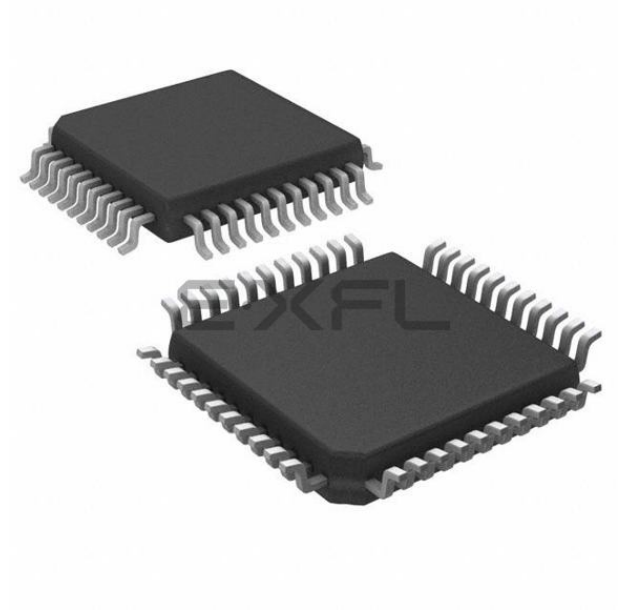

Welcome to E-XFL.COM

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 Microcontroller
Core Processor	8051
Program Memory Type	ROMless
Controller Series	AN213x
RAM Size	8K x 8
Interface	I²C, USB
Number of I/O	8
Voltage - Supply	3V ~ 3.6V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-PQFP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/an2136sc

Figure 13-5. Fast Transfer Mode Block Diagram 13-6
Figure 13-6. Fast Transfer Read Timing [Mode 00] 13-7
Figure 13-7. Fast Transfer Write Timing [Mode 00] 13-7
Figure 13-8. Fast Transfer Read Timing [Mode 01] 13-8
Figure 13-9. Fast Transfer Write Timing [MODE 01] 13-8
Figure 13-10. Fast Transfer Read Timing [Mode 10] 13-9
Figure 13-11. Fast Transfer Write Timing [Mode 10] 13-9
Figure 13-12. Fast Transfer Read Timing [Mode 11] 13-10
Figure 13-13. Fast Transfer Write Timing [Mode 11] 13-10
Figure 14-1. 44-Pin PQFP Package (Top View) 14-1
Figure 14-2. 44-Pin PQFP Package (Side View) 14-1
Figure 14-3. 44-Pin PQFP Package (Detail View) 14-2
Figure 14-4. 80-Pin PQFP Package (Top View) 14-3
Figure 14-5. 80-Pin PQFP Package (Side View) 14-3
Figure 14-6. 80-Pin PQFP Package (Detail View) 14-4
Figure 14-7. 48-Pin TQFP Package (Side View) 14-5
Figure 14-8. 48-Pin TQFP Package (Top View) 14-5
Figure 14-9. 48-Pin TQFP Package (Detail View) 14-6

1.13.3 EZ-USB Interrupt Endpoints

Interrupt endpoints are almost identical to bulk endpoints. Fourteen EZ-USB endpoints (EP1-EP7, IN, and OUT) may be used as interrupt endpoints. Interrupt endpoints have maximum packet sizes up to 64, and contain a “polling interval” byte in their descriptor to tell the host how often to service them. The 8051 transfers data over interrupt endpoints in exactly the same way as for bulk endpoints. Interrupt endpoints are described in Chapter 6, "EZ-USB Bulk Transfers."

1.13.4 EZ-USB Isochronous Endpoints

Isochronous endpoints deliver high bandwidth, time critical data over USB. Isochronous endpoints are used to stream data to devices such as audio DACs, and from devices such as cameras and scanners. Time of delivery is the most critical requirement, and isochronous endpoints are tailored to this requirement. Once a device has been granted an isochronous bandwidth slot by the host, it is guaranteed to be able to send or receive its data every frame.

EZ-USB contains 16 isochronous endpoints, numbered 8-15 (8IN-15IN, and 8OUT-15OUT). 1,024 bytes of FIFO memory are available to the 16 endpoints, and may be FIFO memory to provide double-buffering. Using double buffering, the 8051 reads OUT data from isochronous endpoint FIFOs containing data from the previous frame while the host writes current frame data into the other buffer. Similarly, the 8051 loads IN data into isochronous endpoint FIFOs that will be transmitted over USB during the next frame while the host reads current frame data from the other buffer. At every SOF the USB FIFOs and 8051 FIFOs switch, or *ping-pong*.

Isochronous transfers are described in Chapter 8, "EZ-USB Isochronous Transfers."

1.14 Fast Transfer Modes

The following versions of the EZ-USB have a fast transfer mode: AN2125SC, AN2126SC, AN2135SC, AN2136SC, and AN2131QC, that is, those versions that have a data bus (see Table 1-2). The fast transfer mode minimizes the transfer time from EZ-USB core also supplies external FIFO read and write strobes to synchronize the transfers.

Using the fast transfer mode, the 8051 transfers a byte of data between an internal FIFO and the external bus using a single 8051 MOVX instruction, which takes two cycles or 333 ns. Both Isochronous and Bulk endpoints can use this fast transfer mode.

