Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Hub/Microcontroller
Core Processor	AVR
Program Memory Type	-
Controller Series	AT43USB
RAM Size	512 x 8
Interface	SPI Serial, USB, UART
Number of I/O	32
Voltage - Supply	4.4V ~ 5.25V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at43usb320a-ac

Architectural Overview

The peripherals and features of the AT43USB320A microcontroller are similar to those of the AT90S8515, with the exception of the following modifications:

- External Program Memory
- No EEPROM
- No external data memory accesses
- No Analog Comparation
- Idle mode not supported
- USB Hub with attached function
- No internal pull-ups in the general-purpose I/O pin PA, PB, PC, PD

The embedded USB hardware of the AT43USB320A is a compound device, consisting of a 5 port hub with a permanently attached function on one port. The hub and attached function are two independent USB devices, each having its own device addresses and control endpoints. The hub has its dedicated interrupt endpoint, while the USB function has 2 additional programmable endpoints with separate 8-byte FIFOs.

The microcontroller always runs from a 12 MHz clock that is generated by the USB hardware. While the nominal and average period of this clock is 83.3 ns, it may have single cycles that deviate by ±20.8 ns during a phase adjustment by the SIE's clock/data separator of the USB hardware.

The microcontroller shares most of the control and status registers of the megaAVR™ Microcontroller Family. The registers for managing the USB operations are mapped into its SRAM space. The I/O section on page 16 summarizes the available I/O registers. The “AVR Register Set” on page 36 covers the AVR registers. Please refer to the Atmel AVR manual for more information.

The fast-access register file concept contains 32 x 8-bit general-purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one Arithmetic Logic Unit (ALU) operation is executed. Two operands are output from the register file, the operation is executed, and the result is stored back in the register file – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing - enabling efficient address calculations. One of the three address pointers is also used as the address pointer for look-up tables in program memory. These added function registers are the 16-bit X-, Y- and Z-registers.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 2 on page 6 shows the AT43USB320A AVR Enhanced RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the register file as well. This is enabled by the fact that the register file is assigned the 32 lowest Data Space addresses ($00 - $1 F), allowing them to be accessed as though they were ordinary memory locations.

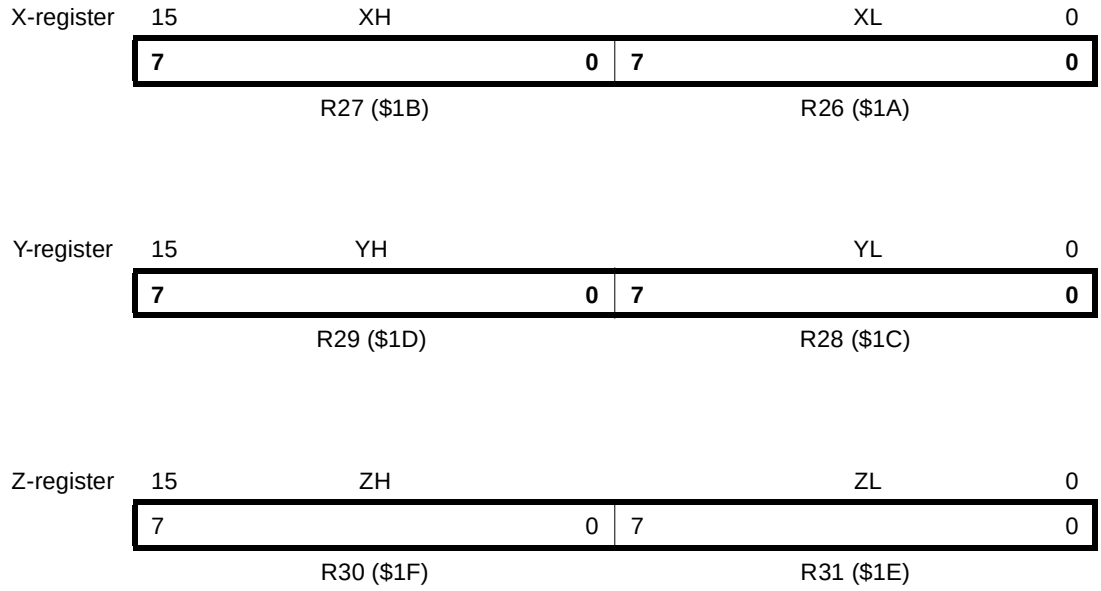
The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the register file, $20 - $5F.

The AVR uses a Harvard architecture concept – with separate memories and buses for program and data. The program memory is executed with a single-level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is a downloadable SRAM or a mask programmed ROM.

With the relative jump and call instructions, the whole 24K address space is directly accessed. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

X-, Y- and Z-Registers

Registers R26..R31 contain some added functions to their general-purpose usage. These registers are address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as:



In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories – arithmetic, logical and bit-functions.

Program Memory

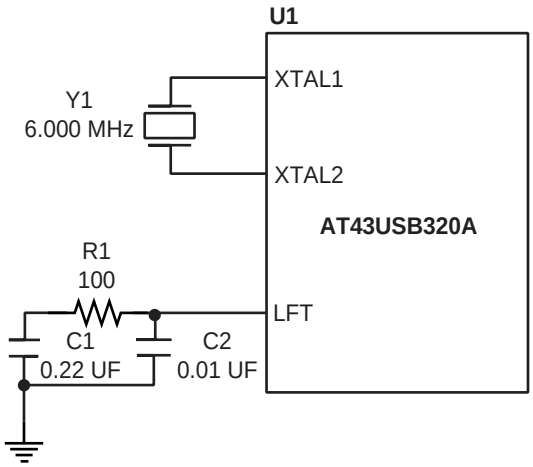
The AT43USB320A operates from an external program memory. Since all instructions are 16- or 32-bit words, the program memory is organized as X16. The AT43USB320A Program Counter (PC) is 16 bits wide, thus addressing the 64K program memory addresses.

