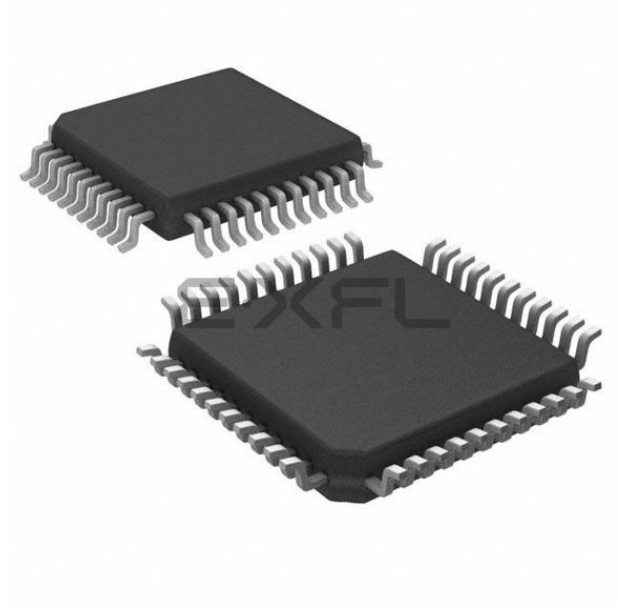

Welcome to E-XFL.COM

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/an2135sc

Development Kit — Getting Started

Documentation for the EZ-USB™ Xcelerator™ Development it. Includes an overview of the kit, descriptions of kit components with installation instructions, and details about the development board.

Technical Reference

Documentation of the EZ-USB controller. Includes details about the CPU, memory, input/output, ReNumeration™, bulk transfers, endpoint zero, isochronous transfers, interrupts, resets, power management, registers, AC/DC parameters, and packages.

Appendices

Documentation for the 8051 enhanced core. Includes an introduction, an architectural overview, and a hardware description.

Registers

EZ-USB register maps.

Technical Support:
Phone: (858) 613-7929
E-mail: usbapps@cypress.com
Website:
www.cypress.com

EZ-USB
Technical Reference Manual
Version 1.9
May 2000

1.4 Tokens and PIDs

In this manual, you will read statements like, “When the host sends an IN token...” or “The device responds with an ACK.” What do these terms mean? A USB transaction consists of data packets identified by special codes called Packet IDs or PIDs. A PID signifies what kind of packet is being transmitted. There are four PID types, as shown in Table 1-1.

Table 1-1. USB PIDs

PID Type	PID Name
Token Data	IN, OUT, SOF, SETUP, DATA0, DATA1
Handshake	ACK, NAK, STALL
Special	PRE

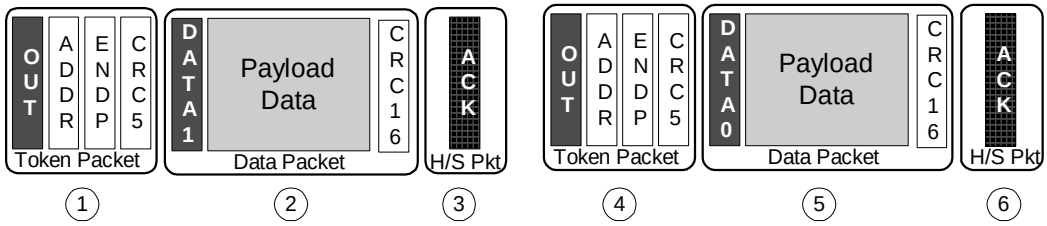

Figure 1-3. USB Packets

Figure 1-3 illustrates a USB transfer. Packet j is an OUT token, indicated by the OUT PID. The OUT token signifies that data from the host is about to be transmitted over the bus. Packet !k contains data, as indicated by the DATA1 PID. Packet l is a handshake packet, sent by the device using the ACK (acknowledge) PID to signify to the host that the device received the data error-free.

Continuing with Figure 1-3, a second transaction begins with another OUT token m, followed by more data n, this time using the DATA0 PID. Finally, the device again indicates success by transmitting the ACK PID in a handshake packet o.