2.5 EZ-USB Internal RAM

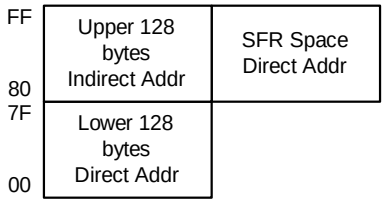

Figure 2-1. 8051 Registers

Like the standard 8051, the EZ-USB 8051 core contains 128 bytes of register RAM at 00-7F, and a partially populated SFR register space at 80-FF. An additional 128 indirectly addressed registers (sometimes called “IDATA”) are also available at 80-FF.

All internal EZ-USB RAM, which includes program/data memory, bulk endpoint buffer memory, and the EZ-USB register set, is addressed as *add-on* 8051 memory. The 8051 reads or writes these bytes as data using the MOVX (move external) instruction. Even though the MOVX instruction implies external memory, the EZ-USB RAM and register set is actually inside the EZ-USB chip. External memory attached to the AN2131Q address and data busses can also be accessed by the MOVX instruction. The EZ-USB core encodes its memory strobe and select signals (RD#, WR#, CS#, and OE#) to eliminate the need for external logic to separate the internal and external memory spaces.

2.6 I/O Ports

A standard 8051 communicates with its IO ports 0-3 through four Special Function Registers (SFRs). Standard 8051 IO pins are *quasi-bidirectional* with weak pullups that briefly drive high only when the pin makes a zero-to-one transition.

The EZ-USB core implements IO ports differently than a standard 8051, as described in Chapter 4, "EZ-USB Input/Output." Instead of using the 8051 IO ports and SFRs, the EZ-USB core implements a flexible IO system that is controlled *via* EZ-USB register set. Each EZ-USB IO pin functions identically, having the following resources:

- An output latch. Used when the pin is an output port.
- A bit that indicates the state of the IO pin, regardless of its configuration (input or output).

IN tokens (4) and (7) until the data is ready. Eventually, the 8051 fills the endpoint buffer with data, and then loads the endpoint's byte count register (INnBC) with the number of bytes in the packet (6). Loading the byte count re-arms the given endpoint. When the next IN token arrives (7) the USB core transfers the next data packet (8).

6.3 Interrupt Transfers

Interrupt transfers are handled just like bulk transfers.

The only difference between a bulk endpoint and an interrupt endpoint exists in the endpoint descriptor, where the endpoint is identified as type *interrupt*, and a *polling interval* is specified. The polling interval determines how often the USB host issues IN tokens to the interrupt endpoint.

6.4 EZ-USB Bulk IN Example

Suppose 220 bytes are to be transferred to the host using endpoint 2-IN. Further assume that MaxPacketSize of 64 bytes for endpoint 2-IN has been reported to the host during enumeration. Because the total transfer size exceeds the maximum packet size, the 8051 divides the 220-byte transfer into four transfers of 64, 64, 64, and 28 bytes.

After loading the first 64 bytes into IN2BUF (at 0x7C00), the 8051 loads the byte count register IN6BC with the value 64. Writing the byte count register instructs the EZ-USB core to respond to the next host IN token by transmitting the 64 bytes in the buffer. Until the byte count register is loaded to *arm* the IN transfer, any IN tokens issued by the host are answered by EZ-USB with NAK (Not-Acknowledge) tokens, telling the USB host that the endpoint is not yet ready with data. The host continues to issue IN tokens to endpoint 2-IN until data is ready for transfer—whereupon the EZ-USB core replaces NAKs with valid data.

When the 8051 initiates an IN transfer by loading the endpoint's byte count register, the EZ-USB core sets a busy bit to instruct the 8051 to hold off loading IN2BUF until the USB transfer is finished. When the IN transfer is complete and successfully acknowledged, the EZ-USB core resets the endpoint 2-IN busy bit and generates an endpoint 2-IN interrupt request. If the endpoint 2-IN interrupt is enabled, program control automatically vectors to the data transfer routine for further action (Autovectoring is enabled by setting AVEN=1; refer to Chapter 9, "EZ-USB Interrupts").

2. Write the INT2 interrupt vector.

```
; -----------------
; Interrupt Vectors
; -----------------
        org             43h                     ; int2 is the USB vector
        ljmp            USB_Jump_Table          ; Autovector will replace byte 45
```

Figure 6-8. INT2 Interrupt Vector

3. Write the interrupt service routine.

Put it anywhere in memory and the jump table in step 1 will automatically jump to it.

```
; ----------------------------
; USB Interrupt Service Routine
; ----------------------------
EP2OUT_ISR   push   dps             ; save both dptrs, dps, and acc
             push   dpl
             push   dph
             push   dpl1
             push   dph1
             push   acc

             mov    a,EXIF           ; clear USB IRQ (INT2)
             clr    acc.4
             mov    EXIF,a

             mov    dptr,#OUT07IRQ
             mov    a,#01000000b     ;  a “1” clears the IRQ bit
             movx   @dptr,a          ; clear OUT2 int request
             setb   got_EP2_data     ; set my flag

             pop    acc              ; restore vital registers
             pop    dph1
             pop    dpl1
             pop    dph
             pop    dpl
             pop    dps
             reti
```

Figure 6-9. Interrupt Service Routine (ISR) for Endpoint 2-OUT

In this example, the ISR simply sets the 8051 flag “got_EP2_data” to indicate to the background program that the endpoint requires service. Note that both data pointers and the DPS (Data Pointer Select) registers must be saved and restored in addition to the accumulator.