Constant tables can be allocated within the entire program memory address space (see the LPM - Load Program Memory instruction description).

Table 4. USB Hub and Function Registers

Name	Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GLB_STATE	$1FFB	–			SUSP FLG	RESUME FLG	RMWUPE	CONFG	HADD EN
SPRSR	$1FFA	–	–	–	–	–	FRWUP	RSM	GLB SUSP
SPRSIE	$1FF9	–	–	–	–	–	FRWUP IE	RSM IE	GLB SUSP IE
UISR	$1FF7	SOF INT	EOF2 INT	–	FEP3 INT	HEP0 INT	FEP2 INT	FEP1 INT	FEP0 INT
UIAR	$1FF5	SOF INTACK	EOF2 INTACK	–	FEP3 INTACK	HEP0 INTACK	FEP2 INTACK	FEP1 INTACK	FEP0 INTACK
UIER	$1FF3	SOF IE	EOF2 IE	–	FEP3 IE	HEP0 IE	FEP2 IE	FEP1 IE	FEP0 IE
HADDR	$1FEF	SAEN	HADD6	HADD5	HADD4	HADD3	HADD2	HADD1	HADD0
FADDR	$1FEE	FEN	FADD6	FADD5	FADD4	FADD3	FADD2	FADD1	FADD0
HENDP0_CNTR	$1FE7	EPEN	–	–	–	DTGLE	EPDIR	EPTYPE1	EPTYPE0
FENDP0_CNTR	$1FE5	EPEN	–	–	–	DTGLE	EPDIR	EPTYPE1	EPTYPE0
FENDP1_CNTR	$1FE4	EPEN	–	–	–	DTGLE	EPDIR	EPTYPE1	EPTYPE0
FENDP2_CNTR	$1FE3	EPEN	–	–	–	DTGLE	EPDIR	EPTYPE1	EPTYPE0
HCSR0	$1FDF	–	–	–	–	STALL SENT	RX SETUP	RX OUT PACKET	TX CEMPLETE
FCSR0	$1FDD	–	–	–	–	STALL SENT	RX SETUP	RX OUT PACKET	TX COMPLETE
FCSR1	$1FDC	–	–	–	–	STALL SENT	RX SETUP	RX OUT PACKET	TX COMPLETE
FCSR2	$1FDB	–	–	–	–	STALL SENT	RX SETUP	RX OUT PACKET	TX COMPLETE
HDR0	$1FD7	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
FDR0	$1FD5	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
FDR1	$1FD4	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
FDR2	$1FD3	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
HBYTE_CNT0	$1FCF	–	–	–	BYTCT4	BYTCT3	BYTCT2	BYTCT1	BYTCT0
FBYTE_CNT0	$1FCD	–	–	–	BYTCT4	BYTCT3	BYTCT2	BYTCT1	BYTCT0
FBYTE_CNT1	$1FCC	–	–	–	BYTCT4	BYTCT3	BYTCT2	BYTCT1	BYTCT0
FBYTE_CNT2	$1FCB	–	–	–	BYTCT4	BYTCT3	BYTCT2	BYTCT1	BYTCT0
HSTR	$1FC7	–	–	–	–	OVLSC	LPSC	OVI	LPS
HPCON	$1FC5	–	HPCON2	HPCON1	HPCON0	–	HPADD2	HPADD1	HPADD0
HPSTAT5	$1FBC	–	LSP	PPSTAT	PRSTAT	POCI	PSSTAT	PESTAT	PCSTAT
HPSTAT4	$1FBB	–	LSP	PPSTAT	PRSTAT	POCI	PSSTAT	PESTAT	PCSTAT
HPSTAT3	$1FBA	–	LSP	PPSTAT	PRSTAT	POCI	PSSTAT	PESTAT	PCSTAT
HPSTAT2	$1FB9	–	LSP	PPSTAT	PRSTAT	POCI	PSSTAT	PESTAT	PCSTAT
HPSTAT1	$1FB8	–	LSP	PPSTAT	PRSTAT	POCI	PSSTAT	PESTAT	PCSTAT
HPSCR5	$1FB4	–	–	–	RSTSC	POCIC	PSSC	PESC	PCSC
HPSCR4	$1FB3	–	–	–	RSTSC	POCIC	PSSC	PESC	PCSC
HPSCR3	$1FB2	–	–	–	RSTSC	POCIC	PSSC	PESC	PCSC
HPSCR2	$1FB1	–	–	–	RSTSC	POCIC	PSSC	PESC	PCSC
HPSCR1	$1FB0	–	–	–	RSTSC	POCIC	PSSC	PESC	PCSC
PSTATE5	$1FAC	–	–	–	–	–	–	DPSTATE	DMSTATE
PSTATE4	$1FAB	–	–	–	–	–	–	DPSTATE	DMSTATE
PSTATE3	$1FAA	–	–	–	–	–	–	DPSTATE	DMSTATE
PSTATE2	$1FA9	–	–	–	–	–	–	DPSTATE	DMSTATE
PSTATE1	$1FA8	–	–	–	–	–	–	DPSTATE	DMSTATE
HCAR0	$1FA7	CTL DIR	DATA END	FORCE STALL	TX PACKET READY	STALL_SENT-ACK	RX_SETUP_ACK	RX_OUT_PACKET_ACK	TX_COMPLETE-ACK