Why two DATA PIDs, DATA0 and DATA1? It’s because the USB architects took error correction very seriously. As mentioned previously, the ACK handshake is a signal to the host that the peripheral received data without error (the CRC portion of the packet is used to detect errors). But what if a handshake packet itself is garbled in transmission? To detect this, each side, host and device maintains a *data toggle* bit, which is toggled between data packet transfers. The state of this internal toggle bit is compared with the

Table 1-3. EZ-USB Series 2100 Pinouts by Pin Function

2131Q	2121S 2122S 2131S	2125S 2126S 2135S 2136S	2122T	2126T	Name	Type	Default	Description
71	N/A	N/A	N/A	N/A	PA3 or CS#	I/O	I (PA3)	Multiplexed pin whose function is selected by the CS bit of the PORTACFG register. If CS=0, the pin is the bi-directional I/O port pin PA3. If CS=1, the pin is an active-low chip select for external memory. If the CS# pin is used, it should be externally pulled up to Vcc to ensure that the write strobe is inactive (high) at power-on.
73	39	39	N/A	N/A	PA4 or FWR#	I/O	I (PA4)	Multiplexed pin whose function is selected by the FWR (Fast Write) bit of the PORTAFCG register. If FWR=0, the pin is the bi-directional I/O port pin PA4. If FWR=1, the pin is the write strobe for an external FIFO. If the FWR# pin is used, it should be externally pulled up to Vcc to ensure that the write strobe is inactive (high) at power-on.
74	40	40	N/A	N/A	PA5 or FRD#	I/O	I (PA5)	Multiplexed pin whose function is selected by the FRD (Fast Read) bit of the PORTAFCG register. If FRD=0, the pin is the bi-directional I/O port pin PA5. If FRD=1, the pin is the read strobe for an external FIFO. If the FRD# pin is used, it should be externally pulled up to Vcc to ensure that the write strobe is inactive (high) at power-on.
75	N/A	N/A	44	44	PA6 or RXD0OUT	I/O	I (PA6)	Multiplexed pin whose function is selected by the RXD0OUT bit of the PORTAFCG register. If RXD0OUT=0 (default), the pin is the bi-directional I/O port bit PA6. If RXD0OUT=1, the pin is the active-high RXD0OUT signal from 8051 UART0. If RXD0OUT is selected and UART0 is in mode 0, this pin provides the output data for UART0 only when it is in sync mode. Otherwise, it is a 1.
76	N/A	N/A	8	8	PA7 OR RXD1OUT	I/O	I (PA7)	Multiplexed pin whose function is selected by the RXD1OUT bit of the PORTAFCG register. If RXD1OUT=0 (default), the pin is the bi-directional I/O port bit PA7. If RXD1OUT=1, the pin is the active-high RXD1OUT signal from 8051 UART1. When RXD1OUT is selected and UART1 is in mode 0, this pin provides the output data for UaRT1 only when it is in sync mode. In modes 1, 2, and 3, this pin is a 1.

Table 1-3. EZ-USB Series 2100 Pinouts by Pin Function

2131Q	2121S 2122S 2131S	2125S 2126S 2135S 2136S	2122T	2126T	Name	Type	Default	Description
40	20	20	22	22	PC6 or WR#	I/O	I (PC6)	Multiplexed pin whose function is selected by the WR bit of the PORTCCFG register. If WR=0, the pin is the bi-directional I/O port bit PC6. If WR=1, the pin is the active-low write signal for external memory. If the WR# signal is used, it should be externally pulled up to Vcc to ensure that the write strobe is inactive at power-on.
41	21	21	23	23	PC7 or RD#	I/O	I (PC7)	Multiplexed pin whose function is selected by the RD bit of the PORTCCFG register. If RD#=0, the pin is the bi-directional I/O port bit PC7. If RD#=1, the pin is the active-low read signal for external memory. If the RD# signal is used, it should be externally pulled up to Vcc to ensure that the read strobe is inactive at power-on.
4	2	2	2	2	CLK24	Output		**24-MHz clock**, phase locked to the 12-MHz input clock. It operates at 12 MHz in 12-MHz mode (48-pin package). Output is disabled by setting the OUTCLKEN bit = 0 in the CPUCS register.
66	37	37	40	40	WAKEUP#	Input	N/A	**USB Wakeup**. If the 8051 is in suspend, a high to low edge on this pin starts up the oscillator and interrupts the 8051 to allow it to exit the suspend mode. Holding WAKEUP# LOW inhibits the EZ-USB chip from entering the suspend state.
65	36	36	39	39	SCL	OD	Z	**I^2C Clock**. Pull up to Vcc with a 2.2K-ohm resistor, even if no I^2C device is connected.
64	35	35	38	38	SDA	OD	Z	**I^2C Data**. Connect to Vcc with a 2.2K-ohm resistor even if no I^2C device is connected.
2, 22, 42, 62	11, 22, 33, 44	11, 22, 33, 44	12, 24, 36, 48	12, 24, 36, 48	Vcc		N/A	**Vcc**. 3.3V power source.
3, 5, 6, 13, 14, 17, 23, 43, 56, 63, 72, 78	1, 3, 4, 5, 6, 12, 23, 34, 38	1, 3, 4, 5, 6, 12, 23, 34, 38	1, 3, 4, 5, 6, 13, 25, 37, 41	1, 3, 4, 5, 6, 13, 25, 37, 41	GND		N/A	Ground. Note: On the 80-pin package, pins 5, 6, 13, 14, and 72 are test pins that must be grounded for normal operation. Driving pin 72 high floats all functional pins for automated board test. The corresponding pins on the 44-pin package are pins 3, 4, 5, 6, and 38. Driving pin 38 high floats all functional pins for automated board test. The corresponding pins on the 48-pin package are pins 3, 4, 5, 6, and 41. Driving pin 41 high floats all functional pins for automated board testing.
N/A	N/A	N/A	15	15	CPU12MHZ		N/A	This input controls the speed of the 8051: - Tied High - 12 MHz - Tied Low - 24 MHz
67	N/A	N/A	N/A	N/A	NC		N/A	This pin must be left unconnected.

