# Chapter 15. EZ-USB FX Registers

## 15.1 Introduction

This section describes the EZ-USB FX registers in the order they appear in the EZ-USB FX memory map. The registers are named according to the following conventions.

*Most registers deal with endpoints.* The general register format is **DDDnFFF**, where:

**DDD** is endpoint direction, IN or OUT with respect to the USB host.

**n** is the endpoint number, where:

- "07" refers to endpoints 0-7 as a group.
- 0-7 refers to each individual BULK/INTERRUPT/CONTROL endpoint.
- "ISO" indicates isochronous endpoints as a group.

**FFF** is the function, where:

- CS is a control and status register
- IRQ is an Interrupt Request Bit
- IE is an Interrupt Enable Bit
- BC, BCL, and BCH are byte count registers. BC is used for single byte counts, and BCL/H are used as the low and high bytes of 16-bit byte counts.
- DATA is a single-register access to a FIFO.
- BUF is the start address of a buffer.

### 15.1.1 Example Register Formats

- IN7BC is the Endpoint 7 IN byte count.

- OUT07IRQ is the register containing interrupt request bits for OUT endpoints 0-7.
- INISOVAL contains valid bits for the isochronous IN endpoints (EP8IN-EP15IN).

### 15.1.2 Other Conventions

**USB** Indicates a global (not endpoint-specific) USB function.

**ADDR** Is an address.

**VAL** Means valid.

**FRAME** Is a frame count.

**PTR** Is an address pointer.

| Register Name | Register Function | Address |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **bitname** | **bitname** | **bitname** | **bitname** | **bitname** | **bitname** | **bitname** | **bitname** |
| R, W access | R, W access | R, W access | R, W access | R, W access | R, W access | R, W access | R, W access |
| Default val | Default val | Default val | Default val | Default val | Default val | Default val | Default val |

*Figure 15-1. Register Description Format*

*Figure 15-1.* illustrates the register description format used in this chapter.

- The top line shows the register name, functional description, and address in the EZ-USB FX memory.
- The second line shows the bit position in the register.
- The third line shows the name of each bit in the register.
- The fourth line shows 8051 accessibility: R(ead), W(rite), or R/W.
- The fifth line shows the default value. These values apply after a Power-On-Reset (POR).

## 15.2 Slave FIFO Registers

### 15.2.1 FIFO A Read Data

| AINDATA | FIFO A Read Data | 7800 |
|---|---|---|

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

*Figure 15-2. FIFO A Read Data*

Each time the 8051 reads a byte from this register, the A-IN FIFO advances to the next byte in the FIFO and the AINBC (byte count) decrements. Reading this register when there is one byte remaining in the A-IN FIFO sets the A-IN FIFO Empty Flag (AINEF, in ABINCS.4), which causes an interrupt request on INT4 (Table 2). Reading this register when the A-IN FIFO is empty returns indeterminate data and has no effect on the FIFO flags byte counts. For more information, see Section 7.2.1. *"FIFO A Read Data"*.

### 15.2.2 A-IN FIFO Byte Count

| AINBC | A-IN FIFO Byte Count | 7801 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-3. A-IN FIFO Byte Count*

This count reflects the number of bytes remaining in the A-IN FIFO. Valid byte counts are 0-64. When non-zero, every byte written by outside logic increments this count, and every 8051 read of AINDATA decrements this count. If AINBC is zero, an 8051 read of AINDATA returns indeterminate data and results in the byte count in AINBC remaining at zero. Data bytes should never be written to the FIFO from outside logic when the AINFULL flag is HI. For more information, see Section 7.2.2. *"A-IN FIFO Byte Count"*.

### 15.2.3 A-IN FIFO Programmable Flag

**AINPF** **A-IN FIFO Programmable Flag** **7802**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| 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 15-4. A-IN FIFO Programmable Flag*

This register controls the sense and value for the internal A-IN FIFO programmable flag. This flag is testable by the 8051. For more information including a list of bit definitions for this register, see Section 7.2.3. *"A-IN FIFO Programmable Flag"*.

### 15.2.4 A-IN FIFO Pin Programmable Flag

**AINPFPIN** **A-IN FIFO Pin Programmable Flag** **7803**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| 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 15-5. A-IN FIFO Pin Programmable Flag*

This register controls the sense and value for the A-IN FIFO Programmable Flag that appears on the AINFLAG *pin*. This pin is used by external logic to regulate external writes to the A-IN FIFO. The AINPFPIN Register is programmed with the same data format as the previous register, AINPF. The only operational difference is that the flag drives a hardware pin rather than existing as an internal register bit. For more information, see Section 7.2.3.3. *"A-IN FIFO Pin Programmable Flag"*.

## 15.2.5 B-IN FIFO Read Data

| BINDATA | B-IN FIFO Read Data | 7805 |
|---|---|---|

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

*Figure 15-6. B-IN FIFO Read Data*

Each time the 8051 reads a byte from this register, the B-IN FIFO advances to the next byte in the FIFO and the BINBC (byte count) decrements. Reading this register when there is one byte remaining in the FIFO sets the B-IN FIFO Empty Flag (BINEF, in ABINCS.1), which causes an INT4 Request. Reading this register when the B-IN FIFO is empty returns indeterminate data and has no effect on the FIFO flags or byte count. For more information, see Section 7.2.4. *"B-IN FIFO Read Data"*.

## 15.2.6 B-IN FIFO Byte Count

| BINBC | B-IN FIFO Byte Count | 7806 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-7. B-IN FIFO Byte Count*

This count reflects the number of bytes remaining in the B-IN FIFO. Valid byte counts are 0-64. When non-zero, every byte written by outside logic increments this count, and every 8051 read of BINDATA decrements this count. If BINBC is zero, an 8051 read of BINDATA returns indeterminate data and results in the byte count in BINBC remaining at zero. Data bytes should never be written to the FIFO from outside logic when the BINFULL flag is HI. For more information, see Section 7.2.5. *"B-IN FIFO Byte Count."*

### 15.2.7 B-IN FIFO Programmable Flag

**BINPF** **B-IN FIFO Programmable Flag** **7807**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| 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 15-8. B-IN FIFO Programmable Flag*

This register controls the sense and value for the internal B-IN FIFO programmable flag. For more information including a list of bit descriptions, see Section 7.2.6. *"B-IN FIFO Programmable Flag."*

### 15.2.8 B-IN FIFO Pin Programmable Flag

**BINPFPIN** **B-IN FIFO Pin Programmable Flag** **7808**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| 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 15-9. B-IN FIFO Pin Programmable Flag*

This register controls the sense and value for the B-IN FIFO Programmable Flag that appears on the BINFLAG *pin*. This pin is used by external logic to regulate external writes to the B-IN FIFO. The BINPFPIN Register is programmed with the same data format as the previous register, BINPF. The only operational difference is that the flag drives a hardware pin rather than existing as an internal register bit. For more information see Section 7.2.7. *"B-IN FIFO Pin Programmable Flag."*

## 15.2.9 Input FIFOs A/B Toggle CTL and Flags

| ABINTCS | Input FIFOs A/B Toggle CTL and Flags | 780A |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **INTOG** | **INSEL** | **AINPF** | **AINEF** | **AINFF** | **BINPF** | **BINEF** | **BINFF** |
| R/W | R/W | R | R | R | R | R | R |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

*Figure 15-10. Input FIFOs A/B Toggle CTL and Flags*

For information about this register, including a list of bit descriptions, see Section 7.2.8. *"Input FIFOs A/B Toggle CTL and Flags."*

## 15.2.10 Input FIFOs A/B Interrupt Enables

| ABINIE | Input FIFOs A/B Interrupt Enables | 780B |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **AINPFIE** | **AINEFIE** | **AINFFIE** | **BINPFIE** | **BINEFIE** | **BINFFIE** |
| 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 15-11. Input FIFOs A/B Interrupt Enables*

For information about this register, including a list of bit descriptions, see Section 7.2.9. *"Input FIFOs A/B Interrupt Enables."*

## 15.2.11 Input FIFOs A/B Interrupt Requests

| ABINIRQ | Input FIFOs A/B Interrupt Enables | 780C |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **AINPFIR** | **AINEFIR** | **AINFFIR** | **BINPFIR** | **BINEFIR** | **BINFFIR** |
| 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 15-12. Input FIFOs A/B Interrupt Requests*

For information about the ABINIRQ Register, including a list of bit descriptions, see Section 7.2.10. *"Input FIFOs A/B Interrupt Requests."*

### 15.2.12 FIFO A Write Data

| AOUTDATA | FIFO A Write Data | 780E |
|---|---|---|

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

*Figure 15-13. FIFO A Write Data*

A-OUT FIFO Write Data. Each time the 8051/DMA writes a byte to this register, the A-OUT FIFO advances to the next open position in the FIFO and the AOUTBC (byte count) increments. Writing this register when there are 63 bytes remaining in the A-OUT FIFO sets the A-FIFO Full Flag (AOUTFF, in ABOUTCS.3), which causes an INT4 Request. Writing this register when the A-OUT FIFO is full (64 bytes) does not update the FIFO or byte count, and has no effect on the FIFO flags or byte count. For more information, see Section 7.2.11. *"FIFO A Write Data."*

### 15.2.13 A-OUT FIFO Byte Count

| AOUTBC | A-OUT FIFO Byte Count | 780F |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-14. Input FIFOs A/B Interrupt Requests*

This count reflects the number of bytes remaining in the A-OUT FIFO. Valid byte counts are 0-64. When non-zero, every byte read by outside logic decrements this count, and every 8051 write of AOUTDATA increments this count. If AOUTBC is zero, reading a data byte by outside logic returns indeterminate data and results in the byte count in AOUTBC remaining at zero. For more information, see Section 7.2.11.1. *"A-OUT FIFO Byte Count."*

## 15.2.14 A-OUT FIFO Programmable Flag

| AOUTPF | A-OUT FIFO Programmable Flag | 7810 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-15. Input FIFOs A/B Interrupt Requests*

This register controls the sense and value for the internal A-OUT FIFO Programmable Flag. The internal flag may be tested by the 8051, and/or enabled to cause an INT4 Interrupt Request. The 8051 tests the internal FIFO programmable flag by reading the AOUTPF Bit in ABOUTCS.5 (register at 0x7818). For more information including a list of bit descriptions, see Section 7.2.12. *"A-OUT FIFO Programmable Flag."*

## 15.2.15 A-OUT FIFO Pin Programmable Flag

| AOUTPFPIN | A-OUT FIFO Pin Programmable Flag | 7811 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-16. A-OUT FIFO Pin Programmable Flag*

This register controls the sense and value for the A-OUT FIFO Programmable Flag that appears on the AOUTFLAG *pin*. This pin is used by external logic to regulate external reads from the A-OUT FIFO. The AOUTPFPIN Register is programmed with the same data format as the previous register, AOUTPF. The only operational difference is that the flag drives a hardware pin rather than existing as an internal register bit. For more information, see Section 7.2.13. *"A-OUT FIFO Pin Programmable Flag."*

### 15.2.16 B-OUT FIFO Write Data

**BOUTDATA** **B-OUT FIFO Write Data** **7813**

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

*Figure 15-17. B-OUT FIFO Write Data*

Each time the 8051/DMA writes a byte to this register, the B-OUT FIFO advances to the next open position in the FIFO and the BOUTBC (Byte count) increments. Writing this register when there are 63 bytes remaining in the B-OUT FIFO sets the B-FIFO Full Flag (BOUTFF, in ABOUTCS.0), which causes an INT4 Interrupt Request. Writing this register when the B-OUT FIFO is full (64 bytes) does not update the FIFO or byte count, and has no effect on the FIFO flags or byte count. For more information, see Section 7.2.14. *"B-OUT FIFO Write Data."*