Figure 4. Oscillator and PLL



Reset and Interrupt Handling

The AT43USB320A provides 22 different interrupt sources with 13 separate reset vectors, each with a separate program vector in the program memory space. Eleven of the interrupt sources share 2 interrupt reset vectors. These 11 are the USB related interrupts. All interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 6. The list also determines the priority levels of the different interrupts. The lower the address, the higher is the priority level. RESET has the highest priority, and next is INT0 – the USB Suspend and Resume Interrupt, etc.

Table 6. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Reset, Power-on Reset and Watchdog Reset
2	$002	INT0	USB Suspend and Resume
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$008	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$00A	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$00C	TIMER1, OVF	Timer/Counter1 Overflow
8	$00E	TIMER0, OVF	Timer/Counter0 Overflow
9	$010	SPI, STC	SPI Serial Transfer Complete
10	$012	UART RX	UART RX Complete
11	$014	UART UDRE	UART RX Data Receiver Output
12	$016	UART TX	UART TX Complete
13	$018	USB HW	USB Hardware

General Interrupt Mask Register – GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	**INT1**	**INT0**	–	–	–	–	–	–	**GIMSK**
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from program memory address $004. See also “External Interrupts” on page 29.

- **Bit 6 – INT0: Interrupt Request 0 (Suspend/Resume Interrupt) Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of Interrupt Request 0 is executed from program memory address $002. See also “External Interrupts” on page 29.

- **Bits 5..0 – Res: Reserved Bits**

These bits are reserved bits in the AT43USB320A and always read as zero.

General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	**INTF1**	**INT F0**	–	–	–	–	–	–	**GIFR**
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: External Interrupt Flag1**

When an event on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $004. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 6 – INTF0: Interrupt Flag0 (Suspend/Resume Interrupt Flag)**

When an event on the INT0 (that is, a USB event-related interrupt) triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $002. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bits 5..0 – Res: Reserved Bits**

These bits are reserved bits in the AT43USB320A and always read as zero.