- An output enable bit that causes the IO pin to be driven from the output latch.
- An alternate function bit that determines whether the pin is general IO or a special 8051 or EZ-USB function.

The SFRs associated with 8051 ports 0-3 are not implemented in EZ-USB. These SFR addresses include P0 (0x80), P1 (0x90), P2 (0xA0), and P3 (0xB0). Because P2 is not implemented, the MOVX@R0/R1 instruction takes the upper address byte from an added Special Function Register (SFR) at location 0x92. This register is called “MPAGE” in the Appendices.

2.7 Interrupts

All standard 8051 interrupts are supported in the enhanced 8051 core. Table 2-1 shows the existing and added 8051 interrupts, and indicates how the added ones are used.

Table 2-1. EZ-USB Interrupts

Standard 8051 Interrupts	Enhanced 8051 Interrupts	Used As
INT0		Device Pin INT0#
INT1		Device Pin INT1#
Timer 0		Internal, Timer 0
Timer 1		Internal, Timer 1
Tx0 & Rx0		Internal, UART0
	INT2	Internal, USB
	INT3	Internal, I²C Controller
	INT4	Device Pin, PB4/INT4
	INT5	Device Pin, PB5/INT5#
	INT6	Device Pin, PB6/INT6
	PF1	Device Pin, USB WAKEUP#
	Tx1 & Rx1	Internal, UART1
	Timer 2	Internal, Timer 2

The EZ-USB chip uses 8051 INT2 for 21 different USB interrupts: 16 bulk endpoints plus SOF, Suspend, SETUP Data, SETUP Token, and USB Bus Reset. To help the 8051 determine which interrupt is active, the EZ-USB core provides a feature called Autovectoring. The core inserts an address byte into the low byte of the 3-byte jump instruction found at the 8051 INT2 vector address. This second level of vectoring automatically transfers control to the appropriate USB ISR. The Autovector mechanism, as well as the EZ-USB interrupt system is the subject of Chapter 9, "EZ-USB Interrupts."

2.8 Power Control

The EZ-USB core implements a power-down mode that allows it to be used in USB bus powered devices that must draw no more than 500 µA when suspended. Power control is accomplished using a combination of 8051 and EZ-USB core resources. The mechanism by which EZ-USB powers down for suspend, and then re-powers to resume operation, is described in detail in Chapter 11, "EZ-USB Power Management."

A suspend operation uses three 8051 resources, the *idle* mode and two interrupts. Many enhanced 8051 architectures provide power control similar (or identical) to the EZ-USB enhanced 8051 core.

A USB suspend operation is indicated by a lack of bus activity for 3 ms. The EZ-USB core detects this, and asserts an interrupt request via the USB interrupt (8051 INT2). The ISR (Interrupt Service Routine) turns off external sub-systems that draw power. When ready to suspend operation, the 8051 sets an SFR bit, PCON.0. This bit causes the 8051 to suspend, waiting for an interrupt.