### 15.2.17 B-OUT FIFO Byte Count

**BOUTBC** **B-OUT FIFO Byte Count** **7814**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-18. B-OUT FIFO Byte Count*

This count reflects the number of bytes remaining in the B-OUT FIFO. Valid byte counts are 0-64. When non-zero, every byte read by outside logic decrements this count, and every 8051 write of BOUTDATA increments this count. If BOUTBC is zero, reading a data byte by outside logic returns indeterminate data and results in the byte count in BOUTBC remaining at zero. For more information, see Section 7.2.15. *"B-OUT FIFO Byte Count."*

### 15.2.18 B-OUT FIFO Programmable Flag

**BOUTPF** **B-OUT FIFO Programmable Flag** **7815**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-19. B-OUT FIFO Programmable Flag*

This register controls the sense and value for the internal B-OUT FIFO Programmable Flag. The internal flag may be tested by the 8051, and/or enabled to cause an INT4 Interrupt Request. For more information including a list of bit descriptions, see Section 7.2.16. *"B-OUT FIFO Programmable Flag."*

### 15.2.19 B-OUT FIFO Pin Programmable Flag

**BOUTPFPIN** **B-OUT FIFO Pin Programmable Flag** **7816**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **LTGT** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| 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 15-20. B-OUTFIFO Pin Programmable Flag*

This register controls the sense and value for the B-OUT FIFO Programmable Flag that appears on the BOUTFLAG *pin*. This pin is used by external logic to regulate external reads from the B-OUT FIFO. The BOUTPFPIN Register is programmed with the same data format as the previous register, BOUTPF. The only operational difference is that the flag drives a hardware pin rather than existing as an internal register bit. For more information including a list of bit descriptions, see Section 7.2.17. *"B-OUT FIFO Pin Programmable Flag."*

## 15.2.20 Output FIFOs A/B Toggle CTL and Flags

**ABOUTCS** **Output FIFOs A/B Toggle CTL and Flags** **7818**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUTINTOG** | **OUTSEL** | **AOUTPF** | **AOUTEF** | **AOUTFF** | **BOUTPF** | **BOUTEF** | **BOUTFF** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

*Figure 15-21. Output FIFOs A/B Toggle CTL and Flags*

For information about this register, including a list of bit descriptions, see Section 7.2.18. *"Output FIFOs A/B Toggle CTL and Flags."*

## 15.2.21 Output FIFOs A/B Interrupt Enables

**ABOUTIE** **Output FIFOs A/B Interrupt Enables** **7819**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **AOUTPFIE** | **AOUTEFIE** | **AOUTFFIE** | **BOUTPFIE** | **BOUTEFIE** | **BOUTFFIE** |
| 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 15-22. Output FIFOs A/B Interrupt Enables*

For information about this register, including a list of bit descriptions, see Section 7.2.19. *"Output FIFOs A/B Interrupt Enables."*

## 15.2.22 Output FIFOs A/B Interrupt Requests

**ABOUTIRQ** **Output FIFOs A/B Interrupt Requests** **781A**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **AOUTPFIR** | **AOUTEFIR** | **AOUTFFIR** | **BOUTPFIR** | **BOUTEFIR** | **BOUTFFIR** |
| 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 15-23. Output FIFOs A/B Interrupt Requests*

For information about this register, including a list of bit descriptions, see Section 7.2.20. *"Output FIFOs A/B Interrupt Requests."*

### 15.2.23 FIFO A/B Setup

**ABSETUP** **FIFO A/B Setup** **781C**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **ASYNC** | **DBLIN** | **0** | **OUTDLY** | **0** | **DBLOUT** |
| 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 15-24. FIFO A/B Setup*

For information about this register, including a list of bit descriptions, see Section 7.2.21. *"FIFO A/B Setup."*

### 15.2.24 FIFO A/B Control Signal Polarities

**ABPOLAR** **FIFO A/B Control Signal Polarities** **781D**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **BOE** | **AOE** | **SLRD** | **SLWR** | **ASEL** | **BSEL** |
| 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 15-25. FIFO A/B Control Signal Polarities*

These bits define the pin polarities for the indicated signals. The 8051 sets a bit LOW for active low, and HI for active high. For more information including a list of bit descriptions, see Section 7.2.22. *"FIFO A/B Control Signal Polarities."*

### 15.2.25 FIFO Flag Reset

| ABFLUSH | Reset All FIFO Flags | 781E |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |
| W | W | W | W | W | W | W | W |
| x | x | x | x | x | x | x | x |

*Figure 15-26. FIFO Flag Reset*

The 8051 writes any value to this register to reset the FIFO byte counts to zero, effectively *flushing* the FIFOs. Consequently, the byte counts are set to zero, the empty flags are set, and the full flags are cleared. Reading this register returns indeterminate data. For more information including a list of bit descriptions, see Section 7.2.23. *"FIFO Flag Reset."*

## 15.3 Waveform Selector

| WFSELECT | Waveform Selector | 7824 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **SINGLEWR0-3** | | **SINGLERD0-3** | | **FIFOWR0-3** | | **FIFORD0-3** | |
| W | W | R | R | W | W | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-27. Waveform Selector*

For detailed information, see *Chapter 8. "General Programmable Interface (GPIF)"*.

## 15.4 GPIF Done, GPIF IDLE Drive Mode

**IDLECS** **GPIF Done, GPIF IDLE Drive Mode** **7825**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **DONE** | **0** | **0** | **0** | **0** | **0** | **0** | **IDLEDRV** |
| 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 15-28. GPIF Done, GPIF IDLE Drive Mode*

For detailed information, see *Chapter 8. "General Programmable Interface (GPIF)"*.

## 15.5 Inactive Bus, CTL States

**IDLECTLOUT** **Inactive Bus, CTL States** **7826**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **IOE3** | **IOE2** | **IOE1/CTL5** | **IOE0/CTL4** | **CTL3** | **CTL2** | **CTL1** | **CTL0** |
| 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 15-29. Inactive Bus, CTL States*

**CTLOUTCFG** **CTLOUT Pin Drive** **7827**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **TRICTL** | **0** | **CTL5** | **CTL4** | **CTL3** | **CTL2** | **CTL1** | **CTL0** |
| 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 15-30. CTLOUT Pin Drive*

## 15.6 GPIF Address LSB

| GPIFADRL | GPIF Address Low | 782A |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **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 15-31. GPIF Address Low*

## 15.7 FIFO A IN Transaction Count

| AINTC | FIFO A IN Transaction Count | 782C |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **FITC** | **AINTC6** | **AINTC5** | **AINTC4** | **AINTC3** | **AINTC2** | **AINTC1** | **AINTC0** |
| 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 15-32. FIFO A IN Transaction Count*

For detailed information, see *Chapter 8. "General Programmable Interface (GPIF)"*.

## 15.8 FIFO A OUT Transaction Count

| AOUTTC | FIFO A OUT Transaction Count | 782D |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **FITC** | **AOUTTC6** | **AOUTTC5** | **AOUTTC4** | **AOUTTC3** | **AOUTTC2** | **AOUTTC1** | **AOUTTC0** |
| 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 15-33. FIFO A OUT Transaction Count*

See *Chapter 8. "General Programmable Interface (GPIF)"* for detailed information.

## 15.9 FIFO A Transaction Trigger

| ATRIG | FIFO A Transaction Trigger | 782E |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |
| 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 15-34. FIFO A Transaction Trigger*

See *Chapter 8. "General Programmable Interface (GPIF)"* for detailed information.

## 15.10 FIFO B IN Transaction Count

**BINTC** **FIFO B IN Transaction Count** **7830**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **BINTC7** | **BINTC6** | **BINTC5** | **BINTC4** | **BINTC3** | **BINTC2** | **BINTC1** | **BINTC0** |
| 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 15-35. FIFO B IN Transaction Count*

For detailed information, see *Chapter 8. "General Programmable Interface (GPIF)"*.

## 15.11 FIFO B OUT Transaction Count

**BOUTTC** **FIFO B OUT Transaction Count** **7831**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **BOUTTC7** | **BOUTTC6** | **BOUTTC5** | **BOUTTC4** | **BOUTTC3** | **BOUTTC2** | **BOUTTC1** | **BOUTTC0** |
| 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 15-36. FIFO B OUT Transaction Count*

For detailed information, see *Chapter 8. "General Programmable Interface (GPIF)"*.

## 15.12 FIFO B Transaction Trigger

| BTRIG | FIFO B Transaction Trigger | 7832 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |
| 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 15-37. FIFO B Transaction*

For detailed information, see *Chapter 8. "General Programmable Interface (GPIF)"*.

## 15.13 GPIF Data H (16-bit mode only)

| SGLDATH | GPIF Data H (16-bit mode only) | 7834 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **D15** | **D14** | **D13** | **D12** | **D11** | **D10** | **D9** | **D8** |
| 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 15-38. GPIF Data H (16-bit mode only)*

## 15.14 Read or Write GPIF Data L and Trigger Read Transaction

| SGLDATLTRIG | R/W GPIF DataL/Trig Rd Transaction | 7835 |
|---|---|---|

| 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 15-39. Read or Write GPIF DataL and Trigger Read Transaction*

## 15.15 Read GPIF Data L, No Read Transaction Trigger

**SGLDATLNTRIG** **Rd GPIF Data L/No Trig Rd Transaction** **7836**

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

*Figure 15-40. Read GPIF Data L, No Read Transaction Trigger*

## 15.16 Internal READY, Sync/Async, READY Pin States

**READY** **Internal Rdy, Sync/Async, Rdy Pin States** **7838**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **INTRDY** | **SAS** | **RDY5** | **RDY4** | **RDY3** | **RDY2** | **RDY1** | **RDY0** |
| R/W | R/W | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-41. Internal READY, Sync/Async, READY Pin States*

## 15.17 Abort GPIF Cycles

**ABORT** **Abort GPIF Cycles** **7839**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |
| W | W | W | W | W | W | W | W |
| x | x | x | x | x | x | x | x |

*Figure 15-42. Abort GPIF Cycles*

## 15.18 General Purpose I/F Interrupt Enable

| GENIE | General Purpose I/F Interrupt Enable | 783B |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **DMADONE** | **GPIFWF** | **GPIFDONE** |
| W | W | W | W | W | W | W | W |
| x | x | x | x | x | x | x | x |

*Figure 15-43. Generic Interrupt Enable*

## 15.19 Generic Interrupt Request

| GENIRQ | Generic Interrupt Request | 783C |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **DMADONE** | **GPIFWF** | **GPIFDONE** |
| 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 15-44. Generic Interrupt Request*

## 15.20 Input/Output Port Registers D and E

For more information, see Section 4.3. *"Input/Output Port Registers"*.