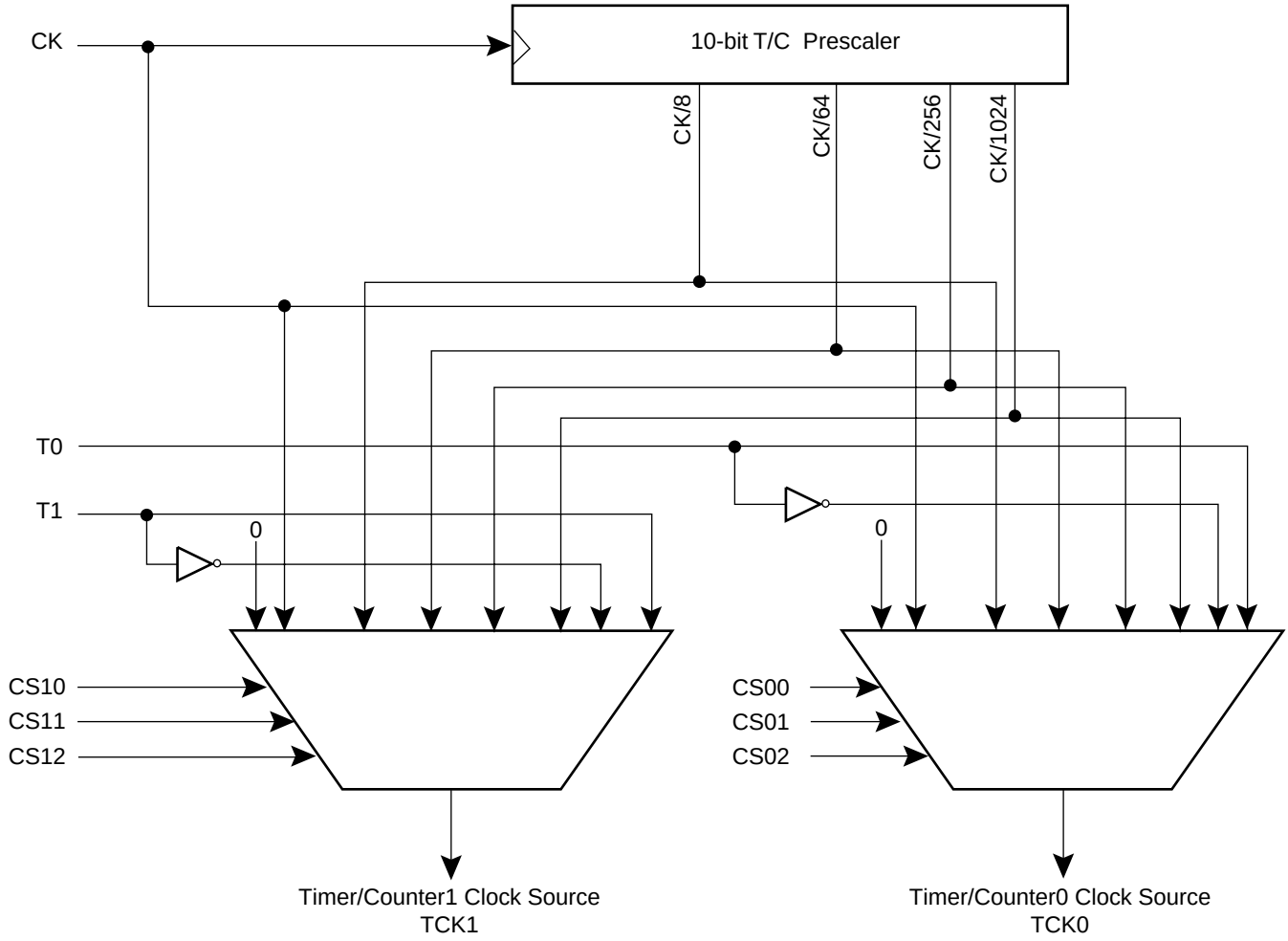
Timer/Counters

The AT43USB320A provides two general-purpose Timer/Counters - one 8-bit T/C and one 16-bit T/C. The Timer/Counters have individual prescaling selection from the same 10-bit prescaling timer. Both Timer/Counters can either be used as a timer with an internal clock timebase or as a counter with an external pin connection which triggers the counting.

Timer/Counter Prescaler

The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024 where CK is the oscillator clock. For the two Timer/Counters, added selections as CK, external source and stop, can be selected as clock sources.