When the 8051 sets PCON.0, a control signal from the 8051 to the EZ-USB core causes the core to shut down the 12-MHz oscillator and internal PLL. This stops all internal clocks to allow the EZ-USB core and 8051 to enter a very low power mode.

The suspended EZ-USB chip can be awakened two ways: USB bus activity may resume, or an EZ-USB pin (WAKEUP#) can be asserted to activate a USB *Remote Wakeup*. Either event triggers the following chain of events:

- The EZ-USB core re-starts the 12-MHz oscillator and PLL, and waits for the clocks to stabilize
- The EZ-USB core asserts a special, high-priority 8051 interrupt to signal a 'resume' interrupt.
- The 8051 vectors to the resume ISR, and upon completion resumes executing code at the instruction following the instruction that set the PCON.0 bit to 1.

7.2 Control Endpoint EP0

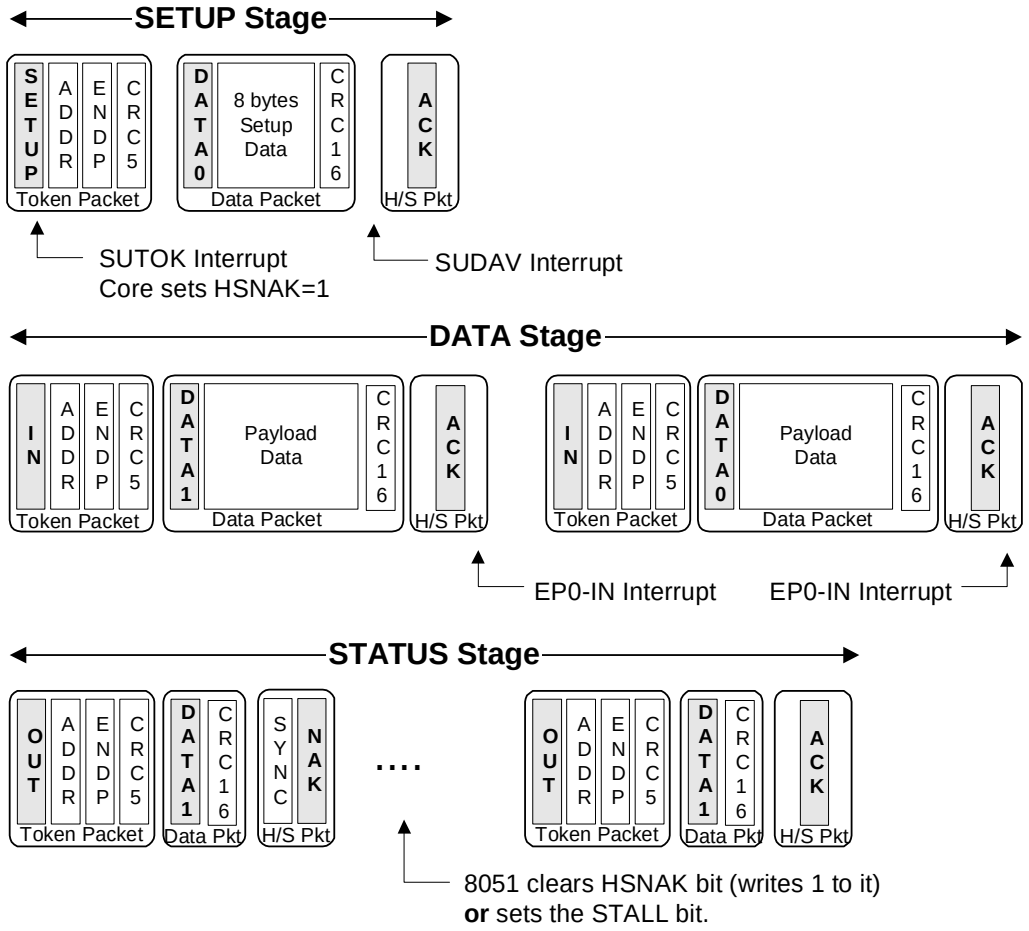

Figure 7-1. A USB Control Transfer (This One Has a Data Stage)

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.