### 15.20.1 Port D Outputs

**OUTD** **Port D Outputs** **7841**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUTD7** | **OUTD6** | **OUTD5** | **OUTD4** | **OUTD3** | **OUTD2** | **OUTD1** | **OUTD0** |
| 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 15-45. Port D Outputs*

### 15.20.2 Input Port D Pins

**PINSD** **Port D Pins** **7842**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **PIND7** | **PIND6** | **PIND5** | **PIND4** | **PIND3** | **PIND2** | **PIND1** | **PIND0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-46. Input Port D Pins*

### 15.20.3 Port D Output Enable

**OED** **Port D Output Enable** **7843**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OED7** | **OED6** | **OED5** | **OED4** | **OED3** | **OED2** | **OED1** | **OED0** |
| 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 15-47. Port D Output Enable Register*

### 15.20.4 Port E Outputs

| OUTE | Port E Outputs | 7845 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUTE7** | **OUTE6** | **OUTE5** | **OUTE4** | **OUTE3** | **OUTE2** | **OUTE1** | **OUTE0** |
| 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 15-48. Port E Outputs*

### 15.20.5 Input Port E Pins

| PINSE | Port E Pins | 7846 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **PINE7** | **PINE6** | **PINE5** | **PINE4** | **PINE3** | **PINE2** | **PINE1** | **PINE0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-49. Input Port E Pins*

### 15.20.6 Port E Output Enable

| OEE | Port E Output Enable | 7847 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OEE7** | **OEE6** | **OEE5** | **OEE4** | **OEE3** | **OEE2** | **OEE1** | **OEE0** |
| 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 15-50. Port E Output Enable Register*

## 15.21 PORTSETUP

**PORTSETUP** **Timer0 Clock Source, Port to SFR Mapping** **7849**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **0** | **TOCLK** | **SFRPORT** |
| 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 15-51. PORTSETUP*

## 15.22 Interface Configuration

**IFCONFIG** **Interface Configuration** **784A**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **52ONE** | **0** | **0** | **0** | **GSTATE** | **BUS16** | **IF1** | **IF0** |
| R/W | R | R | R | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-52. Interface Configuration*

**Bit 7:** **52ONE** *Set to "1" for the 52-pin package*

This bit must be set to "1" for the 52-pin versions of EZ-USB FX. This ensures that certain signals that are driven properly for EZ-USB FX low power operation.

**Bit 6-4:** **Reserved** *Reserved bits, read as 0*

**Bit 3:** **GSTATE** *Output GSTATE*

When GSTATE=1, three bits in Port A take on the signals shown in Table 15-1. The GSTATE bits, which indicate GPIF states, are used for diagnostic purposes.

**Table 15-1. Port A Alternate Functions When GSTATE=1.**

| IO Pin | Alternate Function |
|---|---|
| PA0 | GSTATE[0] |
| PA1 | GSTATE[1] |
| PA2 | GSTATE[2] |

**Bit 2: BUS16** *8- or 16-Bit Slave FIFO Operation*

This bit selects 8-bit (BUS16=0) or 16-bit (BUS16=1) operation for slave FIFOs A and B. See *Chapter 7. "EZ-USB FX Slave FIFOs"* for full details.

**Bit 1-0: Interface Select** *Reconfigure I/O ports*

These bits, along with the BUS16 bit, select different groups of signals for various EZ-USB FX pins. Table 15-2 shows the selections.

**Table 15-2. Pin Configurations Based on IFCONFIG[1..0]**

| IFCONFIG[1..0] | | | | | |
|---|---|---|---|---|---|
| **00** | **01** | **10** | | **11** | |
| | | **BUS16=1** | **BUS16=0** | **BUS16=1** | **BUS16-=0** |
| PE0 | PE0 | adr0 | adr0 | BOUTFLAG | BOUTFLAG |
| PE1 | PE1 | adr1 | adr1 | AINFULL | AINFULL |
| PE2 | PE2 | adr2 | adr2 | BINFULL | BINFULL |
| PE3 | PE3 | adr3 | adr3 | AOUTEMTY | AOUTEMTY |
| PE4 | PE4 | adr4 | adr4 | BOUTEMTY | BOUTEMTY |
| PE5 | PE5 | CTL3 | CTL3 | PE5 | PE5 |
| PE6 | PE6 | CTL4 | CTL4 | PE6 | PE6 |
| PE7 | PE7 | CTL5 | CTL5 | PE7 | PE7 |
| NC | NC | CTL0 | CTL0 | AINFLAG | AINFLAG |
| NC | NC | CTL1 | CTL1 | BINFLAG | BINFLAG |
| NC | NC | CTL2 | CTL2 | AOUTFLAG | AOUTFLAG |
| Strap | Strap | RDY0 | RDY0 | ASEL | ASEL |
| Strap | Strap | RDY1 | RDY1 | BSEL | BSEL |
| Strap | Strap | RDY2 | RDY2 | AOE | AOE |
| Strap | Strap | RDY3 | RDY3 | BOE | BOE |
| Strap | Strap | RDY4 | RDY4 | SLWR | SLWR |
| Strap | Strap | RDY5 | RDY5 | SLRD | SLRD |
| Strap | Strap | adr5 | adr5 | X | X |
| Strap | Strap | XCLK | XCLK | XCLK | XCLK |
| **PORTB** | **D[7..0]** | **GDA[7..0]** | **GDA7..0]** | **AFI[7..0]** | **AFI[7..0]** |
| **PORTD** | **PORTD** | **GDB[7..0]** | **PORTD** | **BFI[7..0]** | **PORTD** |

NC -Package pin must be left unconnected.
Strap -Package pin must be either pulled-up to $V_{DD}$ or pulled-down to GND.

## 15.23 PORTA and PORTC Alternate Configurations

### 15.23.1 Port A Alternate Configuration #2

**PORTACF2** **PORTA Alternate Configuration #2** **784B**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **SLRD** | **SLWR** | **0** | **0** | **0** | **0** |
| 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 15-53. Port A Alternate Configuration #2*

**Bit 5: SLRD** *Select SLRD/RDY5 signal on PA5 pin*

This bit, in conjunction with the PORTACFG.5 Bit and the IFCONFIG[1..0] bits, determines the function of PA5, as shown in Table 15-3.

**Table 15-3. Port A Bit 5**

| **PORTA Bit 5** | | | |
|---|---|---|---|
| PORT-ACFG.5=0 | PORTACFG.5=1 | | |
| | PORTACF2.5=0 | PORTACF2.5=1 | |
| | | IFCONFIG[1..0]=10 | IFCONFIG[1..0]=11 |
| **Port pin PA5** | **FRD#** | **RDY5** | **SLRD** |

**Bit 4: SLWR** *Select SLWR/RDY4 signal on PA4 pin*

This bit, in conjunction with the PORTACFG.4 Bit and the IFCONFIG[1..0] bits, determines the function of PA4, as shown in Table 15-4.

**Table 15-4. Port A Bit 4**

| **PORTA Bit 4** | | | |
|---|---|---|---|
| PORT-ACFG.4=0 | PORTACFG.4=1 | | |
| | PORTACF2.4=0 | PORTACF2.4=1 | |
| | | IFCONFIG[1..0]=10 | IFCONFIG[1..0]=11 |
| **Port pin PA4** | **FWR#** | **RDY4** | **SLWR** |

### 15.23.2 Port C Alternate Configuration #2

**PORTCCF2** **PORTC Alternate Configuration #2** **784C**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **CTL5** | **CTL4** | **CTL3** | **CTL1** | **RDY3** | **0** | **RDY1** | **RDY0** |
| 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 15-54. Port C Alternate Configuration #2*

**Bit 7: CTL5** *Select CTL5 on PC7 pin*

This bit, in conjunction with the PORTCCFG.7 Bit, determines the function of PC7, as shown in Table 15-5.

**Table 15-5. Port C Bit 7**

| **PORTC Bit 7** | | | |
|---|---|---|---|
| PORTCCFG.7=0 | PORTCCFG.7=1 | | |
| | PORTCCF2.7=0 | PORTCCF2.7=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC7** | **RD#** | **CTL5** | **X** |

**Bit 6: CTL4** *Select CTL4 on PC6 pin*

This bit, in conjunction with the PORTCCFG.6 Bit, determines the function of PC6, as shown in Table 15-6.

**Table 15-6. Port C Bit 6**

| **PORTC Bit 6** | | | |
|---|---|---|---|
| PORTCCFG.6=0 | PORTCCFG.6=1 | | |
| | PORTCCF2.6=0 | PORTCCF2.6=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC6** | **WR#** | **CTL4** | **X** |

**Bit 5:** **CTL3** *Select CTL3 on PC5 pin*

This bit, in conjunction with the PORTCCFG.5 Bit, determines the function of PC5, as shown in Table 15-7.

**Table 15-7. Port C Bit 5**

| PORTC Bit 5 | | | |
|---|---|---|---|
| PORTCCFG.5=0 | PORTCCFG.5=1 | | |
| | PORTCCF2.5=0 | PORTCCF2.5=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC5** | **T1** | **CTL3** | **X** |

**Bit 4:** **CTL1** *Select CTL1 on PC4 pin*

This bit, in conjunction with the PORTCCFG.4 Bit, determines the function of PC4, as shown in Table 15-8.

**Table 15-8. Port C Bit 4**

| PORTC Bit 4 | | | |
|---|---|---|---|
| PORTCCFG.4=0 | PORTCCFG.4=1 | | |
| | PORTCCF2.4=0 | PORTCCF2.4=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC4** | **T0** | **CTL1** | **X** |

**Bit 3:** **RDY3** *Select RDY3 on PC3 pin*

This bit, in conjunction with the PORTCCFG.3 Bit, determines the function of PC3, as shown in Table 15-9.

**Table 15-9. Port C Bit 3**

| PORTC Bit 3 | | | |
|---|---|---|---|
| PORTCCFG.3=0 | PORTCCFG.3=1 | | |
| | PORTCCF2.3=0 | PORTCCF2.3=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC3** | **INT1** | **RDY3** | **X** |

**Bit 2:** **Reserved** *Reads as 0*

**Bit 1:** **RDY1** *Select RDY1 on PC1 pin*

This bit, in conjunction with the PORTCCFG.1 Bit, determines the function of PC1, as shown in Table 15-10.

**Table 15-10. Port C Bit 1**

| PORTC Bit 1 | | | |
|---|---|---|---|
| PORTCCFG.1=0 | PORTCCFG.1=1 | | |
| | PORTCCF2.1=0 | PORTCCF2.1=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC1** | **TxD0** | **RDY1** | **X** |

**Bit 0:** **CTL5** *Select CTL5 on PC0 pin*

This bit, in conjunction with the PORTCCFG.0 Bit, determines the function of PC0, as shown in Table 15-11.

**Table 15-11. Port C Bit 0**

| PORTC Bit 0 | | | |
|---|---|---|---|
| PORTCCFG.0=0 | PORTCCFG.0=1 | | |
| | PORTCCF2.0=0 | PORTCCF2.0=1 | |
| | | IFCONFIG[1..0]=10 | 00, 01, 11 not valid |
| **Port pin PC0** | **RxD0** | **RDY0** | **X** |

## 15.24 DMA Registers

For more information on these DMA registers, see *Section 11.2. "DMA Register Descriptions"*.

### 15.24.1 Source, Destination, Transfer Length Address Registers

**DMASRCH** **DMA Source Address (H)** **784F**

| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-55. Upper Byte of the DMA Source Address*

**DMASRCL** **DMA Source Address (L)** **7850**

| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-56. Lower Byte of the DMA Source Address*

**DMADESTH** **DMA Destination Address (H)** **7851**

| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-57. Upper Byte of the DMA Destination Address*

| DMADESTL | DMA Destination Address (L) | 7852 |
|---|---|---|

| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-58. Lower Byte of the DMA Destination Address*

| DMALEN | DMA Transfer Length | 7854 |
|---|---|---|

| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-59. DMA Transfer Length (0=256 Bytes, 1=1 Byte, ... 255=255 Bytes)*

### *15.24.2 DMA Start and Status Register*

For further information on DMA Registers, see *Section 11.2. "DMA Register Descriptions".*

| DMAGO | DMA Start and Status | 7855 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **DONE** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| R/W | x | x | x | x | x | x | x |
| 0 | x | x | x | x | x | x | x |

*Figure 15-60. DMA Start and Status Register*

### 15.24.3 DMA Synchronous Burst Enables Register

| DMABURST | Synchronous Burst Enables | 7857 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **DSTR2** | **DSTR1** | **DSTR0** | **BR** | **BW** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

*Figure 15-61. Synchronous Burst Enables*

### 15.24.4 Select 8051 A/D busses as External FIFO

| DMAEXTFIFO | Use A/D Buses as External FIFO | 7858 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **n/a** | **n/a** | **n/a** | **n/a** | **n/a** | **n/a** | **n/a** | **n/a** |
| n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |

*Figure 15-62. Dummy Register*

## 15.25 Slave FIFO Interrupt (INT4)

The EZ-USB FX slave FIFOs contain various flags to alert the 8051 when a FIFO needs attention. These flags are encoded into the INT4 Autovector, which the 8051 can read in the INT4IVEC Register. The encoded values for each INT4 source are shown in Table 12-4. For more information including bit descriptions, see Section 12.15. *"Slave FIFO Interrupt (INT4)"*.