Figure 9. Timer/Counter Prescaler



16-bit Timer/Counter1

Figure 11. Timer/Counter1 Block Diagram

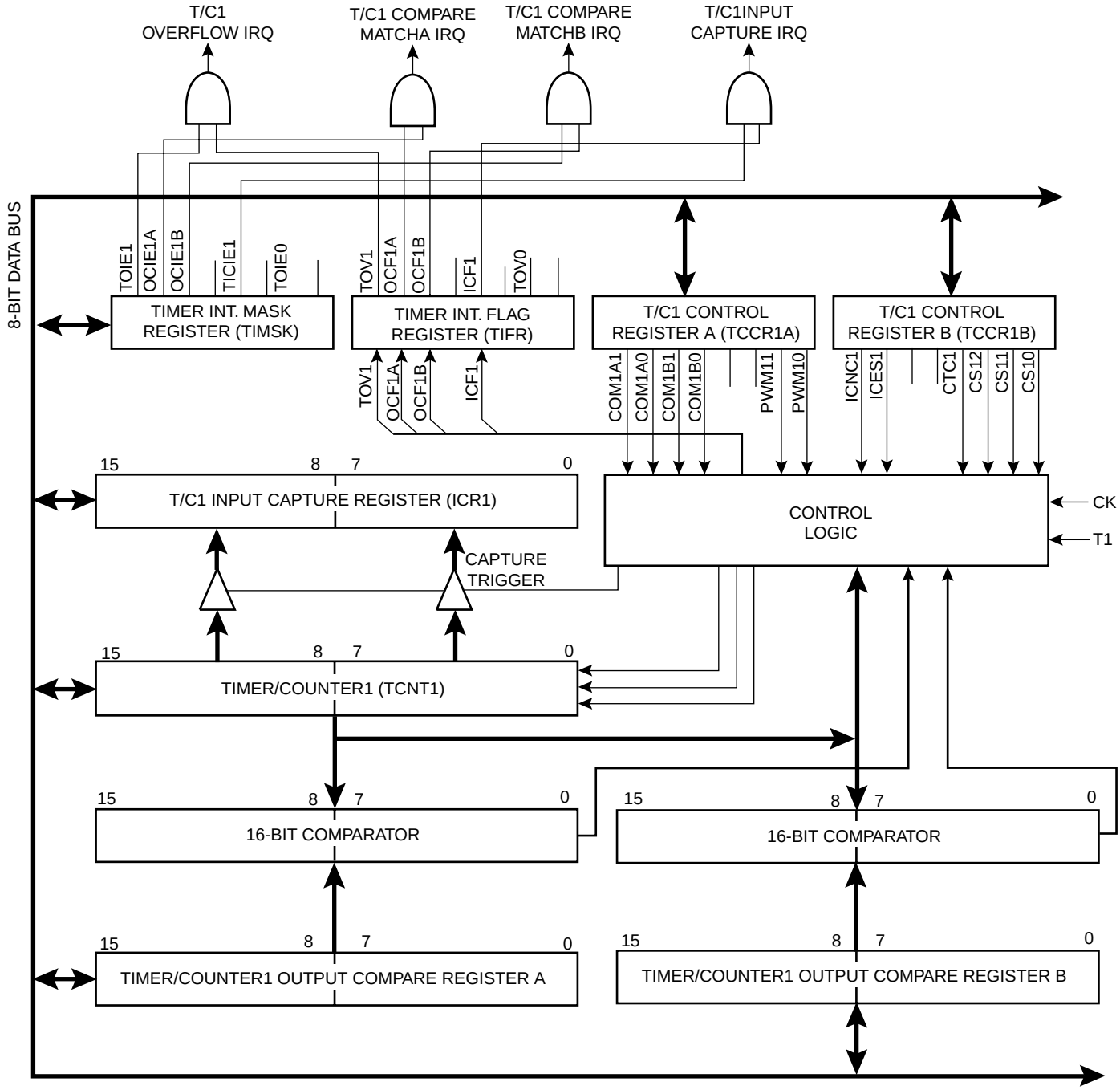


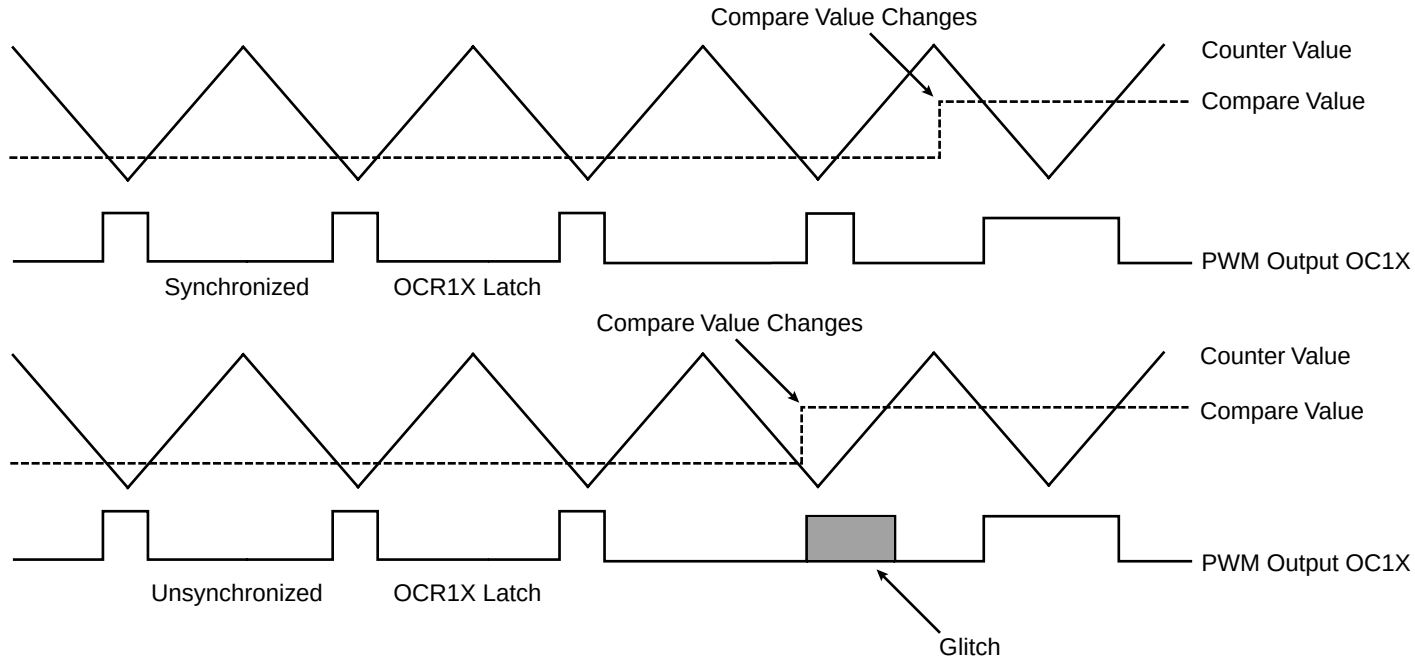
Table 17. Compare1 Mode Select in PWM Mode

COM1X1	COM1X0	Effect on OCX1
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, up-counting. Set on compare match, down-counting (non-inverted PWM).
1	1	Cleared on compare match, down-counting. Set on compare match, up-counting (inverted PWM).

Note: X = A or B

Note that in the PWM mode, the 10 least significant OCR1A/OCR1B bits, when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 13 for an example.

Figure 13. Effects on Unsynchronized OCR1 Latching



Note: X = A or B

During the time between the write and the latch operation, a read from OCR1A or OCR1B will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A/B

When the OCR1 contains $0000 or TOP, the output OC1A/OC1B is updated to low or high on the next compare match, according to the settings of COM1A1/COM1A0 or COM1B1/COM1B0. This is shown in Table 18.

Note: If the compare register contains the TOP value and the prescaler is not in use (CS12..CS10 = 001), the PWM output will not produce any pulse at all, because up-counting and down-counting values are reached simultaneously. When the prescaler is in use (CS12..CS10 = 001 or 000), the PWM output goes active when the counter reaches the TOP

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the AT43USB320A and will always read as zero.

- **Bit 4 – WDTOE: Watch Dog Turn-Off Enable**

This bit must be set (one) when the WDE bit is cleared. Otherwise, the watchdog will not be disabled. Once set, the hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a watchdog disable procedure.

- **Bit 3 – WDE: Watch Dog Enable**

When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set (one). To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDTOE and WDE. A logical one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logical 0 to WDE. This disables the watchdog.

- **Bits 2..0 – WDP2, WDP1, WDP0: Watch Dog Timer Prescaler 2, 1 and 0**

The WDP2, WDP1 and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Time-out Periods are shown in Table 19.