The CONTROL transaction starts in the usual way, with the EZ-USB core transferring the eight bytes in the SETUP packet into RAM at SETUPDAT and activating the SUDAV interrupt request. The 8051 decodes the Get_Descriptor request, and responds by clearing the HSNAK bit (by writing "1" to it), and then loading the SUDPTR registers with the address of the requested descriptor. Loading the SUDPTRL register causes the EZ-USB core to automatically respond to two IN transfers with 64 bytes and 27 bytes of data using SUDPTR as a base address, and then to respond to (ACK) the STATUS stage.

The usual endpoint zero interrupts, SUDAV and EP0IN, remain active during this automated transfer. The 8051 normally disables these interrupts because the transfer requires no 8051 intervention.

Three types of descriptors are defined: Device, Configuration, and String.

7.3.4.1 Get Descriptor-Device

*Table 7-10. Get Descriptor-**Device***

Byte	Field	Value	Meaning	8051 Response
0	bmRequestType	**0x80**	IN, Device	*Set SUDPTR H-L to start of*
1	bRequest	**0x06**	"Get_Descriptor"	*Device Descriptor table in RAM*
2	wValueL	0x00		
3	wValueH	**0x01**	Descriptor Type: Device	
4	wIndexL	0x00		
5	wIndexH	0x00		
6	wLengthL	**LenL**		
7	wLengthH	**LenH**		

As illustrated in Figure 7-5, the 8051 loads the 2-byte SUDPTR with the starting address of the Device Descriptor table. When SUDPTRL is loaded, the EZ-USB core performs the following operations:

1. Reads the requested number of bytes for the transfer from bytes 6 and 7 of the SETUP packet (**LenL** and **LenH** in Table 7-11).

2. Reads the requested string's descriptor to determine the actual string length.

3. Sends the smaller of (a) the requested number of bytes or (b) the actual number of bytes in the string, over IN0BUF using the Setup Data Pointer as a data table

7.3.4.3 Get Descriptor-String

*Table 7-12. Get Descriptor-**String***

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**	String Number	
3	wValueH	**0x02**	Descriptor Type: String	
4	wIndexL	0x00	(Language ID L)	
5	wIndexH	0x00	(Language ID H)	
6	wLengthL	**LenL**		
7	wLengthH	**LenH**		

Configuration and string descriptors are handled similarly to device descriptors. The 8051 firmware reads byte 2 of the SETUP data to determine which configuration or string is being requested, loads the corresponding table pointer into SUDPTRH-L, and the EZ-USB core does the rest.

7.3.5 Set Descriptor

*Table 7-13. Set Descriptor-**Device***

Byte	Field	Value	Meaning	8051 Response
0	bmRequestType	**0x00**	OUT, Device	*Read device descriptor data over OUT0BUF*
1	bRequest	**0x07**	"Set_Descriptor"	
2	wValueL	0x00		
3	wValueH	0x01	Descriptor Type: Device	
4	wIndexL	0x00		
5	wIndexH	0x00		
6	wLengthL	**LenL**		
7	wLengthH	**LenH**		

7.3.8 Set Interface

This confusingly named USB command actually sets and reads back *alternate settings* for a specified interface.

USB devices can have multiple concurrent interfaces. For example a device may have an audio system that supports different sample rates, and a graphic control panel that supports different languages. Each interface has a collection of endpoints. Except for endpoint 0, which each interface uses for device control, endpoints may not be shared between interfaces.

Interfaces may report alternate settings in their descriptors. For example, the audio interface may have setting 0, 1, and 2 for 8-KHz, 22-KHz, and 44-KHz sample rates, and the panel interface may have settings 0 and 1 for English and Spanish. The Set/Get_Interface requests select between the various alternate settings in an interface.

Table 7-18. Set Interface (Actually, Set Alternate Setting AS for Interface IF)

Byte	Field	Value	Meaning	8051 Response
0	bmRequestType	**0x00**	OUT, Device	*Read and stash byte 2 (AS) for Interface IF, change setting for Interface IF in firmware*
1	bRequest	**0x0B**	"Set_Interface"	
2	wValueL	**AS**	Alt Setting Number	
3	wValueH	0x00		
4	wIndexL	**IF**	For this interface	
5	wIndexH	0x00		
6	wLengthL	0x00		
7	wLengthH	0x00		

The 8051 should respond to a Set_Interface request by performing the following steps:

