled up when booting up

After reset the EBI address lines 23, 24 and 25 are tristated with pullups. Booting from a flash larger than 8 MB using these lines will fail, as the flash will be accessed with these address bits set.

Fix/Workaround

Add external pulldown resistors (5 kΩ) on these lines if booting from a flash larger than 8 MB using these address lines.

28. SSC - Additional delay on TD output

A delay from 2 to 3 system clock cycles is added to TD output when:

TCMR.START = Receive Start,

TCMR.STTDLY = more than ZERO,

RCMR.START = Start on falling edge / Start on Rising edge / Start on any edge

RFMR.FSOS = None (input)

Fix/Workaround

None.

29. SSC - TF output is not correct

TF output is not correct (at least emitted one serial clock cycle later than expected) when:

TFMR.FSOS = Driven Low during data transfer/ Driven High during data transfer

MCI data write operation with less than 12 bytes is impossible. The Data Write operation with a number of bytes less than 12 leaves the internal MCI FIFO in an inconsistent state. Subsequent reads and writes will not function properly.

Fix/Workaround

Always transfer 12 or more bytes at a time. If less than 12 bytes are transferred, the only recovery mechanism is to perform a software reset of the MCI.

5. **MMC SDIO interrupt only works for slot A**

   If 1-bit data bus width and on other slots than slot A, the SDIO interrupt can not be captured.

   Fix/Workaround

   Use slot A.

6. **PSIF TXEN/RXEN may disable the transmitter/receiver**

   Writing a '0' to RXEN will disable the receiver. Writing '0' to TXEN will disable the transmitter.

   Fix/Workaround

   When accessing the PS/2 Control Register always write '1' to RXEN to keep the receiver enabled, and write '1' to TXEN to keep the transmitter enabled.

7. **PSIF TXRDY interrupt corrupts transfers**

   When writing to the Transmit Holding Register (THR), the data will be transferred to the data shift register immediately, regardless of the state of the data shift register. If a transfer is ongoing, it will be interrupted and a new transfer will be started with the new data written to THR.

   Fix/Workaround

   Use the TXEMPTY-interrupt instead of the TXRDY-interrupt to update the THR. This ensures that a transfer is completed.

8. **PSIF Status Register bits return 0**

   The PARITY, NACK and OVRUN bits in the PSIF Status Register cannot be read. Reading these bits will always return zero.

   Fix/Workaround

   None

9. **PSIF Transmit does not work as intended**

   While PSIF receiving works, transmitting using the PSIF does not work.

   Fix/Workaround

   Do not transmit using the PSIF.

10. **LCD memory error interupt does not work**

    Writing to the MERIT-bit in the LCD Interrupt Test Register (ITR) does not cause an interrupt as intended. The MERIC-bit in the LCD Interrupt Clear Register (ICR) cannot be written. This means that if the MERIS-bit in ISR is set, it cannot be cleared.

    Fix/Workaround

    Memory error interrupt should not be used.

PDC/PDCA transfers: None.

Manual transfers (no PDC and TX slave only): Read the RHR every time the THR is written. The OVRS flag of the status register will track any UNDERRUN on the TX side.

30. HMATRIX - Fixed priority arbitration does not work

Fixed priority arbitration does not work.

Fix/Workaround

Use Round-robin arbitration instead.

31. OSC32 is not available for RTC, WDT, TIMERs and USARTs at startup

Right after startup the osc32 clock to internal modules is not valid. The osc32 clock will be valid for use approximately 128 osc32 cycles after the the first instruction is executed. This has consequences if you are planning to use the RTC, WDT, going into sleep mode and USARTs with SCK and TCs with TIMER_CLOCK0.

Fix/Workaround

Before executing any code the user should enable the RTC with the smallest prescaler and poll that the RTC is counting before doing anything in your program. Another way to ensure that the osc32 is valid is to use interrupts with TOP=1.

Example:

```
//reset the counter register
  AVR32_RTC.val = 0x0;
//enable the RTC with the smallest prescaler
  AVR32_RTC.ctrl = 0x1;
//wait until the value increases
  while(AVR32_RTC.val == 0);
```

32. SPI can generate a false RXREADY signal in SLAVE mode

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

Fix/Workaround

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

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

33. EBI address lines 23, 24, and 25 are pulled up when booting up

After reset the EBI address lines 23, 24 and 25 are tristated with pullups. Booting from a flash larger than 8 MB using these lines will fail, as the flash will be accessed with these address bits set.

Fix/Workaround

Add external pulldown resistors (5 kΩ) on these lines if booting from a flash larger than 8 MB using these address lines.

34. SSC - Additional delay on TD output

A delay from 2 to 3 system clock cycles is added to TD output when:

TCMR.START = Receive Start,

TCMR.STTDLY = more than ZERO,

RCMR.START = Start on falling edge / Start on Rising edge / Start on any edge

RFMR.FSOS = None (input)

Fix/Workaround

None.

35. SSC - TF output is not correct

TF output is not correct (at least emitted one serial clock cycle later than expected) when:

TFMR.FSOS = Driven Low during data transfer/ Driven High during data transfer

TCMR.START = Receive start

RFMR.FSOS = None (Input)

RCMR.START = any on RF (edge/level)

Fix/Workaround

None.

36. USART - TXD signal is floating in Modem and Hardware Handshaking mode

The TXD signal is floating in Modem and Hardware Handshaking mode, but should be pulled up.

Fix/Workaround

Enable pullup on this line in the PIO.

37. PWM - Impossible to update a period equal to 0 by using the CUPD register

It is impossible to UPDATE a period equal to 0 by the using of the UPDATE register (CUPD).

Fix/Workaround

To update a period equal to 0, write directly to the CPRD register.

38. WDT Clear is blocked after WDT Reset

A watchdog timer event will, after reset, block writes to the WDT_CLEAR register, preventing the program to clear the next Watchdog Timer Reset.

Fix/Workaround

If the RTC is not used a write to AVR32_RTC.ctrl.pclr = 1, instead of writing to AVR32_WDT.clr, will reset the prescaler and thus prevent the watchdog event from happening. This will render the RTC useless, but prevents WDT reset because the RTC and WDT share the same prescaler. Another sideeffect of this is that the watchdog timeout period will be half the expected timeout period.

If the RTC is used one can disable the Watchdog Timer (WDT) after a WDT reset has occured. This will prevent the WDT resetting the system. To make the WDT functional again a hard reset (power on reset or RESET_N) must be applied. If you still want to use the WDT after a WDT reset a small code can be inserted at the startup checking the AVR32_PM.rcause register for WDT reset and use a GPIO pin to reset the system. This method requires that one of the GPIO pins are available and connected externally to the

Table of Contents