Table 19. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator cycles	Time-out
0	0	0	16K cycles	15 ms
0	0	1	32K cycles	30 ms
0	1	0	64K cycles	60 ms
0	1	1	128K cycles	0.12 s
1	0	0	256K cycles	0.24 s
1	0	1	512K cycles	0.49 s
1	1	0	1,024K cycles	0.97 s
1	1	1	2,048K cycles	1.9 s

Note: The WDR (Watchdog Reset) instruction should always be executed before the Watchdog Timer is enabled. This ensures that the reset period will be in accordance with the Watchdog Timer prescale settings. If the Watchdog Timer is enabled without reset, the watchdog timer may not start to count from zero. To avoid unintentional MCU reset, the Watchdog Timer should be disabled or reset before changing the Watchdog Timer Prescale Select.

SS Pin Functionality

When the SPI is configured as a master (MSTR in SPCR is set), the user can determine the direction of the SS pin. If SS is configured as an output, the pin is a general output pin which does not affect the SPI system. If SS is configured as an input, it must be held high to ensure Master SPI operation. If the SS pin is driven low by peripheral circuitry when the SPI is configured as master with the SS pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a slave. As a result of the SPI becoming a slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled and the I-bit in SREG are set, the interrupt routine will be executed.

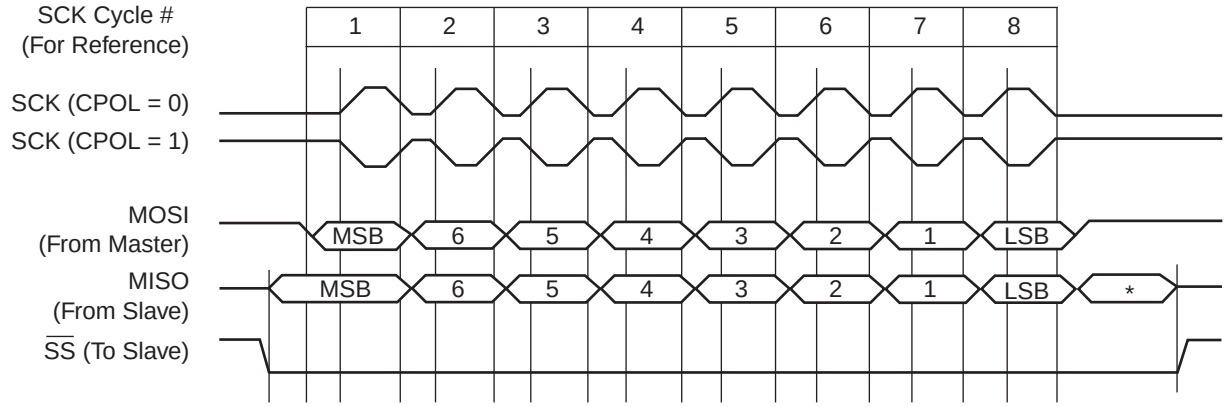
Thus, when interrupt-driven SPI transmittal is used in master mode, and there exists a possibility that SS is driven low, the interrupt should always check that the MSTR bit is still set. Once the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI master mode.

When the SPI is configured as a slave, the SS pin is always input. When SS is held low, the SPI is activated and MISO becomes an output if configured so by the user. All other pins are inputs. When SS is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the SS pin is brought high. If the SS pin is brought high during a transmission, the SPI will stop sending and receiving immediately and both data received and data sent must be considered as lost.

Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 17 and Figure 18.

Figure 17. SPI Transfer Format with CPHA = 0 and DORD = 0



Note: * Not defined but normally LSB of character just received.

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
$0D ($2D)	**SPIE**	**SPE**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**	**SPCR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR register is set and the global interrupts are enabled.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is set (one), the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is set (one), the LSB of the data word is transmitted first.

When the DORD bit is cleared (zero), the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI mode when set (one), and Slave SPI mode when cleared (zero). If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI master mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is set (one), SCK is high when idle. When CPOL is cleared (zero), SCK is low when idle. Refer to Figure 17 and Figure 18 for additional information.

- **Bit 2 – CPHA: Clock Phase**

Refer to Figure 17 or Figure 18 for the functionality of this bit.

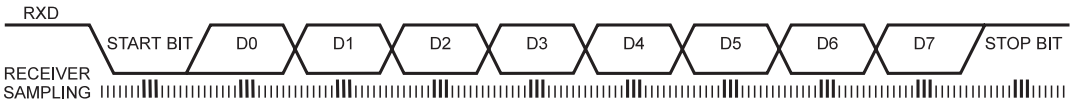
- **Bits 1,0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the Oscillator Clock frequency f_{CL} is shown in the following table:

Table 21. Relationship Between SCK and the Oscillator Frequency

SPR1	SPR0	SCK Frequency
0	0	3 MHz
0	1	750 kHz
1	0	187.5 kHz
1	1	93.75 kHz

Figure 21. Sampling Received Data



UART Control

UART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
$0D ($2C)	**MSB**	–	–	–	–	–	–	**LSB**	UDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The UDR register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data register is written. When reading from UDR, the UART Receive Data register is read.

UART Status Register – USR

Bit	7	6	5	4	3	2	1	0	
$0D ($2B)	**RXC**	**TXC**	**UDRE**	**FE**	**OR**	–	–	–	USR
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The USR register is a read-only register providing information on the UART status.

- **Bits 7 – RXC: UART Receive Complete**

This bit is set (one) when a received character is transferred from the Receiver Shift register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set (one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

- **Bit 6 – TXC: UART Transmit Complete**

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out an d no new data has been written to UDR. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCR is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical "1"to the bit.