7.2 Control Endpoint EP0

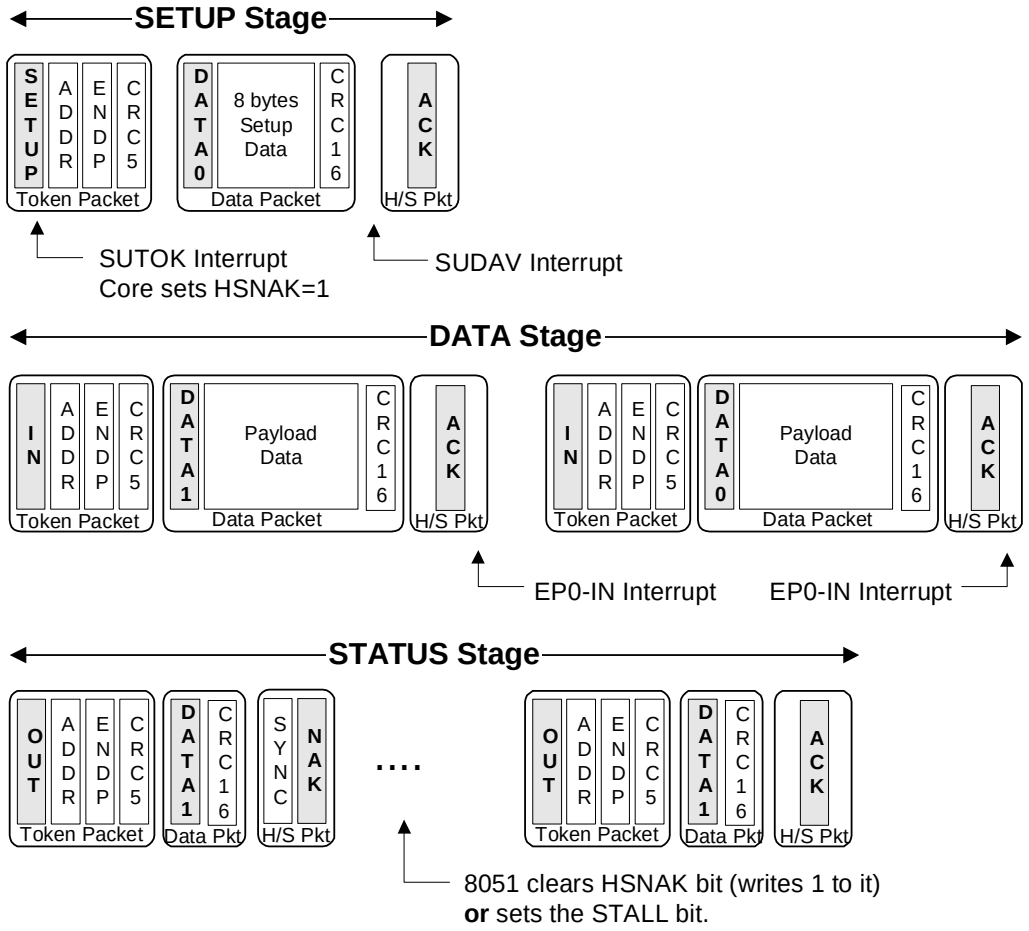

Figure 7-1. A USB Control Transfer (This One Has a Data Stage)

Endpoint zero accepts a special SETUP packet, which contains an 8-byte data structure that provides host information about the CONTROL transaction. CONTROL transfers include a final STATUS phase, constructed from standard PIDs (IN/OUT, DATA1, and ACK/NAK).

Some CONTROL transactions include all required data in their 8-byte SETUP Data packet. Other CONTROL transactions require more OUT data than will fit into the eight bytes, or require IN data from the device. These transactions use standard bulk-like transfers to move the data. Note in Figure 7-1 that the "DATA Stage" looks exactly like a bulk transfer. As with BULK endpoints, the endpoint zero byte count registers must be loaded to ACK the data transfer stage of a CONTROL transfer.

index. This constitutes the second phase of the three-phase CONTROL transfer. The core Packetizes the data into multiple data transfers as necessary.

4. Automatically checks for errors and re-transmits data packets if necessary.

5. Responds to the third (handshake) phase of the CONTROL transfer to terminate the operation.

The Setup Data Pointer can be used for any Get_Descriptor request; for example, Get_Descriptor-String. It can also be used for vendor-specific requests (that you define), as long as bytes 6-7 contain the number of bytes in the transfer (for step 1).

It is possible for the 8051 to do *manual* CONTROL transfers, directly loading the IN0BUF buffer with the various packets and keeping track of which SETUP phase is in effect. This would be a good USB training exercise, but not necessary due to the hardware support built into the EZ-USB core for CONTROL transfers.

For DATA stage transfers of fewer than 64 bytes, moving the data into the IN0BUF buffer and then loading the EP0INBC register with the byte count would be equivalent to loading the Setup Data Pointer. However, this would waste 8051 overhead because the Setup Data Pointer requires no byte transfers into the IN0BUF buffer.

7.3.4.2 Get Descriptor-Configuration

*Table 7-11. Get Descriptor-**Configuration***

Byte	Field	Value	Meaning	8051 Response
0	bmRequestType	**0x80**	IN, Device	*Set SUDPTR H-L to start of Configuration Descriptor table in RAM*
1	bRequest	**0x06**	"Get_Descriptor"	
2	wValueL	**CFG**	Config Number	
3	wValueH	**0x02**	Descriptor Type: Configuration	
4	wIndexL	0x00		
5	wIndexH	0x00		
6	wLengthL	**LenL**		
7	wLengthH	**LenH**		

8 EZ-USB Isochronous Transfers

8.1 Introduction

Isochronous endpoints typically handle time-critical, streamed data that is delivered or consumed in byte-sequential order. Examples might be audio data sent to a DAC over USB, or teleconferencing video data sent from a camera to the host. Due to the byte-sequential nature of this data, the EZ-USB chip makes isochronous data available as a single byte that represents the head or tail of an endpoint FIFO.