### 15.25.1 Interrupt 4 Autovector

**INT4IVEC** **Interrupt 4 Autovector** **785D**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **I4V3** | **I4V2** | **I4V1** | **I4V0** | **0** | **0** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-63. Interrupt 4 Autovector*

### 15.25.2 Interrupt 4 Autovector

**INT4SETUP** **Interrupt 4 Setup** **785E**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **INT4FC** | **INTRNL** | **AV4EN** |
| R | R | R | R | R | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-64. Interrupt 4 Setup*

## 15.26 Waveform Descriptors

| WFDESC | Waveform Descriptors | 7900 |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **x** | **x** | **x** | **x** | **x** | **x** | **x** | **x** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-65. Waveform Descriptors*

For detailed information, see *Section 8.1. "What is GPIF?"*.

## 15.27 Bulk Data Buffers

| INnBUF,OUTnBUF | Endpoint 0-7 IN/OUT Data Buffers | 7B40-7F3F* |
|---|---|---|

| 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 |

* See Table 15-12 for individual endpoint buffer addresses.

*Figure 15-66. Bulk Data Buffers*

**Table 15-12. Bulk Endpoint Buffer Memory Addresses**

| Address | Address | Name | Size |
|---|---|---|---|
| 1F00-1F3F | 7F00-7F3F | IN0BUF | 64 |
| 1EC0-1EFF | 7EC0-7EFF | OUT0BUF | 64 |
| 1E80-1EBF | 7E80-7EBF | IN1BUF | 64 |
| 1E40-1E7F | 7E40-7E7F | OUT1BUF | 64 |
| 1E00-1E3F | 7E00-7E3F | IN2BUF | 64 |
| 1DC0-1DFF | 7DC0-7DFF | OUT2BUF | 64 |
| 1D80-1DBF | 7D80-7DBF | IN3BUF | 64 |
| 1D40-1D7F | 7D40-7D7F | OUT3BUF | 64 |
| 1D00-1D3F | 7D00-7D3F | IN4BUF | 64 |
| 1CC0-1CFF | 7CC0-7CFF | OUT4BUF | 64 |
| 1C80-1CBF | 7C80-7CBF | IN5BUF | 64 |
| 1C40-1C7F | 7C40-7C7F | OUT5BUF | 64 |
| 1C00-1C3F | 7C00-7C3F | IN6BUF | 64 |
| 1BC0-1BFF | 7BC0-7BFF | OUT6BUF | 64 |
| 1B80-1BBF | 7B80-7BBF | IN7BUF | 64 |
| 1B40-1B7F | 7B40-7B7F | OUT7BUF | 64 |

Sixteen 64-byte bulk data buffers appear at 0x1B40 **and** 0x7B40 in the 8K version of EZ-USB FX. The endpoints are ordered to permit the reuse of the buffer space as contiguous RAM when the higher numbered endpoints are not used. These registers default to unknown states.

## 15.28 Isochronous Data FIFOs

| OUTnDATA | EP8OUT-EP15OUT FIFO Registers | 7F60-7F67* |
|---|---|---|

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

| INnDATA | EP8IN-EP15IN FIFO Registers | 7F68-7F6F* |
|---|---|---|

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

* See Table 15-13 for individual endpoint buffer addresses.

*Figure 15-67. Isochronous Data FIFOs*

**Table 15-13. Isochronous Endpoint FIFO Register Addresses**

| Address | Isochronous Data | Name |
|---|---|---|
| 7F60 | Endpoint 8 OUT Data | OUT8DATA |
| 7F61 | Endpoint 9 OUT Data | OUT9DATA |
| 7F62 | Endpoint 10 OUT Data | OUT10DATA |
| 7F63 | Endpoint 11 OUT Data | OUT11DATA |
| 7F64 | Endpoint 12 OUT Data | OUT12DATA |
| 7F65 | Endpoint 13 OUT Data | OUT13DATA |
| 7F66 | Endpoint 14 OUT Data | OUT14DATA |
| 7F67 | Endpoint 15 OUT Data | OUT15DATA |
| 7F68 | Endpoint 8 IN Data | IN8DATA |
| 7F69 | Endpoint 9 IN Data | IN9DATA |
| 7F6A | Endpoint 10 IN Data | IN10DATA |
| 7F6B | Endpoint 11 IN Data | IN11DATA |
| 7F6C | Endpoint 12 IN Data | IN12DATA |
| 7F6D | Endpoint 13 IN Data | IN13DATA |
| 7F6E | Endpoint 14 IN Data | IN14DATA |
| 7F6F | Endpoint 15 IN Data | IN15DATA |

Sixteen addressable data registers hold data from the eight isochronous IN endpoints and the eight isochronous OUT endpoints. Reading a data register reads a FIFO byte (USB OUT data); writing a Data Register loads a FIFO byte (USB IN data).

## 15.29 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 15-14 for individual endpoint buffer addresses.

*Figure 15-68. Isochronous Byte Counts*

**Table 15-14. 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 |

The USB core uses the byte count registers to report isochronous data payload sizes for OUT data transferred from the host to the USB core. Ten bits of byte count data allow payload size up to 1,023 bytes. A byte count of zero is valid, meaning that the host sent no isochronous data during the previous frame. The default values of these registers are unknown.

Byte counts are valid only for OUT endpoints. The byte counts indicate the number of bytes remaining in the endpoint's OUT FIFO. Every time the 8051 reads a byte from the ISODATA Register, the byte count decrements by one.

To read USB OUT data, the 8051 first reads byte count registers OUTnBCL and OUTnBCH to determine how many bytes to transfer out of the OUT FIFO. (The 8051 can also quickly test ISO output endpoints for zero byte counts using the ZBCOUT Register.) Then, the CPU reads that number of bytes from the ISODATA Register. Separate byte counts are maintained for each endpoint, so the CPU can read the FIFOs in a discontinuous manner. For example, if EP8 indicates a byte count of 100, and EP9 indicates a byte count of 50, the CPU could read 50 bytes from EP8, then read 10 bytes from EP9, and resume reading EP8.

There are no byte count registers for the IN endpoints. The USB core automatically tracks the number of bytes loaded by the 8051.

If the 8051 does not load an IN isochronous endpoint FIFO during a 1-ms frame, and the host requests data from that endpoint during the next frame (IN token), the USB core responds according to the setting of the ISOSEND0 Bit (USBPAIR.7). If ISOSEND0=1, the core returns a zero-

length data packet in response to the host IN token. If ISOSEND=0, the core does not respond to the IN token.

It is the responsibility of the 8051 programmer to ensure that the number of bytes written to the IN FIFO does not exceed the maximum packet size as reported during enumeration.

## 15.30 CPU Registers

**CPUCS** **CPU Control and Status** **7F92**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **RV3** | **RV2** | **RV1** | **RV0** | **24/48** | **CLKINV** | **CLKOE** | **8051RES** |
| R | R | R | R | R | R | R/W | R |
| RV3 | RV2 | RV1 | RV0 | 0 | 0 | 1 | 1 |

*Figure 15-69. CPU Control and Status Register*

This register enables the CLKOE output and permits the host to reset the 8051 using a firmware download.

**Bit 7-4:** **RV[3..0]** *Silicon Revision*

These register bits define the silicon revision. Consult individual Cypress Semiconductor data sheets for values.

**Bit 3:** **24/48** *8051 Clock Frequency*

This read-only bit indicates that the 8051 clock rate is 24 or 48 MHz. This bit is set at power-on according to a bit in the EEPROM connected to the EZ-USB FX $I^2C$-compatible bus. If no EEPROM is connected, the EZ-USB FX defaults to a 24-MHz 8051 clock. Once running (after boot), the 8051 cannot change the clock rate.

**Bit 2:** **CLKINV** *Invert the CLKOUT signal*

This read-only bit indicates that the CLKOUT signal is inverted. This bit is set at power-on according to a bit in the EEPROM connected to the EZ-USB FX $I^2C$-compatible bus. If no EEPROM is connected, the EZ-USB FX defaults to a non-inverted 24-MHz 8051 clock.

When CLKINV=0, the clock has the polarity shown in all the timing diagrams in this manual. When CLKINV=1, the clock is inverted.

**Bit 1:** **CLKOE** *CLKOUT pin output enable*

The CLKOUT signal may be disabled by floating the CLKOUT pin. The 8051 does this by clearing CLKOE. This is a good idea if the CLKOUT pin is not used since it reduces EMI.

**Bit 0:** **8051RES** *8051 reset*

The USB host writes "1" to this bit to reset the 8051, and "0" to run the 8051. Only the USB host can write this bit.

## 15.31 Port Configuration

**PORTACFG** **I/O Port A Configuration** **7F93**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **RxD1OUT** | **RxD0OUT** | **FRD** | **FWR** | **CS** | **OE** | **T1OUT** | **T0OUT** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTBCFG** **I/O Port B Configuration** **7F94**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **T2OUT** | **INT6** | **INT5** | **INT4** | **TXD1** | **RXD1** | **T2EX** | **T2** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORTCCFG** **I/O Port C Configuration** **7F95**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **RD** | **WR** | **T1** | **T0** | **INT1** | **INT0** | **TXD0** | **RXD0** |
| 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 15-70. I/O Port Configuration Registers*

These three registers select between I/O ports and various alternate functions for I/O ports PORTA, PORTB, and PORTC. They are read/write by the 8051.

When PORTnCFG=0, the port pin functions as I/O, using the OUT, PINS, and OE control bits. Data written to an OUTn Registers appears on an I/O Port pin if the corresponding output enable bit (OEn) is HI.

When PORTnCFG=1, the pin assumes the alternate function shown in Table 15-15 on the following page.

For more information, see *Section 4.3. "Input/Output Port Registers".*

*These registers are used in conjunction with the IFCONFIG PORTACF2 Registers to define the pin functions.*

**Table 15-15. I/O Pin Alternate Functions**

| I/O | Name | Alternate Functions |
|---|---|---|
| PA0 | T0OUT | Timer 0 Output |
| PA1 | T1OUT | Timer 1 Output |
| PA2 | OE# | External Memory Output Enable |
| PA3 | CS# | External Memory Chip Select |
| PA4 | FWR# | Fast Access Write Strobe |
| PA5 | FRD# | Fast Access Read Strobe |
| PA6 | RXD0OUT | Mode 0: UART0 Synchronous Data Output |
| PA7 | RXD1OUT | Mode 0: UART1 Synchronous Data Output |
| PB0 | T2 | Timer/Counter 2 Clock Input |
| PB1 | T2EX | Timer/Counter 2 Capture/Reload Input |
| PB2 | RxD1 | Serial Port 1 Input |
| PB3 | TxD1 | Mode 0: Clock Output<br>Modes 1-3: Serial Port 1 Data Output |
| PB4 | INT4 | INT4 Interrupt Request |
| PB5 | INT5# | INT5 Interrupt Request |
| PB6 | INT6 | INT6 Interrupt Request |
| PB7 | T2OUT | Timer/Counter 2 Overflow Indication |
| PC0 | RxD0 | Serial Port 0 Input |
| PC1 | TxD0 | Mode 0: Clock Output<br>Modes 1-3: Serial Port 0 Data Output |
| PC2 | INT0# | INT0 Interrupt Request |
| PC3 | INT1# | INT1 Interrupt Request |
| PC4 | T0 | Timer/Counter 0 External Input |
| PC5 | T1 | Timer/Counter 1 External Input |
| PC6 | WR# | External Memory Write Strobe |
| PC7 | RD# | External Memory Read Strobe |

## 15.32 Input/Output Port Registers A - C

For more information, see Section 4.3. *"Input/Output Port Registers."*

### 15.32.1 Outputs

The OUTn Registers provide the data that drives the port pin when OE=1 **and** the pin is configured for port output. If the port pin is selected as an input (OE=0), the value stored in the corresponding OUTn Bit is stored in an output latch but not used.