- Perform the internal operation requested (such as adjusting a sampling rate).
- Reset the data toggles for every endpoint in the interface.
- For an IN endpoint, clear the busy bit for every endpoint in the interface.
- For an OUT endpoint, load any value into the byte count register for every endpoint in the interface.
- Clear the HSNAK bit (by writing "1" to it) to terminate the Set_Feature/Stall CONTROL transfer.

```
tb     EICON.5     ; enable Resume interrupt
```

The 8051 reads the RESUME interrupt request bit in EICON.4, and clears the interrupt request by writing a zero to EICON.4.

```
Resume_isr:   clr      EICON.4     ; clear the 8051 W/U
                                   ; interrupt request
              reti
```

9.14 I²C STOP Complete Interrupt - (AN2122/AN2126 only)

I2CMODE **I²C Mode** **7FA7**

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	STOPIE	0
R	R	R	R	R	R	R/W	R
0	0	0	0	0	0	0	0

Figure 9-11. I²C Mode Register

The I²C interrupt includes one additional interrupt source in the AN2122/AN2126, a 1-0 transition of the STOP bit. To enable this interrupt, set the STOPIE bit in the I2CMODE register. The 8051 determines the interrupt source by checking the DONE and STOP bits in the I2CS register.

I2CS **I²C Control and Status** **7FA5**

b7	b6	b5	b4	b3	b2	b1	b0
START	STOP	LASTRD	ID1	ID0	BERR	ACK	DONE
R/W	R/W	R/W	R	R	R	R	R
0	0	0	X	X	0	0	0

Figure 9-12. I²C Control and Status Register

I2DAT **I²C Data** **7FA6**

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 9-13. I²C Data

From Table 10-1, at power-on:

- Endpoint data buffers and byte counts are un-initialized (1,2).
- The 8051 is held in reset, and the CLK24 pin is enabled (3).
- All port pins are configured as input ports (4-6).
- USB interrupts are disabled, and USB interrupt requests are cleared (7-8).
- Bulk IN and OUT endpoints are unarmed, and their stall bits are cleared (9). The EZ-USB core will NAK IN or OUT tokens while the 8051 is reset. OUT endpoints are enabled when the 8051 is released from reset.
- Endpoint toggle bits are cleared (11).
- The ReNum bit is cleared. This means that the EZ-USB core, and not the 8051, initially responds to USB device requests (12).
- The USB function address register is set to zero (13).
- The endpoint valid bits are set to match the endpoints used by the default USB device (14-17).
- Endpoint pairing is disabled. Also, ISOSend0=0, meaning that if an Isochronous endpoint receives an IN token without being loaded by the 8051 in the previous frame, the EZ-USB core does not generate any response (18).
- The breakpoint condition is cleared, and autovectoring is turned off (19).
- Configuration Zero, Alternate Setting Zero is in effect (20-21).

10.3 Releasing the 8051 Reset

The EZ-USB register bit CPUCS.0 resets the 8051. This bit is HI at power-on, initially holding the 8051 in reset. There are three ways to release the 8051 from reset:

- By the host, as the final step of a RAM download.
- Automatically, as part of an EEPROM load.
- Automatically, when external ROM is used (EA=1).

OEA **Port A Output Enable** **7F9C**

b7	b6	b5	b4	b3	b2	b1	b0
OEA7	**OEA6**	**OEA5**	**OEA4**	**OEA3**	**OEA2**	**OEA1**	**OEA0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

OEB **Port B Output Enable** **7F9D**

b7	b6	b5	b4	b3	b2	b1	b0
OEB7	**OEB6**	**OEB5**	**OEB4**	**OEB3**	**OEB2**	**OEB1**	**OEB0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

OEC **Port C Output Enable** **7F9E**

b7	b6	b5	b4	b3	b2	b1	b0
OEC7	**OEC6**	**OEC5**	**OEC4**	**OEC3**	**OEC2**	**OEC1**	**OEC0**
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 12-9. Output Enable Registers

The OE registers control the output enables on the tri-state drivers connected to the port pins. When these bits are "1," the port is an output, unless the corresponding PORTnCFG bit is set to a "1."

If the CONTROL transfer uses an OUT data phase, the 8051 must load a dummy byte count into OUT0BC to arm the OUT endpoint buffer. Until it does, the EZ-USB core will NAK the OUT tokens.