- **Bit 2 – CHR9: 9-bit Characters**

When this bit is set (one) transmitted and received characters are 9 bits long plus start and stop bits. The ninth bit is read and written by using the RXB8 and TXB8 bits in UCR, respectively. The ninth data bit can be used as an extra stop bit or a parity bit.

- **Bit 1 – RXB8: Receive Data Bit 8**

When CHR9 is set (one), RXB8 is the ninth data bit of the received character.

- **Bit 0 – TXB8: Transmit Data Bit 8**

When CHR9 is set (one), TXB8 is the ninth data bit in the character to be transmitted.

Baud Rate Generator

The baud rate generator is a frequency divider that generates baud rates according to the following equation:

BAUD = SYSCLK/16(UBRR + 1)

- **BAUD = Baud rate**
- **SYSCLK = 16 MHz**
- **UBRR = Contents of the UART Baud Rate register, UBRR (0 – 255)**

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBRR settings in Table 22. UBRR values that yield an actual baud rate differing less than 2% from the target baud rate are boldface in the table. However, using baud rates that have more than 1% error is not recommended. High error ratings give less noise immunity.

Table 22. UBRR Settings

Baud Rate	UBRR	% Error
2400	416	0.08
4800	207	0.16
9600	103	0.16
14400	68	0.64
19200	51	0.16
28800	34	0.79
38400	25	0.16
57600	16	2.12
76800	12	0.16
115200	8	3.55

UART BAUD Rate Register – UBRR

Bit	7	6	5	4	3	2	1	0	
$09 ($29)	**MSB**							**LSB**	UBRR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UBRR register is an 8-bit read/write register that specifies the UART Baud Rate according to the equation on the previous page.

Port D as General Digital I/O

PDn, General I/O Pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is con-figured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PORTDn is set (one) when the pin is configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, the PORTDn has to be cleared (zero) or the pin has to configured as an output pin. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not active.

Table 28. DDDn Bits on Port D Pins

DDDn	PORTDn	I/O	Comment
0	0	Input	Tri-state (Hi-Z)
0	1	Input	Tri-state (Hi-Z)
1	0	Output	Push-Pull Zero Output
1	1	Output	Push-Pull One Output

Note: n: 7, 6...0, pin number.

Programming the USB Module

The USB hardware consists of two devices, hub and function, each with their own device address and endpoints. Its operation is controlled through a set of memory mapped registers. The exact configuration of the USB device is defined by the software and it can be programmed to operate as a compound device, or as a hub only or as a function only. The hub has the required control and interrupt endpoints. The number of external downstream ports is programmable from 0 to 4. The DP and DM pins of the unused port(s) must be connected to ground. The USB function has one control endpoint and 2 programmable endpoints. All the endpoints have their own 8-byte FIFO. If the hub is disabled, one extra endpoint becomes available to the function.

The USB Function

The USB function hardware is designed to operate in the single packet mode and to manage the USB protocol layer. It consists of a Serial Interface Engine (SIE), endpoint FIFOs and a Function Interface Unit (FIU). The SIE performs the following tasks: USB signaling detection/generation, data serialization/de-serialization, data encoding/decoding, bit stuffing and unstuffing, clock/data separation, and CRC generation/checking. It also decodes and manages all packet data types and packet fields.

The endpoint FIFO buffers the data to be sent out or data received. The FIU manages the flow of data between the SIE, FIFO and the internal microcontroller bus. It controls the FIFO and monitors the status of the transactions and interfaces to the CPU. It initiates interrupts and acts upon commands sent by the firmware.

The USB function hardware of the AT43USB320A makes the physical interface and the protocol layer transparent to the user. To start the process, the firmware must first enable the endpoints and which place them in receive mode by default. The device address by default is address 0. The USB function hardware then waits for a setup token from the host. When a valid the setup token is received, it automatically stores the data packet in endpoint 0 FIFO and responds with an ACK. It then notifies the microcontroller through an interrupt. The microcontroller reads the FIFO and parses the request.

Transactions for the non-control endpoints are even simpler. Once the endpoint is enabled, it waits for an IN or an OUT token depending whether it is programmed as an IN or OUT endpoint. For example, if it is an IN endpoint, the microcontroller simply loads the data into the endpoint's FIFO and sets a bit in the control and status register. The USB hardware will assemble the data in a USB packet and waits for an IN token. When it receives one, it automatically responds by transmitting the data packet and completes the transaction by waiting for the host's ACK. When one is received, the USB hardware will signal the microcontroller

that the transaction has been completed successfully. Retries and data toggles are performed automatically by the USB hardware. When the IN endpoint is not ready to send data, in the case where the microcontroller has not filled the FIFO, it will automatically respond with a NAK.

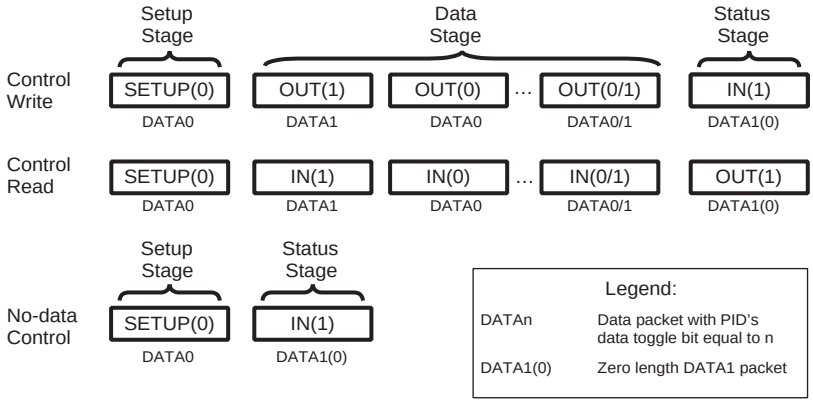
Similarly, an OUT endpoint will wait for an OUT token. When one is received, it will store the data in the FIFO, completes the transaction and interrupt the microcontroller, which then reads the FIFO and enables the endpoint for the next packet. If the FIFO is not cleared, the USB hardware will responds with a NAK.