**OUTA** **Port A Outputs** **7F96**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUTA7** | **OUTA6** | **OUTA5** | **OUTA4** | **OUTA3** | **OUTA2** | **OUTA1** | **OUTA0** |
| 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 15-71. Port A Outputs*

**OUTB** **Port B Outputs** **7F97**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUTB7** | **OUTB6** | **OUTB5** | **OUTB4** | **OUTB3** | **OUTB2** | **OUTB1** | **OUTB0** |
| 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 15-72. Port B Outputs*

**OUTC** **Port C Outputs** **7F98**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUTC7** | **OUTC6** | **OUTC5** | **OUTC4** | **OUTC3** | **OUTC2** | **OUTC1** | **OUTC0** |
| 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 15-73. Port C Outputs*

### 15.32.2 Pins

The PINSn Registers contain the current value of the port pins, whether they are selected as I/O ports or as alternate functions.

**PINSA** **Port A Pins** **7F99**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **PINA7** | **PINA6** | **PINA5** | **PINA4** | **PINA3** | **PINA2** | **PINA1** | **PINA0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-74. Port A Pins*

**PINSB** **Port B Pins** **7F9A**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **PINB7** | **PINB6** | **PINB5** | **PINB4** | **PINB3** | **PINB2** | **PINB1** | **PINB0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-75. Port B Pins*

**OUTC** **Port C Pins** **7F98**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **PINC7** | **PINC6** | **PINC5** | **PINC4** | **PINC3** | **PINC2** | **PINC1** | **PINC0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-76. Port C Pins*

### 15.32.3 Output Enables

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.”

**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 |

*Figure 15-77. Port A Output Enable*

**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 |

*Figure 15-78. Port B Output Enable*

**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 15-79. Port C Output Enable*

## 15.33 Isochronous Control/Status Registers

**ISOERR** **Isochronous OUT EP Error** **7FA0**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **ISO15ERR** | **ISO14ERR** | **ISO13ERR** | **ISO12ERR** | **ISO11ERR** | **ISO10ERR** | **ISO9ERR** | **ISO8ERR** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-80. Isochronous OUT Endpoint Error Register*

The ISOERR bits are updated at every SOF. They indicate that a CRC error was received on a data packet for the current frame. The ISOERR Bit status refers to the USB data received in the previous frame, and which is currently in the endpoint's OUT FIFO. If the ISOERR Bit = 1, indicating a bad CRC check, the data is still available in the OUTnDATA Register.

**ISOCTL** **Isochronous Control** **7FA1**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **-** | **PPSTAT** | **MBZ** | **MBZ** | **ISODISAB** |
| R | R | R | R | R | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-81. Isochronous Control Register*

**Bit 3:** **PPSTAT** *Ping-Pong Status*

This bit indicates the isochronous buffer currently in use by the USB core. It is used only for diagnostic purposes.

**Bits 2,1:** **MBZ** *Must be zero*

These bits must always be written with zeros.

**Bit 0:** **ISODISAB** *ISO Endpoints Disable*

ISODISAB=0 enables all 16 isochronous endpoints

ISODISAB=1 disables *all 16* isochronous endpoints, making the 2,048 bytes of isochronous FIFO memory available as 8051 data memory at 0x2000-0x27FF.

**ZBCOUT** **Zero Byte Count Bits** **7FA2**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **EP15** | **EP14** | **EP13** | **EP12** | **EP11** | **EP10** | **EP9** | **EP8** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-82. Zero Byte Count Register*

**Bits 0-7:** **EP(n)** *Zero Byte Count for ISO OUT Endpoints*

The 8051 can check these bits as a fast way to check all of the OUT isochronous endpoints at once for no data received during the previous frame. A "1" in any bit position means that a zero byte Isochronous OUT packet was received for the indicated endpoint.

## 15.34 I²C-Compatible Registers

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

**I2CS** **I²C-Compatible Control and Status** **7FA5†**
**Read/write latency applies**

| 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 |

**I2DAT** **I²C-Compatible Data** **7FA6†**
**Read/write latency applies**

| 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 15-83. I²C-Compatible Transfer Registers*

The 8051 uses these registers to transfer data over the EZ-USB FX I$^2$C-compatible bus. For I$^2$C-compatible peripherals that support it, the EZ-USB FX I$^2$C-compatible bus can run at 400 KHz. For compatibility, the EZ-USB FX powers-up at the 100-KHz frequency.

In the EZ-USB FX family, an I$^2$C-compatible Interrupt Request occurs on INT3 whenever the DONE Bit (I2CS.0) makes a zero-to-one transition. This interrupt signals the 8051 that the I$^2$C-compatible controller is ready for another command. For more information on the I$^2$C-compatible interrupt, see Section 12.14. *"I2C-Compatible STOP Complete Interrupt."*

**Bit 7: START** *Signal START condition*

The 8051 sets the START Bit to "1" to prepare an I$^2$C-compatible bus transfer. If START=1, the next 8051 load to I2DAT will generate the start condition followed by the serialized byte of data in I2DAT. The 8051 loads byte data into I2DAT after setting the START Bit. The I$^2$C-compatible controller clears the START Bit during the ACK interval.

**Bit 6: STOP** *Signal STOP condition*

The 8051 sets STOP=1 to terminate an I$^2$C-compatible bus transfer. The I$^2$C-compatible controller clears the STOP Bit after completing the STOP condition. If the 8051 sets the STOP Bit during a byte transfer, the STOP condition will be generated immediately following the ACK phase of the byte transfer. If no byte transfer is occurring when the STOP Bit is set, the STOP condition will be carried out immediately on the bus. Data should not be written to I2CS or I2DAT until the STOP Bit returns low.

**Bit 5: LASTRD** *Last Data Read*

To read data over the I$^2$C-compatible bus, an I$^2$C-compatible master floats the SDA line and issues clock pulses on the SCL line. After every eight bits, the master drives SDA low for one clock to indicate ACK. To signal the last byte of the read transfer, the master floats SDA at ACK time to instruct the slave to stop sending. This is controlled by the 8051 by setting LastRD=1 before reading the last byte of a read transfer. The I$^2$C-compatible controller clears the LastRD Bit at the end of the transfer (at ACK time).

*Setting LastRD does not automatically generate a STOP condition. The 8051 should also set the STOP Bit at the end of a read transfer.*

**Bit 4-3: ID1,ID0** *Boot EEPROM ID*

These bits are set by the boot loader to indicate whether an 8-bit address or 16-bit address EEPROM at slave address 000 or 001 was detected at power-on. Normally, they are used for debug purposes only.

**Bit 2:** **BERR** *Bus Error*

This bit indicates an I²C-compatible bus error. BERR=1 indicates that there was bus contention, which results when an outside device drives the bus LO when it shouldn't, or when another bus master wins arbitration, taking control of the bus. BERR is cleared when 8051 reads or writes the IDATA Register.

**Bit 1:** **ACK** *Acknowledge Bit*

Every ninth SCL or a write transfer the slave indicates reception of the byte by asserting ACK. The EZ-USB FX controller floats SDA during this time, samples the SDA line, and updates the ACK Bit with the complement of the detected value. ACK=1 indicates acknowledge, and ACK=0 indicates not-acknowledge. The USB core updates the ACK Bit at the same time it sets DONE=1. The ACK Bit should be ignored for read transfers on the bus.

**Bit 0:** **DONE** *I²C-CompatibleTransfer DONE*

The I²C-compatible controller sets this bit whenever it completes a byte transfer, right after the ACK stage. The controller also generates an I²C-compatible Interrupt Request (8051 INT3) when it sets the DONE Bit. The I²C-compatible controller automatically clears the DONE Bit and the I²C-compatible Interrupt Request bit whenever the 8051 reads or writes the I2DAT Register.

**I2CMODE** **I²C-Compatible Mode** **7FA7†**

**Read/write latency applies**

| 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 15-84. I²C-Compatible Mode Register*

The I²C-compatible STOP Bit Interrupt Request is activated when the STOP Bit makes a 1-0 transition. 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.

## 15.35 Interrupts

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

**IVEC** **Interrupt Vector** **7FA8**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **IV4** | **IV3** | **IV2** | **IV1** | **IV0** | **0** | **0** |
| R | R | R | R | R | R | R | R |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-85. Interrupt Vector Register*

IVEC indicates the source of an interrupt from the USB Core. When the USB core generates an INT2 (USB) Interrupt Request, it updates IVEC to indicate the source of the interrupt. The interrupt sources are encoded on IV[4..0] as shown in *Figure 12-1.*

**IN07IRQ** **Endpoint 0-7 IN Interrupt Request** **7FA9**†

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **IN7IR** | **IN6IR** | **IN5IR** | **IN4IR** | **IN3IR** | **IN2IR** | **IN1IR** | **IN0IR** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**OUT07IRQ** **Endpoint 0-7 OUT Interrupt Requests** **7FAA**†

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUT7IR** | **OUT6IR** | **OUT5IR** | **OUT4IR** | **OUT3IR** | **OUT2IR** | **OUT1IR** | **OUT0IR** |
| 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 15-86. IN/OUT Interrupt Request (IRQ) Registers*

These interrupt request (IRQ) registers indicate the pending interrupts for each bulk endpoint. An interrupt request (IR) Bit becomes active when the BSY Bit for an endpoint makes a transition from one to zero (when the endpoint becomes *un-busy*, giving access to the 8051). The IR bits function

independently of the Interrupt Enable (IE) bits, so interrupt requests are held whether or not the interrupts are enabled.

The 8051 clears an interrupt request bit by writing a “1” to it. (See the following Note).

*Do **not** clear an IRQ Bit by reading an IRQ Register, ORing its contents with a bit mask, and writing back the IRQ Register. This will clear ALL pending interrupts. Instead, simply write the bit mask value (with a “1” in the bit position of the IRQ you want to clear) directly to the IRQ Register.*

**USBIRQ** **USB Interrupt Request** **7FAB†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **IBNIR** | **USESIR** | **SUSPIR** | **SUTOKIR** | **SOFIR** | **SUDAVIR** |
| 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 15-87. USB Interrupt Request (IRQ) Registers*

USBIRQ indicates the interrupt request status of the USB reset, suspend, setup token, start of frame, and setup data available interrupts.

**Bit 5: IBNIR** *IN Bulk NAK Interrupt Request*

The USB core sets this bit when any of the IN bulk endpoints responds to an IN token with a NAK. This interrupt occurs when the host sends an IN token to a bulk IN endpoint which has not been armed by the 8051 writing its byte count register. Individual enables and requests (per endpoint) are controlled by the IBNIRQ and IBNIEN Registers (7FB0, 7FB1). Write a “1” to this bit to clear the interrupt request.

**Bit 4: URESIR** *USB Reset Interrupt Request*

The USB core sets this bit to “1” when it detects a USB bus reset.

Because this bit can change state while the 8051 is in reset, it may be active when the 8051 comes out of reset, although it is reset to “0” by a power-on reset. Write a “1” to this bit to clear the interrupt request. See *Chapter 13. "EZ-USB FX Resets"* for more information about this bit.