The EZ-USB chips that support isochronous transfers implement sixteen isochronous endpoints, IN8-IN15 and OUT8-OUT15. 1,024 bytes of FIFO memory may be distributed over the 16 endpoint addresses. FIFO sizes for the isochronous endpoints are programmable.

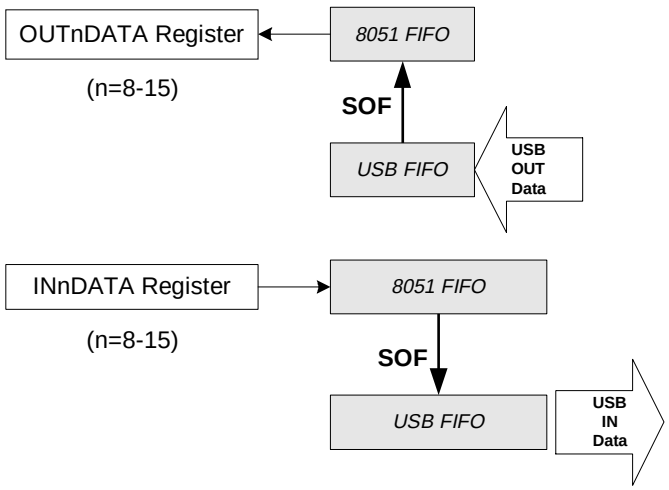

Figure 8-1. EZ-USB Isochronous Endpoints 8-15

The 8051 reads or writes isochronous data using sixteen FIFO data registers, one per endpoint. These FIFO registers are shown in Figure 8-1 as INnDATA (Endpoint n IN Data) and OUTnDATA (Endpoint n OUT Data).

The EZ-USB core provides a total of 2,048 bytes of FIFO memory (1,024 bytes, double-buffered) for ISO endpoints. This memory is in addition to the 8051 program/data memory, and normally exists outside of the 8051 memory space. The 1,024 FIFO bytes may be divided among the sixteen isochronous endpoints. The 8051 writes sixteen EZ-USB registers to allocate the FIFO buffer space to the isochronous endpoints. The 8051 also sets *endpoint valid* bits to enable isochronous endpoints.

8.2 Isochronous IN Transfers

IN transfers travel from device to host. Figure 8-2 shows the EZ-USB registers and bits associated with isochronous IN transfers.

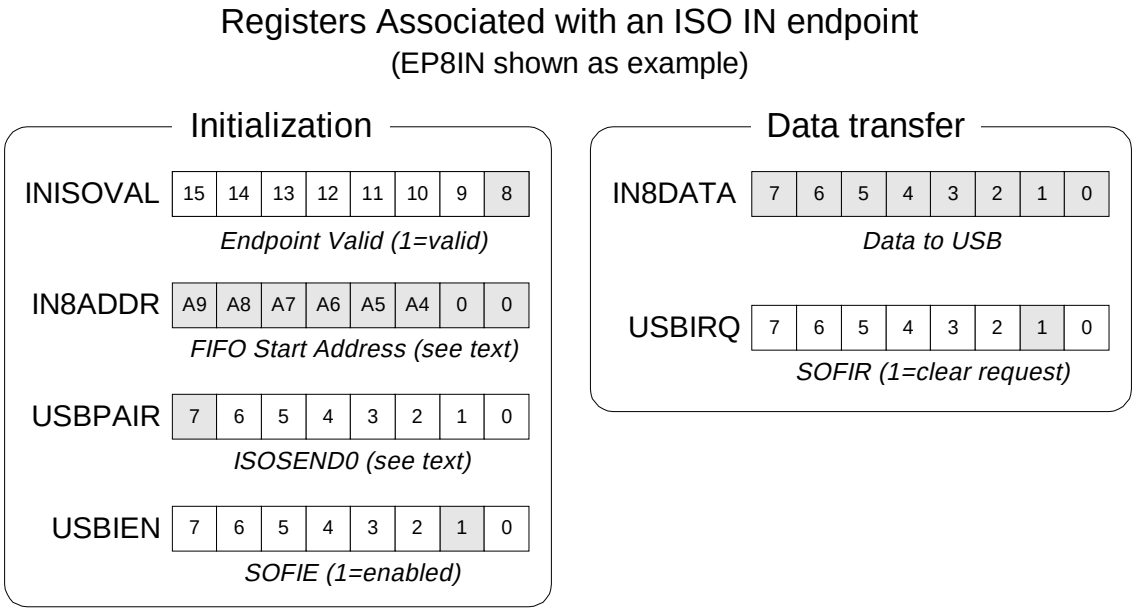

Figure 8-2. Isochronous IN Endpoint Registers

8.2.1 Initialization

To initialize an isochronous IN endpoint, the 8051 performs the following:

- Sets the endpoint valid bit for the endpoint.
- Sets the endpoint's FIFO size by loading a starting address (Section 8.4, "Setting Isochronous FIFO Sizes").
- Sets the ISOSEND0 bit in the USBPAIR register for the desired response.
- Enables the SOF interrupt. All isochronous endpoints are serviced in response to the SOF interrupt.