Bit 2: INBSY *IN Endpoint Busy*

INBSY is a read-only bit that is automatically cleared when a SETUP token arrives. The 8051 sets the INBSY bit by writing a byte count to IN0BC.

If the CONTROL transfer uses an IN data phase, the 8051 loads the requested data into the IN0BUF buffer, and then loads the byte count into IN0BC to arm the data phase of the CONTROL transfer. Alternatively, the 8051 can arm the data transfer by loading an address into the Setup Data Pointer registers SUDPTRH/L. Until armed, the EZ-USB core will NAK the IN tokens.

Bit 1: HSNAK *Handshake NAK*

HSNAK (Handshake NAK) is a read/write bit that is automatically set when a SETUP token arrives. The 8051 clears HSNAK by writing a "1" to the register bit.

While HSNAK=1, the EZ-USB core NAKs the handshake (status) phase of the CONTROL transfer. When HSNAK=0, it ACKs the handshake phase. The 8051 can clear HSNAK at any time during a CONTROL transfer.

Bit 0: EP0STALL *Endpoint Zero Stall*

EP0STALL is a read/write bit that is automatically cleared when a SETUP token arrives. The 8051 sets EP0STALL by writing a "1" to the register bit.

While EP0STALL=1, the EZ-USB core sends the STALL PID for any IN or OUT token. This can occur in either the data or handshake phase of the CONTROL transfer.

Note

To indicate an endpoint stall on endpoint zero, set both EP0STALL and HSNAK bits. Setting the EP0STALL bit alone causes endpoint zero to NAK forever because the host keeps the control transfer pending.

12.14 Global USB Registers

SUDPTRH **Setup Data Pointer High** **7FD4**

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

SUDPTRL **Setup Data Pointer Low** **7FD5**

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

Figure 12-30. Setup Data Pointer High/Low Registers

When the EZ-USB chip receives a “Get_Descriptor” request on endpoint zero, it can instruct the EZ-USB core to handle the multi-packet IN transfer by loading these registers with the address of an internal table containing the descriptor data. The descriptor data tables may be placed in internal program/data RAM or in unused *Endpoint 0-7 RAM*. The SUDPTR does not operate with external memory. The SUDPTR registers should be loaded in HIGH/LOW order.

In addition to loading SUDPTRL, the 8051 must also clear the HSNAK bit in the EP0CS register (by writing a “1” to it) to complete the CONTROL transfer.

Note

Any host request that uses the EZ-USB Setup Data Pointer to transfer IN data must indicate the number of bytes to transfer in bytes 6 (wLenghthL) and 7 (wLengthH) of the SETUP packet. These bytes are pre-assigned in the USB Specification to be length bytes in all standard device requests such as “Get_Descriptor.” If vendor-specific requests are used to transfer large blocks of data using the Setup Data Pointer, they must include this pre-defined length field in bytes 6-7 to tell the EZ-USB core how many bytes to transfer using the Setup Data Pointer.