**Bit 3:** **SUSPIR** *USB Suspend Interrupt Request*

The USB core sets this bit to “1” when it detects USB SUSPEND signaling (no bus activity for 3 ms). Write a “1” to this bit to clear the interrupt request.

Because this bit can change state while the 8051 is in reset, it may be active when the 8051 comes out of reset, although it is reset to “0” by a power-on reset. See *Chapter 14. "EZ-USB FX Power Management"* for more information about this bit.

**Bit 2:** **SUTOKIR** *SETUP Token Interrupt Request*

The USB core sets this bit to “1” when it receives a SETUP token. Write a “1” to this bit to clear the interrupt request. See *Chapter 9. "EZ-USB FX Endpoint Zero"* for more information on the handling of SETUP tokens.

Because this bit can change state while the 8051 is in reset, it may be active when the 8051 comes out of reset, although it is reset to “0” by a power-on reset.

**Bit 1:** **SOFIR** *Start of frame Interrupt Request*

The USB core sets this bit to “1” when it receives a SOF packet. Write a “1” to this bit to clear the interrupt request.

Because this bit can change state while the 8051 is in reset, it may be active when the 8051 comes out of reset, although it is reset to “0” by a power-on reset.

**Bit 0:** **SUDAVIR** *SETUP data available Interrupt Request*

The USB core sets this bit to “1” when it has transferred the eight data bytes from an endpoint zero SETUP packet into internal registers (at SETUPDAT). Write a “1” to this bit to clear the interrupt request.

Because this bit can change state while the 8051 is in reset, it may be active when the 8051 comes out of reset, although it is reset to “0” by a power-on reset.

**IN07EN** **Endpoint 0-7 IN Interrupt Enables** **7FAC†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **IN7IEN** | **IN6IEN** | **IN5IEN** | **IN4IEN** | **IN3IEN** | **IN2IEN** | **IN1IEN** | **IN0IEN** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**OUT07IEN** **Endpoint 0-7 OUT Interrupt Enables** **7FAD†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUT7IEN** | **OUT6IEN** | **OUT5IEN** | **OUT4IEN** | **OUT3IEN** | **OUT2IEN** | **OUT1IEN** | **OUT0IEN** |
| 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 15-88. IN/OUT Interrupt Enable Registers*

The Endpoint Interrupt Enable individually enable the BULK endpoint interrupts. They do not affect the endpoint action, only the generation of an interrupt in response to endpoint events.

When the IEN Bit for an endpoint is “0,” the interrupt request bit for that endpoint is ignored, but saved. When the IEN Bit for an endpoint is “1,” any IRQ Bit equal to “1” generates an 8051 INT2 Request.

*The INT2 interrupt (EIE.0) and the 8051 global interrupt enable (EA) must be enabled for the end-point interrupts to propagate to the 8051. Once the INT2 interrupt is active, it must be cleared by software.*

**USBIEN** **USB Interrupt Enable** **7FAE†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **IBNIE*** | **URESIE** | **SUSPIE** | **SUTOKIE** | **SOFIE** | **SUDAVIE** |
| 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 15-89. USB Interrupt Enable Register*

USBIEN bits gate the interrupt request to the 8051 for USB reset, suspend, SETUP token, start of frame, and SETUP data available.

**Bit 5:** **IBNIE** *IN bulk NAK Interrupt Enable*

The 8051 sets this bit to enable the IN-bulk-NAK interrupt. This interrupt occurs when the host sends an IN token to a bulk IN endpoint which has not been*armed* by the 8051 writing its byte count register. Individual enables and requests (per endpoint) are controlled by the IBNIRQ and IBNIEN Registers (7FB0, 7FB1).

**Bit 4:** **URESIE** *USB Reset Interrupt Enable*

This bit is the interrupt mask for the URESIR Bit. When this bit is "1," the interrupt is enabled, when it is "0," the interrupt is disabled.

**Bit 3:** **SUSPIE** *USB Suspend Interrupt Enable*

This bit is the interrupt mask for the SUSPIR Bit. When this bit is "1," the interrupt is enabled, when it is "0," the interrupt is disabled.

**Bit 2:** **SUTOKIE** *SETUP Token Interrupt Enable*

This bit is the interrupt mask for the SUTOKIR Bit. When this bit is "1," the interrupt is enabled, when it is "0," the interrupt is disabled.

**Bit 1:** **SOFIE** *Start of frame Interrupt Enable*

This bit is the interrupt mask for the SOFIE Bit. When this bit is "1," the interrupt is enabled, when it is "0," the interrupt is disabled.

**Bit 0:** **SUDAVIE** *SETUP data available Interrupt Enable*

This bit is the interrupt mask for the SUDAVIE Bit. When this bit is "1," the interrupt is enabled, when it is "0," the interrupt is disabled.

**USBBAV** **Breakpoint and Autovector** **7FAF**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **-** | **-** | **INT2SFC** | **BREAK** | **BPPULSE** | **BPEN** | **AVEN** |
| 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 15-90. Breakpoint and Autovector Register*

**Bit 4:** **INT2SFC** *Interrupt 2 cleared by SFR*

If INT2SFC=1, the IRQ2 flag can be quickly cleared by writing any value to the INT2CLR SFR .

**Bit 3:** **BREAK** *Breakpoint enable*

The BREAK Bit is set when the 8051 address bus matches the address held in the bit breakpoint address registers (7FB2, 7FB3). The BKPT pin reflects the state of this bit. The 8051 writes a "1" to the BREAK Bit to clear it. It is not necessary to clear the BREAK Bit if the pulse mode bit (BPPULSE) is set.

**Bit 2:** **BPPULSE** *Breakpoint pulse mode*

The 8051 sets this bit to "1" to pulse the BREAK Bit (and BKPT pin) high for 8 CLKOUT cycles when the 8051 address bus matches the address held in the breakpoint address registers. When this bit is set to "0," the BREAK Bit (and BKPT pin) remains high until it is cleared by the 8051.

**Bit 1:** **BPEN** *Breakpoint enable*

If this bit is "1," a BREAK signal is generated whenever the 16-bit address lines match the value in the Breakpoint Address Registers (BPADDRH/L). The behavior of the BREAK Bit and associated BKPT pin signal is either latched or pulsed, depending on the state of the BPPULSE Bit.

**Bit 0:** **AVEN** *Auto-vector enable*

If this bit is "1," the EZ-USB FX Auto-vector feature is enabled for USB (INT2) interrupts. If it is 0, the auto-vector feature is disabled. See *Chapter 12. "EZ-USB FX Interrupts"* for more information on the auto-vector feature. Note: a separate bit, AV4EN in the INT4SETUP (785E) enables the INT4 autovector.

**IBNIRQ** **IN Bulk NAK Interrupt Requests** **7FB0†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **EP6IR** | **EP5IR** | **EP4IR** | **EP3IR** | **EP2IR** | **EP1IR** | **EP0IR** |
| 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 15-91. IN Bulk NAK Interrupt Request Register*

These bits are set when a bulk IN endpoint (0-6) received an IN token while the endpoint was not armed for data transfer. In this case the SIE automatically sends a NAK response, and sets the corresponding IBNIRQ Bit. If the IBN interrupt is enabled (USBIEN.5=1), and the endpoint interrupt is enabled in the IBNIEN Register, an interrupt is request generated. The 8051 can test the

IBNIRQ Register to determine which of the endpoints caused the interrupt. The 8051 clears an IBNIRQ Bit by writing a “1” to it.

**IBNIEN** **IN Bulk NAK Interrupt Enables** **7FB1†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **EP6IE** | **EP5IE** | **EP4IE** | **EP3IE** | **EP2IE** | **EP1IE** | **EP0IE** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| x | x | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-92. IN Bulk NAK Interrupt Enable Register*

Each of the individual IN endpoints may be enabled for an IBN interrupt using the IBNEN Register. The 8051 sets an interrupt enable bit to 1 to enable the corresponding interrupt.

**BPADDRH** **Breakpoint Address High** **7FB2**

| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**BPADDRL** **Breakpoint Address Low** **7FB3**

| 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 15-93. IN/OUT Interrupt Enable Registers*

When the current 16-bit address (code or XDATA) matches the BPADDRH/BPADDRL address, a breakpoint event occurs. The BPPULSE and BPEN bits in the USBBAV Register control the action taken on a breakpoint event.

If the BPEN Bit is “0,” address breakpoints are ignored. If BPEN is “1” and BPPULSE is “1,” an 8 CLKOUT wide pulse appears on the BKPT pin. If BPEN is “1” and BPPULSE is “0,” the BKPT pin remains active until the 8051 clears the BREAK Bit by writing “1” to it.

## 15.36 Endpoint 0 Control and Status Registers

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

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

**Read/write latency applies**

| 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†**

**Read/write latency applies**

| 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†**

**Read/write latency applies**

| 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 15-94. Port Configuration Registers*

These registers control EZ-USB FX 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.

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 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 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 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 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.

*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.*

## 15.37 Endpoint 1-7 Control and Status Registers

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

Endpoints 1-7 IN and OUT are used for bulk or interrupt data. Table 15-16 shows the addresses for the control/status and byte count registers associated with these endpoints. The bi-directional CONTROL endpoint zero registers are described in Section 15.36. *"Endpoint 0 Control and Status Registers."*

**Table 15-16. Control and Status Register Addresses for Endpoints 0-7**

| Address | Function | Name |
|---|---|---|
| 7FB4† | Control and Status - Endpoint IN0 | EP0CS |
| 7FB5† | Byte Count - Endpoint IN0 | IN0BC |
| 7FB6† | Control and Status - Endpoint IN1 | IN1CS |
| 7FB7† | Byte Count - Endpoint IN1 | IN1BC |
| 7FB8† | Control and Status - Endpoint IN2 | IN2CS |
| 7FB9† | Byte Count - Endpoint IN2 | IN2BC |
| 7FBA† | Control and Status - Endpoint IN3 | IN3CS |
| 7FBB† | Byte Count - Endpoint IN3 | IN3BC |
| 7FBC† | Control and Status - Endpoint IN4 | IN4CS |
| 7FBD† | Byte Count - Endpoint IN4 | IN4BC |
| 7FBE† | Control and Status - Endpoint IN5 | IN5CS |
| 7FBF† | Byte Count - Endpoint IN5 | IN5BC |
| 7FC0† | Control and Status - Endpoint IN6 | IN6CS |
| 7FC1† | Byte Count - Endpoint IN6 | IN6BC |
| 7FC2† | Control and Status - Endpoint IN7 | IN7CS |
| 7FC3† | Byte Count - Endpoint IN7 | IN7BC |
| 7FC4 | *Reserved* | |
| 7FC5† | Byte Count - Endpoint OUT0 | OUT0BC |
| 7FC6† | Control and Status - Endpoint OUT1 | OUT1CS |
| 7FC7† | Byte Count - Endpoint OUT1 | OUT1BC |
| 7FC8† | Control and Status - Endpoint OUT2 | OUT2CS |
| 7FC9† | Byte Count - Endpoint OUT2 | OUT2BC |
| 7FCA† | Control and Status - Endpoint OUT3 | OU37CS |
| 7FCB† | Byte Count - Endpoint OUT3 | OUT3BC |
| 7FCC† | Control and Status - Endpoint OUT4 | OUT4CS |
| 7FCD† | Byte Count - Endpoint OUT4 | OUT4BC |
| 7FCE† | Control and Status - Endpoint OUT5 | OUT5CS |
| 7FCF† | Byte Count - Endpoint OUT5 | OUT5BC |
| 7FD0† | Control and Status - Endpoint OUT6 | OUT6CS |
| 7FD1† | Byte Count - Endpoint OUT6 | OUT6BC |
| 7FD2† | Control and Status - Endpoint OUT7 | OUT7CS |
| 7FD3† | Byte Count - Endpoint OUT7 | OUT7BC |

| INnCS | Endpoint (1-7) IN Control and Status | 7FB6-7FC2*† |
|---|---|---|
| | | Read/write latency applies |

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | **INnBSY** | **INnSTL** |
| R | R | R | R | R | R | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* See Table 15-16 for individual control/status register addresses.