USB Signaling - These include 16 bulk endpoint interrupts, three interrupts not specific to a particular endpoint (SOF), Suspend, USB Reset), and two interrupts for CONTROL transfers (SUTOK, SUDAV). These interrupts share the USB interrupt (INT2). The AN2122/26 versions have an interrupt indicating that a bulk packet was NAKd.

I^2C Transfers - (INT3).

9.3 Wakeup Interrupt

Chapter 10, "EZ-USB Resets" describes suspend-resume signaling in detail, along with a code example that uses the Wakeup interrupt.

Briefly, the USB host puts a device into SUSPEND by stopping bus activity to the device. When the EZ-USB core detects 3 ms of no bus activity, it activates the USB suspend interrupt request. If enabled, the 8051 takes the suspend interrupt, does power management housekeeping (shutting down power to external logic), and finishes by setting SFR bit PCON.0. This signals the EZ-USB core to enter a very low power mode by turning off the 12-MHz oscillator.

When the 8051 sets PCON.0, it enters an idle state. 8051 execution is resumed by activation of any enabled interrupt. The EZ-USB chip uses a dedicated interrupt for USB Resume. When external logic pulls WAKEUP# low (for example, when a keyboard key is pressed or a modem receives a ring signal) or USB bus activity resumes, the EZ-USB core re-starts the 12-MHz oscillator, allowing the 8051 to recognize the interrupt and continue executing instructions.

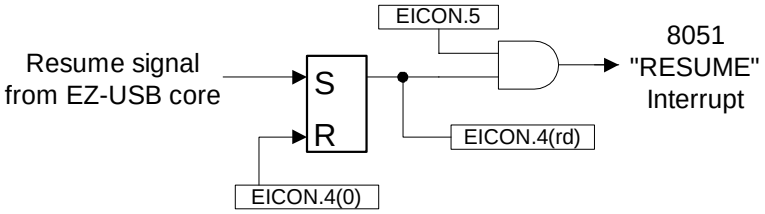

Figure 9-1. EZ-USB Wakeup Interrupt

Figure 9-1 shows the 8051 SFR bits associated with the RESUME interrupt. The EZ-USB core asserts the resume signal when the EZ-USB core senses a USB Global Resume, or when the EZ-USB WAKEUP# pin is pulled low. The 8051 enables the RESUME interrupt by setting EICON.5.

12.4 Isochronous Byte Counts

OUTnBCH **OUT(8-15) Byte Count High** **7F70-7F7F***

b7	b6	b5	b4	b3	b2	b1	b0
0	**0**	**0**	**0**	**0**	**0**	**BC9**	**BC8**
R	R	R	R	R	R	R	R
x	x	x	x	x	x	x	x

INnBCL **OUT(8-15) Byte Count Low** **7F70-7F7F***

b7	b6	b5	b4	b3	b2	b1	b0
BC7	**BC6**	**BC5**	**BC4**	**BC3**	**BC2**	**BC1**	**BC0**
R	R	R	R	R	R	R	R
x	x	x	x	x	x	x	x

* See Table 12-3 for individual endpoint buffer addresses.

Figure 12-4. Isochronous Byte Counts

Table 12-3. Isochronous Endpoint Byte Count Register Addresses

Address	Isochronous Data	Name
7F70	Endpoint 8 Byte Count High	OUT8BCH
7F71	Endpoint 8 Byte Count Low	OUT8BCL
7F72	Endpoint 9 Byte Count High	OUT9BCH
7F73	Endpoint 9 Byte Count Low	OUT9BCL
7F74	Endpoint 10 Byte Count High	OUT10BCH
7F75	Endpoint 10 Byte Count Low	OUT10BCL
7F76	Endpoint 11 Byte Count High	OUT11BCH
7F77	Endpoint 11 Byte Count Low	OUT11BCL
7F78	Endpoint 12 Byte Count High	OUT12BCH
7F79	Endpoint 12 Byte Count Low	OUT12BCL
7F7A	Endpoint 13 Byte Count High	OUT13BCH
7F7B	Endpoint 13 Byte Count Low	OUT13BCL
7F7C	Endpoint 14 Byte Count High	OUT14BCH
7F7D	Endpoint 14 Byte Count Low	OUT14BCL
7F7E	Endpoint 15 Byte Count High	OUT15BCH
7F7F	Endpoint 15 Byte Count Low	OUT15BCL

12.12 Endpoint 0 Control and Status Registers

EP0CS **Endpoint Zero Control and Status** **7FB4**

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	**OUTBSY**	**INBSY**	**HSNAK**	**EP0STALL**
R	R	R	R	R	R	R/W	R/W
0	0	0	0	1	0	0	0

IN0BC **Endpoint Zero IN Byte Count** **7FB5**

b7	b6	b5	b4	b3	b2	b1	b0
-	**BC6**	**BC5**	**BC4**	**BC3**	**BC2**	**BC1**	**BC0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

OUT0BC **Endpoint Zero OUT Byte Count** **7FC5**

b7	b6	b5	b4	b3	b2	b1	b0
-	**BC6**	**BC5**	**BC4**	**BC3**	**BC2**	**BC1**	**BC0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

Figure 12-25. Port Configuration Registers

These registers control EZ-USB CONTROL endpoint zero. Because endpoint zero is a bi-directional endpoint, the IN and OUT functionality is controlled by a single control and status (CS) register, unlike endpoints 1-7, which have separate INCS and OUTCS registers.