EZ-USB Registers

Addr	Name	Description	D7	D6	D5	D4	D3	D2	D1	D0			Notes
	Global USB Registers												
7FD4	SUDPTRH	Setup Data Ptr H	A15	A14	A13	A12	A11	A10	A9	A8			
7FD5	SUDPTRL	Setup Data Ptr L	A7	A6	A5	A4	A3	A2	A1	A0			
7FD6	USBCS	USB Control & Status	WakeSRC	*	*	*	DisCon	DiscOE	ReNum	SIGRSUME			Clear b7 by writing "1"
7FD7	TOGCTL	Toggle Control	Q	S	R	IO	0	EP2	EP1	EP0			
7FD8	USBFRAMEL	Frame Number L	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0			
7FD9	USBFRAMEH	Frame Number H	0	0	0	0	0	FC10	FC9	FC8			
7FDA		(reserved)											
7FDB	FNADDR	Function Address	0	FA6	FA5	FA4	FA3	FA2	FA1	FA0			
7FDC		(reserved)											
7FDD	USBPAIR	Endpoint Control	ISOsend0	*	PR6OUT	PR4OUT	PR2OUT	PR6IN	PR4IN	PR2IN			PRx = 1 to pair EP
7FDE	IN07VAL	Input Endpoint 0-7 valid	IN7VAL	IN6VAL	IN5VAL	IN4VAL	IN3VAL	IN2VAL	IN1VAL	1			VAL =1 means valid
7FDF	OUT07VAL	Output Endpoint 0-7 valid	OUT7VAL	OUT6VAL	OUT5VAL	OUT4VAL	OUT3VAL	OUT2VAL	OUT1VAL	1			VAL =1 means valid
7FE0	INISOVAL	Input EP 8-15 valid	IN15VAL	IN14VAL	IN13VAL	IN12VAL	IN11VAL	IN10VAL	IN9VAL	IN8VAL			VAL =1 means valid
7FE1	OUTISOVAL	Output EP 8-15 valid	OUT15VAL	OUT14VAL	OUT13VAL	OUT12VAL	OUT11VAL	OUT10VAL	OUT9VAL	OUT8VAL			VAL =1 means valid
7FE2	FASTXFR	Fast Transfer Mode	FISO	FBLK	RPOL	RMOD1	RMOD0	WPOL	WMOD1	WMOD0			
7FE3	AUTOPTRH	Auto-Pointer H	A15	A14	A13	A12	A11	A10	A9	A8			
7FE4	AUTOPTRL	Auto-Pointer L	A7	A6	A5	A4	A3	A2	A1	A0			
7FE5	AUTODATA	Auto Pointer Data	D7	D6	D5	D4	D3	D2	D1	D0			
7FE6		(reserved)											
7FE7		(reserved)											
	Setup Data												
7FE8	SETUPDAT	8 bytes of SETUP data	d7	d6	d5	d4	d3	d2	d1	d0			
	Isochronous FIFO Sizes												
7FF0	OUT8ADDR	Endpt 8 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF1	OUT9ADDR	Endpt 9 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF2	OUT10ADDR	Endpt 10 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF3	OUT11ADDR	Endpt 11 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF4	OUT12ADDR	Endpt 12 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF5	OUT13ADDR	Endpt 13 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF6	OUT14ADDR	Endpt 14 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF7	OUT15ADDR	Endpt 15 OUT Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF8	IN8ADDR	Endpt 8 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FF9	IN9ADDR	Endpt 9 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FFA	IN19ADDR	Endpt 10 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FFB	IN11ADDR	Endpt 11 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FFC	IN12ADDR	Endpt 12 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FFD	IN13ADDR	Endpt 13 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FFE	IN14ADDR	Endpt 14 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			
7FFF	IN15ADDR	Endpt 15 IN Start Addr	A9	A8	A7	A6	A5	A4	0	0			

Table B-2. 8051 Instruction Set

Mnemonic	Description	Byte	Instr. Cycles	Hex Code
JNB bit, rel	Jump on direct bit = 0	3	4	30
JBC bit, rel	Jump on direct bit = 1 and clear	3	4	10
JMP @ A+DPTR	Jump indirect relative DPTR	1	3	73
JZ rel	Jump on accumulator = 0	2	3	60
JNZ rel	Jump on accumulator /= 0	2	3	70
CJNE A, direct, rel	Compare A, direct JNE relative	3	4	B5
CJNE A, #d, rel	Compare A, immediate JNE relative	3	4	B4
CJNE Rn, #d, rel	Compare reg, immediate JNE relative	3	4	B8-BF
CJNE @ Ri, #d, rel	Compare Ind, immediate JNE relative	3	4	B6-B7
DJNZ Rn, rel	Decrement register, JNZ relative	2	3	D8-DF
DJNZ direct, rel	Decrement direct byte, JNZ relative	3	4	D5
Miscellaneous				
NOP	No operation	1	1	00
There is an additional reserved opcode (A5) that performs the same function as NOP. All mnemonics are copyrighted. Intel Corporation 1980.				

B.1.3 Instruction Timing

Instruction cycles in the 8051 core are 4 clock cycles in length, as opposed to the 12 clock cycles per instruction cycle in the standard 8051. This translates to a 3X improvement in execution time for most instructions.

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.

For example, in the standard 8051, the instructions `MOVX A, @DPTR` and `MOV direct, direct` each take 2 instruction cycles (24 clock cycles) to execute. In the 8051 core, `MOVX A, @DPTR` takes two instruction cycles (8 clock cycles) and `MOV direct, direct` takes three instruction cycles (12 clock cycles). Both instructions execute faster on the 8051 core than they do on the standard 8051, but require different numbers of clock cycles.

For timing of real-time events, use the numbers of instruction cycles from Table B-1. to calculate the timing of software loops. The bytes column indicates the number of memory