*Figure 15-95. IN Control and Status Registers*

**Bit 1: INnBSY** *IN Endpoint (1-7) Busy*

The BSY Bit indicates the status of the endpoint's IN Buffer INnBUF. The USB core sets BSY=0 when the endpoint's IN buffer is empty and ready for loading by the 8051. The 8051 causes BSY=1 by loading the endpoint's byte count register.

When BSY=1, the 8051 should not write data to an IN endpoint buffer, because the endpoint FIFO could be in the act of transferring data to the host over the USB. BSY=0 when the USB IN transfer is complete and endpoint RAM data is available for 8051 access. USB IN tokens for the endpoint are NAKd while BSY=0 (the 8051 is still loading data into the endpoint buffer).

A 1-to-0 transition of BSY (indicating that the 8051 can access the buffer) generates an interrupt request for the IN endpoint. After the 8051 writes the data to be transferred to the IN endpoint buffer, it loads the endpoint's byte count register with the number of bytes to transfer, which automatically sets BSY=1. This enables the IN transfer of data to the host in response to the next IN token. Again, the CPU should never load endpoint data while BSY=1.

The 8051 writes a "1" to an IN endpoint busy bit to disarm a previously armed endpoint. (This sets BSY=0.) The 8051 program should do this only after a USB bus reset, or when the host selects a new interface or alternate setting that uses the endpoint. This prevents stale data from a previous setting from being accepted by the host's first IN transfer that uses the new setting.

*Even though the register description shows bit 1 as "R/W," the 8051 can only clear this bit by writing a "1" to it. The 8051 can not directly set this bit.*

To disarm a paired IN endpoint, write a "1" to the busy bit for *both* endpoints in the pair.

**Bit 0:** **INnSTL** *IN Endpoint (1-7) Stall*

The 8051 sets this bit to "1" to *stall* an endpoint, and to "0" to clear a stall.

When the stall bit is "1," the USB core returns a STALL Handshake for all requests to the endpoint. This notifies the host that something unexpected has happened.

The 8051 sets an endpoint's stall bit under two circumstances:

1. The host sends a "Set_Feature—Endpoint Stall" Request to the specific endpoint.
2. The 8051 encounters any *show stopper* error on the endpoint, and sets the stall bit to tell the host to halt traffic to the endpoint.

The 8051 clears an endpoint's stall bit under two circumstances:

1. The host sends a "Clear_Feature—Endpoint Stall" Request to the specific endpoint.
2. The 8051 receives some other indication from the host that the stall should be cleared (this is referred to as "host intervention" in the USB Specification). This indication could be a USB bus reset.

All stall bits are automatically cleared when the EZ-USB FX chip ReNumerates™ by pulsing the DISCON Bit HI.

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

**Read/write latency applies**

| 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 15-16 for individual byte count register addresses.

*Figure 15-96. 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 (*Chapter 6. "EZ-USB FX 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.

**OUTnCS** **Endpoint (1-7) OUT Control and Status** **7FC6-7FD2*†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | **OUTnBSY** | **OUTnSTL** |
| R | R | R | R | R | R | R | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* See Table 15-16 for individual control/status register addresses.

*Figure 15-97. OUT Control and Status Registers*

**Bit 1: OUTnBSY** *OUT Endpoint (1-7) Busy*

The BSY Bit indicates the status of the endpoint's OUT Buffer OUTnBUF. The USB core sets BSY=0 when the host data is available in the OUT buffer. The 8051 sets BSY=1 by loading the endpoint's byte count register.

When BSY=1, endpoint RAM data is invalid--the endpoint buffer has been emptied by the 8051 and is waiting for new OUT data from the host, or it is the process of being loaded over the USB. BSY=0 when the USB OUT transfer is complete and endpoint RAM data in OUTnBUF is available for the 8051 to read. USB OUT tokens for the endpoint are NAKd while BSY=1 (the 8051 is still reading data from the OUT endpoint).

A 1-to-0 transition of BSY (indicating that the 8051 can access the buffer) generates an interrupt request for the OUT endpoint. After the 8051 reads the data from the OUT endpoint buffer, it loads the endpoint's byte count register with any value to re-arm the endpoint, which automatically sets BSY=1. This enables the OUT transfer of data from the host in response to the next OUT token. The CPU should never read endpoint data while BSY=1.

**Bit 0: OUTnSTL** *OUT Endpoint (1-7) Stall*

The 8051 sets this bit to "1" to *stall* an endpoint, and to "0" to clear a stall.

When the stall bit is "1," the USB core returns a STALL Handshake for all requests to the endpoint. This notifies the host that something unexpected has happened.

The 8051 sets an endpoint's stall bit under two circumstances:

1. The host sends a "Set_Feature—Endpoint Stall" Request to the specific endpoint.
2. The 8051 encounters any *show stopper* error on the endpoint, and sets the stall bit to tell the host to halt traffic to the endpoint.

The 8051 clears an endpoint's stall bit under two circumstances:

1. The host sends a "Clear_Feature—Endpoint Stall" Request to the specific endpoint.
2. The 8051 receives some other indication from the host that the stall should be cleared (this is referred to as "host intervention" in the USB Specification).

All stall bits are automatically cleared when the EZ-USB FX chip ReNumerates™.

**OUTnBC** **Endpoint (1-7) OUT Byte Count** **7FC7-7FD3*†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **-** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| R | R | R | R | R | R | R | R/W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* See Table 15-16 for individual control/status register addresses.

*Figure 15-98. OUT Byte Count Registers*

The 8051 reads this register to determine the number of bytes sent to an OUT endpoint. Legal sizes are 0 - 64 bytes.

Each EZ-USB FX bulk OUT endpoint has a byte count register, which serves two purposes. The 8051 *reads* the byte count register to determine how many bytes were received during the last OUT transfer from the host. The 8051 *writes* the byte count register (with any value) to tell the USB core that it has finished reading bytes from the buffer, making the buffer available to accept the next OUT transfer. Writing the byte count register sets the endpoint's BSY Bit to "1."

When the register-pairing feature is used, OUT2BC is used for the EP2/EP3 pair, OUT4BC is used for the EP4/EP5 pair, and OUT6BC is used for the EP6/EP7 pair. The odd-numbered byte count registers should not be used. When the 8051 writes a byte to the even numbered byte count register, the USB core switches buffers. If the other buffer already contains data to be read by the 8051, the OUTnBSY Bit remains at "0."

*All OUT tokens are NAKd until the 8051 is released from RESET, whereupon the ACK/NAK behavior is based on pairing.*

## 15.38 Global USB Registers

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

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

**Read/write latency applies**

| 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†**

**Read/write latency applies**

| 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 15-99. Setup Data Pointer High/Low Registers*

When the EZ-USB FX chip receives a “Get_Descriptor” Request on endpoint zero, it can instruct the 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.

*Any host request that uses the EZ-USB FX 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 USB core how many bytes to transfer using the Setup Data Pointer.*

*The USB core transfers the lesser of (a) the bytes requested in the SETUP packet, and (b) the bytes in the length field of the descriptor pointed to by the Setup Data Pointer.*

**USBCS** **USB Control and Status** **7FD6†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **WAKESRC** | **-** | **-** | **-** | **DISCON** | **DISCOE** | **RENUM** | **SIGRSUME** |
| R/W | R | R | R | R/W | R/W | R/W | R/W |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

*Figure 15-100. USB Control and Status Registers*

**Bit 7: WAKESRC** *Wakeup source*

This bit indicates that a high to low transaction was detected on the WAKEUP# pin. Writing a "1" to this bit resets it to "0."

**Bit 3: DISCON** *Signal a Disconnect on the DISCON# pin*

The EZ-USB FX DISCON# pin reflects the complement of this bit. This bit is normally set to 0.

**Bit 2: DISCOE** *Disconnect Output Enable*

DISCOE controls the output buffer on the DISCON# pin. When DISCOE=0, the pin floats, and when DISCOE=1, it drives to the complement of the DISCON Bit (above).

DISCOE is used in conjunction with the RENUM Bit to perform ReNumeration™, (*Chapter 5. "EZ-USB FX Enumeration & ReNumeration™"*).

**Bit 1: RENUM** *ReNumerate*

This bit controls which entity, the USB core or the 8051, handles USB device requests. When RENUM=0, the USB core handles all device requests. When RENUM=1, the 8051 handles all device requests except Set_Address.

The 8051 sets RENUM=1 during a bus disconnect to transfer USB control to the 8051. The USB core automatically sets RENUM=1 under two conditions:

1. Completion of a "B6" boot load (*Chapter 5. "EZ-USB FX Enumeration & ReNumeration™"*.
2. When external memory is used (EA=1) and no boot $I^2C$-compatible EEPROM is used (see Section 13.3.3. *"External ROM"*).

**Bit 0:** **SIGRSUME** *Signal remote device resume*

The 8051 sets SIGRSUME=1 to drive the “K” state onto the USB bus. This should be done only by a device that is capable of remote wakeup, and then only during the SUSPEND state. To signal RESUME, the 8051 sets SIGRSUME=1, waits 10-15 ms, then sets SIGRSUME=0.

**TOGCTL** **Data Toggle Control** **7FD7†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **Q** | **S** | **R** | **IO** | **0** | **EP2** | **EP1** | **EP0** |
| R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| x | x | x | x | x | x | x | x |

*Figure 15-101. Data Toggle Control Register*

**Bit 7:** **Q** *Data Toggle Value*

Q=0 indicates DATA0 and Q=1 indicates DATA1, for the endpoint selected by the I/O and EP[2..0] bits. The 8051 writes the endpoint select bits (IO and EP[2..0]), before reading this value.

**Bit 6:** **S** *Set Data Toggle to DATA1*

After selecting the desired endpoint by writing the endpoint select bits (IO and EP[2..0]) the 8051 sets S=1 to set the data toggle to DATA1. The endpoint selection bits should not be changed while this bit is written.

*At this writing there is no known reason to set an endpoint data toggle to 1. This bit is provided for generality and testing only.*

**Bit 5:** **R** *Set Data Toggle to DATA0*

After selecting the desired endpoint by writing the endpoint select bits (IO and EP[2..0]) the 8051 sets R=1 to set the data toggle to DATA0. The endpoint selection bits should not be changed while this bit is written. For advice on when to reset the data toggle, see *Chapter 9. "EZ-USB FX Endpoint Zero"*.

**Bit 4:** **IO** *Select IN or OUT endpoint*

The 8051 sets this bit to select an endpoint direction prior to setting its R or S Bit. IO=0 selects an OUT endpoint, IO=1 selects an IN endpoint.

**Bit 2-0: EP** *Select endpoint*

The 8051 sets these bits to select an endpoint prior to setting its R or S Bit. Valid values are 0-7 to correspond to bulk endpoints IN0-IN7 and OUT0-OUT7.

**USBFRAMEL** **USB Frame Count Low** **7FD8**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **FC7** | **FC6** | **FC5** | **FC4** | **FC3** | **FC2** | **FC1** | **FC0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

**USBFRAMEH** **USB Frame Count High** **7FD9**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **FC10** | **FC9** | **FC8** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-102. USB Frame Count High/Low Registers*

Every millisecond the host sends a SOF token indicating “Start Of Frame,” along with an 11-bit incrementing frame count. The EZ-USB FX copies the frame count into these registers at every SOF. One use of the frame count is to respond to the USB SYNC_FRAME Request (*Chapter 9. "EZ-USB FX Endpoint Zero"*).