Bit 3: **OUTBSY** *OUT Endpoint Busy*

OUTBSY is a read-only bit that is automatically cleared when a SETUP token arrives. The 8051 sets the OUTBSY bit by writing a byte count to EPOUTBC.

INnBC	Endpoint (1-7) IN Byte Count	7FB7-7FC3*

b7	b6	b5	b4	b3	b2	b1	b0
-	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

* See Table 12-5 for individual byte count register addresses.

Figure 12-27. IN Byte Count Registers

The 8051 writes this register with the number of bytes it loaded into the IN endpoint buffer INnBUF. Writing this register also *arms* the endpoint by setting the endpoint BSY bit to 1.

Legal values for these registers are 0-64. A zero transfer size is used to terminate a transfer that is an integral multiple of MaxPacketSize. For example, a 256-byte transfer with maxPacketSize = 64, would require four packets of 64 bytes each plus one packet of 0 bytes.

The IN byte count should never be written while the endpoint's BUSY bit is set.

When the register pairing feature is used (Section 6, "EZ-USB Bulk Transfers") IN2BC is used for the EP2/EP3 pair, IN4BC is used for the EP4/EP5 pair, and IN6BC is used for the EP6/EP7 pair. In the *paired* (double-buffered) mode, after the first write to the even-numbered byte count register, the endpoint BSY bit remains at 0, indicating that only one of the buffers is full, and the other is still empty. The odd numbered byte count register is not used when endpoints are paired.

INISOVAL **Isochronous IN Endpoint Valid Bits** **7FE0**

b7	b6	b5	b4	b3	b2	b1	b0
IN15VAL	**IN14VAL**	**IN13VAL**	**IN12VAL**	**IN11VAL**	**IN10VAL**	**IN9VAL**	**IN8VAL**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	1	1	1

OUTISOVAL **Isochronous OUT Endpoint Valid Bits** **7FE1**

b7	b6	b5	b4	b3	b2	b1	b0
OUT15VAL	**OUT14VAL**	**OUT13VAL**	**OUT12VAL**	**OUT11VAL**	**OUT10VAL**	**OUT9VAL**	**OUT8VAL**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	1	1	1

Figure 12-37. Isochronous IN/OUT Endpoint Valid Bits Register

The 8051 sets VAL=1 for active endpoints, and VAL=0 for inactive endpoints. These bits instruct the EZ-USB core to return a “no response” if an invalid endpoint is addressed.

The default values of these registers are set to support all endpoints that exist in the default USB device (see Table 5-1).

AUTOPTRH **Auto Pointer Address High** **7FE3**

b7	b6	b5	b4	b3	b2	b1	b0
A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

AUTOPTRL **Auto Pointer Address Low** **7FE4**

b7	b6	b5	b4	b3	b2	b1	b0
A7	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

AUTODATA **Auto Pointer Data** **7FE5**

b7	b6	b5	b4	b3	b2	b1	b0
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

Figure 12-39. Auto Pointer Registers

These registers implement the EZ-USB *Autopointer.*

AUTOPTRH/L

The 8051 loads a 16-bit address into the AUTOPTRH/L registers. Subsequent reads or writes to the AUTODATA register increment the 16-bit value in these registers. The loaded address must be in internal EZ-USB RAM. The 8051 can read these registers to determine the address must be in internal EZ-USB RAM. The 8051 can read these registers to determine the address of the next byte to be accessed via the AUTODATA register.

AUTODATA

8051 data read or written to the AUTODATA register accesses the memory addressed by the AUTOPTRH/L registers, and increments the address *after* the read or write.

These registers allow FIFO access to the bulk endpoint buffers, as well as being useful for internal data movement. Chapter 6, "EZ-USB Bulk Transfers" and Chapter 8, "EZ-USB Isochronous Transfers" explain how to use the Autopointer for fast transfers to and from the EZ-USB endpoint buffers.