A detailed description of how USB transactions are handled is described in the following sections. First for a control endpoint and then for non-control endpoints.

Control Transfers at Control Endpoint EP0

The description given below is for the function control endpoint, but applies to the hub control endpoint as well if the proper registers are used.

The following illustration describes the three possible types of control transfers – Control Write, Control Read and No-data control:



Interrupt/Bulk IN Transfers at Function Endpoint

The firmware must first condition the endpoint through the Endpoint Control Register, FENDP1/2_CNTR:

Set endpoint direction: set EPDIR

Set interrupt or bulk: EPTYPE = 11 or 10

Enable endpoint: set EPEN

The Function Interface Unit receives an IN token from the Host. The FIU responds with NAKs until TX_PACKET_READY is set. The FIU then sends the data in the FIFO upstream, retrying until it successfully receives an ACK from the host. Finally, the FIU clears the TX_PACKET_READY bit and asserts a TX_COMPLETE interrupt.

1. Read UISR
2. Read FCSR1/2
3. Clear TX_COMPLETE

   If more data: fill FIFO, set TX Packet Ready

   Wait for TX_COMPLETE interrupt

   If no more data: set DATA END in FCAR1/2
4. Set UIAR[FEP1/2 INTACK] to clear the interrupt source

Interrupt/Bulk OUT Transfers at Function Endpoint EP1 and 2

The firmware must first condition the endpoint through the Endpoint Control Register, FENDP1/2_CNTR:

Set endpoint direction: clear EPDIR

Set interrupt or bulk: EPTYPE = 11 or 10

Enable endpoint: set EPEN

The Function Interface Unit receives an OUT token from the Host with a DATA packet. The FIU places the incoming data into the FIFO, issues an ACK to the host, and asserts an RX_OUT interrupt.

1. Read UISR
2. Read FCSR1/2
3. Read FIFO
4. Clear RX_OUT

   If more data:

   Wait for RX_OUT interrupt

   If no more data: set DATA END
5. Set UIAR[FEP1/2 INTACK] to clear the interrupt source

Hub Endpoint 0 Data Register – HDR0

Function Endpoint 0..2 Data Register – FDR0..2

Bit	7	6	5	4	3	2	1	0	
$1FD7	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	HDR0
$1FD5	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	FDR0
$1FD4	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	FDR1
$1FD3	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	FDR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

This register is used to read data from or to write data to the Hub Endpoint 0 FIFO.

- **Bit 7..0 – FDAT7..0: FIFO Data**

Hub endpoint 1 has a single byte data register instead of a FIFO. This data register contains the hub and port status change bitmap. This data register is automatically updated by the USB hardware and is not accessible by the firmware. The bits in this register when read by the host will be:

Bit	7	6	5	4	3	2	1	0	
$	–	–	P5 SC	P4 SC	P3 SC	P2 SC	P1 SC	H SC	HDR1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 6 – Reserved**

These bits are reserved in the AT43USB320A and will read as zero.

- **Bit 5 – P5 SC: Port 5 Status Change**
- **Bit 4 – P4 SC: Port 4Status Change**
- **Bit 3 – P3 SC: Port 3 Status Change**
- **Bit 2 – P2 SC: Port 2 Status Change**
- **Bit 1 – P1 SC: Port 1 Status Change**
- **Bit 0 – H SC: Hub Status Change**

Selective Suspend and Resume

The host can selectively suspend and resume a port through the Set Port Feature (PORT_SUSPEND) and Clear Port Feature (PORT_SUSPEND).

A port enters the suspend state after the microcontroller interprets the suspend request and sets the appropriate bits of the Hub Port Control Register, HPCON. From this point on he hub repeater hardware is responsible for proper actions in placing Ports [1:4] in the suspend mode. For Port 5, the embedded function port, the hardware will stop responding to any normal bus traffic, but the microcontroller firmware must place all external circuitry associated with the function in the low-power state.

A port exits from the suspend state when the hub receives a Clear Port Feature (PORT_SUSPEND) or Set Port Feature (PORT_RESET). If the Clear Port Feature (PORT_SUSPEND) is directed towards Ports [1:4], the USB hardware drives a "K" downstream for at least 20 ms followed by a low speed EOP. It then places the port in the enabled state. A Clear Port Feature (PORT_SUSPEND) to Port 1 (the embedded function) causes the firmware to wait 20 ms, take the embedded function out of the suspended state and then enable the port.

The ports can also exit from the suspended state through a remote wakeup if this feature is enabled. For Ports [1:4], this means detection of a connect/disconnect or an upstream directed J to K signaling. Remote wakeup for the embedded function is initiated through an external interrupt at INT0.

Ordering Information

Ordering Code	Package	Operation Range
AT43USB320A-AC	100 LQFP	Commercial (0°C to 70°C)