EZ-USB Registers

Addr	Name	Description	D7	D6	D5	D4	D3	D2	D1	D0			Notes
	Isochronous Byte Counts												
7F70	OUT8BCH	EP8 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F71	OUT8BCL	EP8 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F72	OUT9BCH	EP9 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F73	OUT9BCL	EP9 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F74	OUT10BCH	EP10 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F75	OUT10BCL	EP10 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F76	OUT11BCH	EP11 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F77	OUT11BCL	EP11 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F78	OUT12BCH	EP12 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F79	OUT12BCL	EP12 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F7A	OUT13BCH	EP13 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F7B	OUT13BCL	EP13 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F7C	OUT14BCH	EP14 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F7D	OUT14BCL	EP14 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
7F7E	OUT15BCH	EP15 Out Byte Count H	0	0	0	0	0	0	d9	d8			
7F7F	OUT15BCL	EP15 Out Byte Count L	d7	d6	d5	d4	d3	d2	d1	d0			
		7F80-7F91 (reserved)											
	CPU Registers												
7F92	CPUCS	Control & Status	rv3	rv2	rv1	rv0	0	0	CLK24OE	8051RES			rv[3..0] = chip rev
7F93	PORTACFG	Port A Configuration	RxD1out	RxD0out	FRD	FWR	CS	OE	T1out	T0out			0=port, 1=alt function
7F94	PORTBCFG	Port B Configuration	T2OUT	INT6	INT5	INT4	TxD1	RxD1	T2EX	T2			0=port, 1=alt function
7F95	PORTCCFG	Port C Configuration	RD	WR	T1	T0	INT1	INT0	TxD0	RxD0			0=port, 1=alt function
	Input-Output Port Registers												
7F96	OUTA	Output Register A	OUTA7	OUTA6	OUTA5	OUTA4	OUTA3	OUTA2	OUTA1	OUTA0			
7F97	OUTB	Output Register B	OUTB7	OUTB6	OUTB5	OUTB4	OUTB3	OUTB2	OUTB1	OUTB0			
7F98	OUTC	Output Register C	OUTC7	OUTC6	OUTC5	OUTC4	OUTC3	OUTC2	OUTC1	OUTC0			
7F99	PINSA	Port Pins A	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0			
7F9A	PINSB	Port Pins B	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0			
7F9B	PINSC	Port Pins C	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0			
7F9C	OEA	Output Enable A	OEA7	OEA6	OEA5	OEA4	OEA3	OEA2	OEA1	OEA0			0=off, 1=drive
7F9D	OEB	Output Enable B	OEB7	OEB6	OEB5	OEB4	OEB3	OEB2	OEB1	OEB0			0=off, 1=drive
7F9E	OEC	Output Enable C	OEC7	OEC6	OEC5	OEC4	OEC3	OEC2	OEC1	OEC0			0=off, 1=drive
7F9F	UART230	230Kbaud support	0	0	0	0	0	0	UART1	UART0			1 = 230Kbaud rate
	Isochronous Control/Status Registers												
7FA0	ISOERR	ISO OUT Endpoint Error	ISO15ERR	ISO14ERR	ISO13ERR	ISO12ERR	ISO11ERR	ISO10ERR	ISO9ERR	ISO8ERR			
7FA1	ISOCTL	Isochronous Control	*	*	*	*	PPSTAT	MBZ	MBZ	ISODISAB			"MBZ" = Must Be Zero
7FA2	ZBCOUT	Zero Byte Count bits	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8			
7FA3		(reserved)											
7FA4		(reserved)											
	I^{2}C Registers												
7FA5	I2CS	Control & Status	START	STOP	LASTRD	ID1	ID0	BERR	ACK	DONE			
7FA6	I2DAT	Data	d7	d6	d5	d4	d3	d2	d1	d0			
7FA7	I2CMODE	I2C STOP interrupt enable	0	0	0	0	0	0	STOPIE	0			1=Enable INT3 on STOP

A.3 Performance Overview

The 8051 core has been designed to offer increased performance by executing instructions in a 4-clock bus cycle, as opposed to the 12-clock bus cycle in the standard 8051 (see Figure A-1.). The shortened bus timing improves the instruction execution rate for most instructions by a factor of three over the standard 8051 architectures.

Some instructions require a different number of instruction cycles on the 8051 core than they do on the standard 8051. In the standard 8051, all instructions except for `MUL` and `DIV` take one or two instruction cycles to complete. In the 8051 core, instructions can take between one and five instruction cycles to complete. The average speed improvement for the entire instruction set is approximately 2.5X, calculated as follows:

Number of Opcodes	Speed Improvement
150	3.0X
51	1.5X
43	2.0X
2	2.4X
Total: 255	Average: 2.5X
Note: Comparison is for 8051 and standard 8051 running at the same clock frequency.	

accesses (bytes) needed to execute the instruction. In most cases, the number of bytes is equal to the number of instruction cycles required to complete the instruction. However, as indicated, there are some instructions (for example, `DIV` and `MUL`) that require a greater number of instruction cycles than memory accesses.

By default, the 8051 core timer/counters run at 12 clock cycles per increment so that timer-based events have the same timing as with the standard 8051. The timers can also be configured to run at 4 clock cycles per increment to take advantage of the higher speed of the 8051 core.

B.1.4 CPU Timing

As previously stated, an 8051 core instruction cycle consists of 4 *CLK24* cycles. Each *CLK24* cycle forms a CPU cycle. Therefore, an instruction cycle consists of 4 CPU cycles: C1, C2, C3, and C4, as illustrated in Figure B-3. Various events occur in each CPU cycle, depending on the type of instruction being executed. The labels C1, C2, C3, and C4 in timing descriptions refer to the 4 CPU cycles within a particular instruction cycle.

The execution for instruction *n* is performed during the fetch of instruction *n*+1. Data writes occur during fetch of instruction *n*+2. The level sensitive interrupts are sampled with the rising edge of *CLK24* at the end of C3.

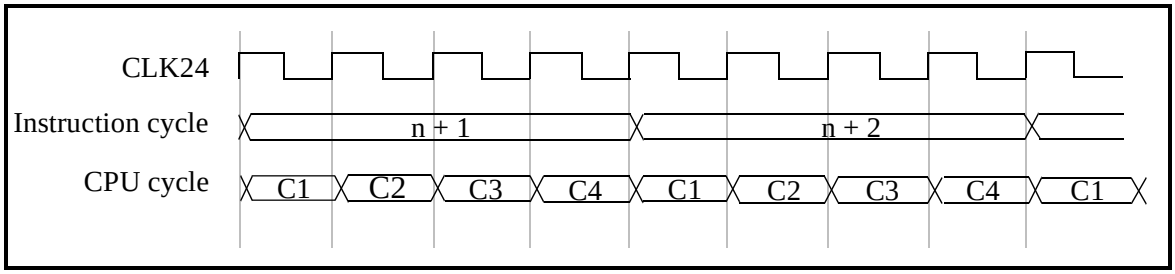

Figure B-3 CPU Timing for Single-Cycle Instruction

B.1.5 Stretch Memory Cycles (Wait States)

The stretch memory cycle feature enables application software to adjust the speed of data memory access. The 8051 core can execute the `MOVX` instruction in as few as 2 instruction cycles. However, it is sometimes desirable to stretch this value; for example to access slow memory or slow memory-mapped peripherals such as UARTs or LCDs.

The three LSBs of the Clock Control Register (at SFR location 8Eh) control the stretch value. You can use stretch values between zero and seven. A stretch value of zero adds zero instruction cycles, resulting in `MOVX` instructions executing in two instruction cycles. A stretch value of seven adds seven instruction cycles, resulting in `MOVX` instructions executing in nine instruction cycles. The stretch value can be changed dynamically under program control.

C.3.5 Mode 3

Mode 3 provides asynchronous, full-duplex communication, using a total of 11 bits: 1 start bit, 8 data bits, a programmable 9th bit, and 1 stop bit. The data bits are transmitted and received LSB first.

The mode 3 transmit and operations are identical to mode 2. The mode 3 baud rate generation is identical to mode 1. That is, mode 3 is a combination of mode 2 protocol and mode 1 baud rate. Figure C-15.illustrates the mode 3 transmit timing. Figure C-16.illustrates the mode 3 receive timing.

Mode 3 operation is identical to that of the standard 8051 when Timers 1 and 2 use CLK24/12 (the default).

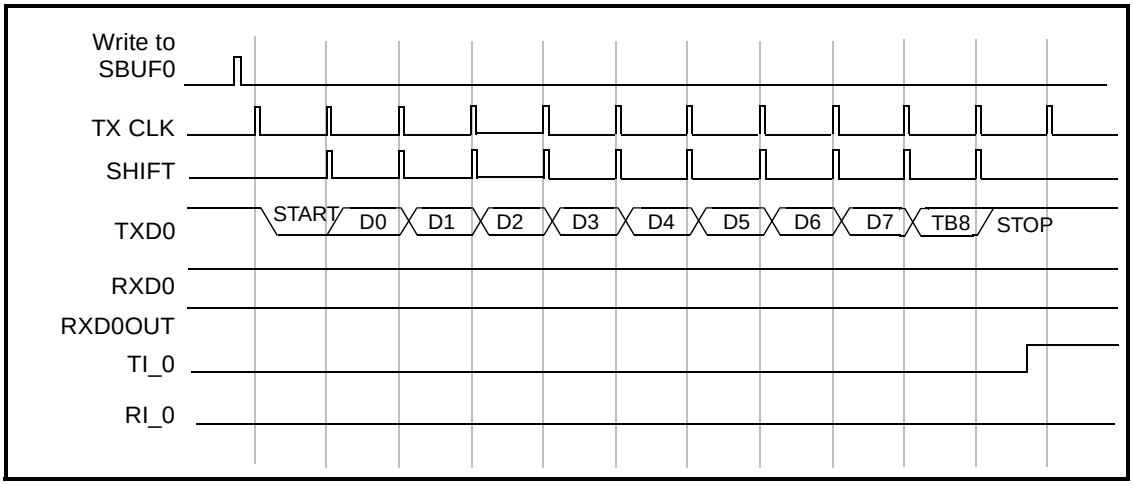

Figure C-15. Serial Port 0 Mode 3 Transmit Timing

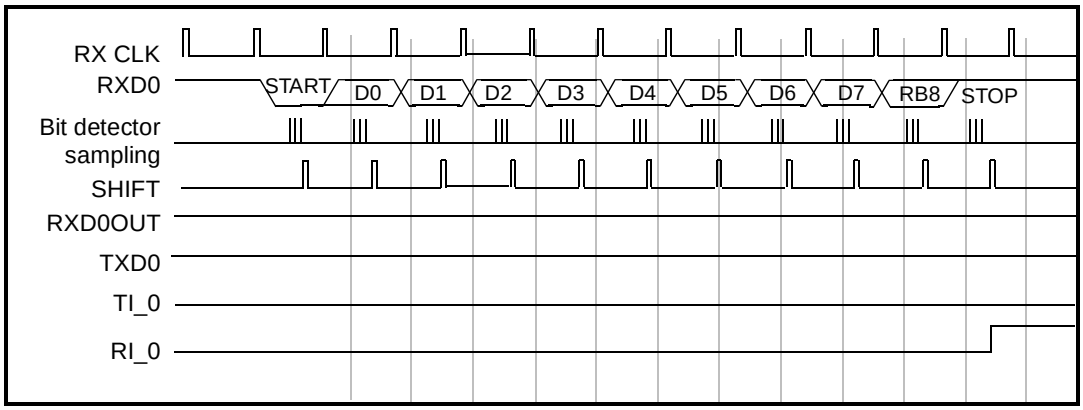

Figure C-16. Serial Port 0 Mode 3 Receive Timing

mode and shuts down the 24 MHz oscillator. See Chapter 11, "EZ-USB Power Management" for a full description of the Suspend/Resume process.

Table C-20. PCON Register - SFR 87h

Bit	Function
PCON.7	SMOD0 - Serial Port 0 baud rate double enable. When SMOD0 = 1, the baud rate for Serial Port 0 is doubled.
PCON.6-4	Reserved.
PCON.3	GF1 - General purpose flag 1. Bit-addressable, general purpose flag for software control.
PCON.2	GF0 - General purpose flag 0. Bit-addressable, general purpose flag for software control.
PCON.1	This bit should always be set to 0.
PCON.0	IDLE - Idle mode select. Setting the IDLE bit places the 8051 in idle mode.