If the USB core detects a missing or garbledSOF, it generates an internal SOF and increments USBFRAMEL-USBRAMEH.

**FNADDR** **Function Address** **7FDB**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **0** | **FA6** | **FA5** | **FA4** | **FA3** | **FA2** | **FA1** | **FA0** |
| R | R | R | R | R | R | R | R |
| x | x | x | x | x | x | x | x |

*Figure 15-103. Function Address Register*

During the USB enumeration process, the host sends a device a unique 7-bit address, which the USB core copies into this register. There is normally no reason for the CPU to know its

USB device address because the USB Core automatically responds only to its assigned address.

*During ReNumeration™ the USB Core sets register to 0 to allow the EZ-USB FX chip to respond to the default address 0.*

**USBPAIR** **USB Endpoint Pairing** **7FDD†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **ISOSEND0** | **-** | **PR6OUT** | **PR4OUT** | **PR2OUT** | **PR6IN** | **PR4IN** | **PR2IN** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | x | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 15-104. USB Endpoint Pairing Register*

**Bit 7: ISOSEND0** *Isochronous Send Zero Length Data Packet*

The ISOSEND0 Bit is used when the EZ-USB FX chip receives an isochronous IN token while the IN FIFO is empty. If ISOSEND0=0 (the default value), the USB core does not respond to the IN token. If ISOSEND0=1, the USB core sends a zero-length data packet in response to the IN token. Which action to take depends on the overall system design. The ISOSEND0 Bit applies to all of the isochronous IN endpoints, IN8BUF through IN15BUF.

**Bit 5-3: PRnOUT** *Pair Bulk OUT Endpoints*

Set the endpoint pairing bits (PRxOUT) to “1” to enable double-buffering of the bulk OUT endpoint buffers. With double buffering enabled, the 8051 can operate on one buffer while another is being transferred over USB. The endpoint busy and interrupt request bits function identically, so the 8051 code requires no code modification to support double buffering.

When an endpoint is paired, the 8051 uses only the even-numbered endpoint of the pair. The 8051 should not use the paired odd endpoint’s IRQ, IEN, VALID bits or the buffer associated with the odd numbered endpoint.

**Bit 2-0: PRnIN** *Pair Bulk IN Endpoints*

Set the endpoint pairing bits (PRxIN) to “1” to enable double-buffering of the bulk IN endpoint buffers. With double buffering enabled, the 8051 can operate on one buffer while another is being transferred over USB.

When an endpoint is paired, the 8051 should access only the even-numbered endpoint of the pair. The 8051 should not use the IRQ, IEN, VALID bits or the buffer associated with the odd numbered endpoint.

**IN07VAL** **Endpoints 0-7 IN Valid Bits** **7FDE†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **IN7VAL** | **IN6VAL** | **IN5VAL** | **IN4VAL** | **IN3VAL** | **IN2VAL** | **IN1VAL** | **IN0VAL** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

**OUT07VAL** **Endpoints 0-7 OUT Valid Bits** **7FDF†**

**Read/write latency applies**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **OUT7VAL** | **OUT6VAL** | **OUT5VAL** | **OUT4VAL** | **OUT3VAL** | **OUT2VAL** | **OUT1VAL** | **OUT0VAL** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

*Figure 15-105. IN/OUT Valid Bits Register*

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

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

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

**Read/write latency applies**

| 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†**

**Read/write latency applies**

| 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 15-106. 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 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 (Table 5-1).

## 15.39 Fast Transfers

**FASTXFR** **Fast Transfer Control** **7FE2**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **FISO** | **FBLK** | **RPOL** | **RMOD1** | **RMOD0** | **WPOL** | **WMOD1** | **WMOD0** |
| R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| x | x | x | x | x | x | x | x |

*Figure 15-107. Fast Transfer Control Register*

The USB core provides a fast transfer mode designed for attaching external FIFOs to the isochronous and bulk endpoint buffers. The FASTXFR Register enables the modes for bulk and/or isochronous transfers, and selects the timing waveforms for the FRD# and FWR# signals.

**Bit 7:** **FISO** *Enable Fast ISO Transfers*

The 8051 sets FISO=1 to enable fast isochronous transfers for all16 isochronous endpoint FIFOs. When FISO=0, fast transfers are disabled for all 16 isochronous endpoints.

**Bit 6:** **FBLK** *Enable Fast BULK Transfers*

The 8051 sets FBLK=1 to enable fast bulk transfers using the Autopointer (see Section 15.40. *"SETUP Data"*) with BULK endpoints. When FBLK=0 fast transfers are disabled for BULK endpoints.

**Bit 5:** **RPOL** *FRD# Pulse Polarity*

The 8051 sets RPOL=0 for active-low FRD# pulses, and RPOL=1 for active high FRD# pulses.

**Bit 4-3:** **RMOD** *FRD# Pulse Mode*

These bits select the phasing and width of the FRD# pulse.

**Bit 2:** **WPOL** *FWR# Pulse Polarity*

The 8051 sets WPOL=0 for active-low FWR# pulses, and WPOL=1 for active high FWR# pulses.

**Bit 1-0:** **WMOD** *FWR# Pulse Mode*

These bits select the phasing and width of the FWR# pulse.

| 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 15-108. Auto Pointer Registers*

These registers control the EZ-USB FX *Autopointer*.

### 15.39.1 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 FX RAM. The 8051 can read these registers to determine the address of the next byte to be accessed via the AUTODATA Register.

### 15.39.2 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 FX Bulk Transfers"* and *Chapter 10. "EZ-USB FX Isochronous Transfers"* explain how to use the Autopointer for fast transfers to and from the EZ-USB FX endpoint buffers.

## 15.40 SETUP Data

| SETUPBUF | SETUP Data Buffer (8 Bytes) | 7FE8-7FEF |
|---|---|---|

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

*Figure 15-109. SETUP Data Buffer*

This buffer contains the 8 bytes of SETUP packet data from the most recently received CONTROL transfer.

The data in SETUPBUF is valid when the SUDAVIR (Setup Data Available Interrupt Request) Bit is set.

## 15.41 Isochronous FIFO Sizes

| OUTnADDR | ISO OUT Endpoint Start Address | 7FF0-7FF7* |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **A9** | **A8** | **A7** | **A6** | **A5** | **A4** | **0** | **0** |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| x | x | x | x | x | x | x | x |

| INnADDR | ISO IN Endpoint Start Address | 7FF8-7FFF* |
|---|---|---|

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| **A9** | **A8** | **A7** | **A6** | **A5** | **A4** | **0** | **0** |
| 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 15-17 for individual start address register addresses.

*Figure 15-110. SETUP Data Buffer*

**Table 15-17. Isochronous FIFO Start Address Registers**

| Address | Endpoint Start Address |
|---|---|
| 7FF0 | Endpoint 8 OUT Start Address |
| 7FF1 | Endpoint 9 OUT Start Address |
| 7FF2 | Endpoint 10 OUT Start Address |
| 7FF3 | Endpoint 11 OUT Start Address |
| 7FF4 | Endpoint 12 OUT Start Address |
| 7FF5 | Endpoint 13 OUT Start Address |
| 7FF6 | Endpoint 14 OUT Start Address |
| 7FF7 | Endpoint 15 OUT Start Address |
| 7FF8 | Endpoint 8 IN Start Address |
| 7FF9 | Endpoint 9 IN Start Address |
| 7FFA | Endpoint 10 IN Start Address |
| 7FFB | Endpoint 11 IN Start Address |
| 7FFC | Endpoint 12 IN Start Address |
| 7FFD | Endpoint 13 IN Start Address |
| 7FFE | Endpoint 14 IN Start Address |
| 7FFF | Endpoint 15 IN Start Address |

EZ-USB FX Isochronous endpoints use a pool of 1,024 double-buffered FIFO bytes. The 1,024 FIFO bytes can be divided between any or all of the isochronous endpoints. The 8051 sets isochronous endpoint FIFO sizes by writing starting addresses to these registers, starting with address 0. Address bits A3-A0 are internally set to zero, so the minimum FIFO size is 16 bytes.

# Chapter 16. 8051 Introduction

## 16.1 Introduction

The EZ-USB FX contains an 8051 core that is binary-compatible with the industry standard 8051 instruction set.

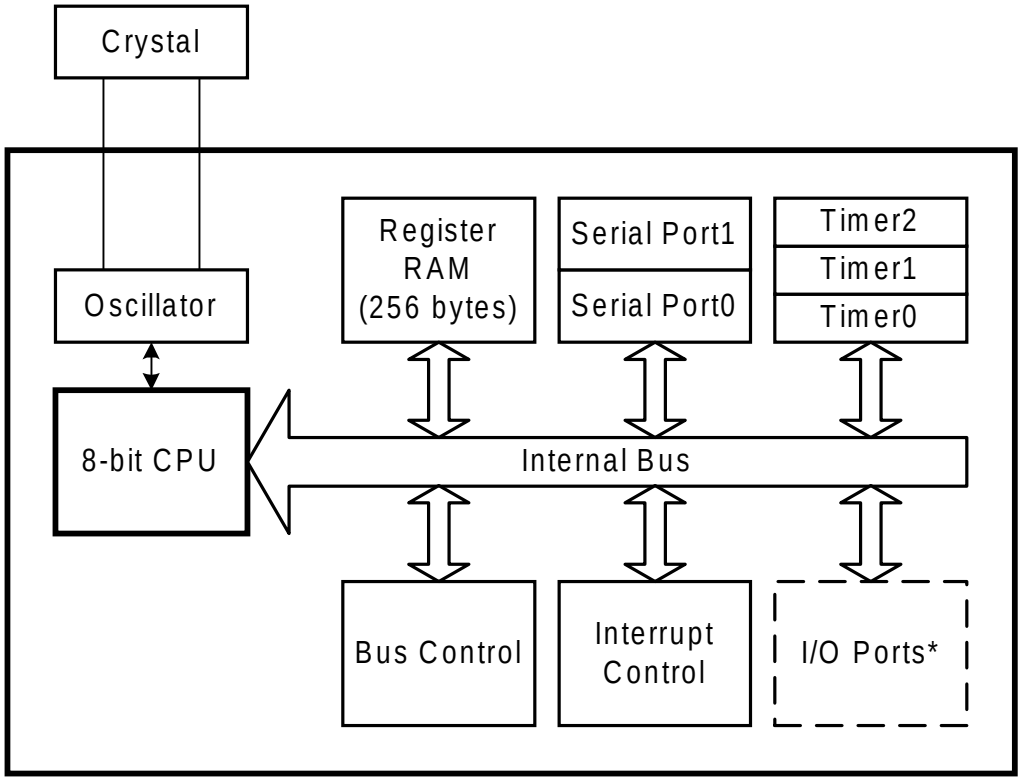


*Figure 16-1. 8051 Features*

This chapter provides an overview of the 8051 core features. The topics are:

- New 8051 Features
- Performance Overview
- Software Compatibility
- 803x/805x Feature Comparison
- 8051/DS80C320 Differences.

## 16.2 8051 Features

The 8051 core provides the following design features and enhancements to the standard 8051 micro-controller:

- Compatible with industry standard 803x/805x:
  - Standard 8051 instruction set
  - Two full-duplex serial ports
  - Three timers
- High-speed architecture:
  - 4 clocks/instruction cycle
  - 2.5X average improvement in instruction execution time over the standard 8051
  - Wasted bus cycles eliminated
  - Dual data pointers
- 256 Bytes internal data RAM
- High-speed external memory interface with 16-bit address bus
- Variable length `MOVX` to access fast/slow RAM peripherals
- Supports industry standard compilers, assemblers, emulators, and ROM monitors

## 16.3 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 16-2*). 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. Table 16-1 catalogs the speed improvements.

**Table 16-1. 8051/Standard 8051 Speed Comparison**

| 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. | |

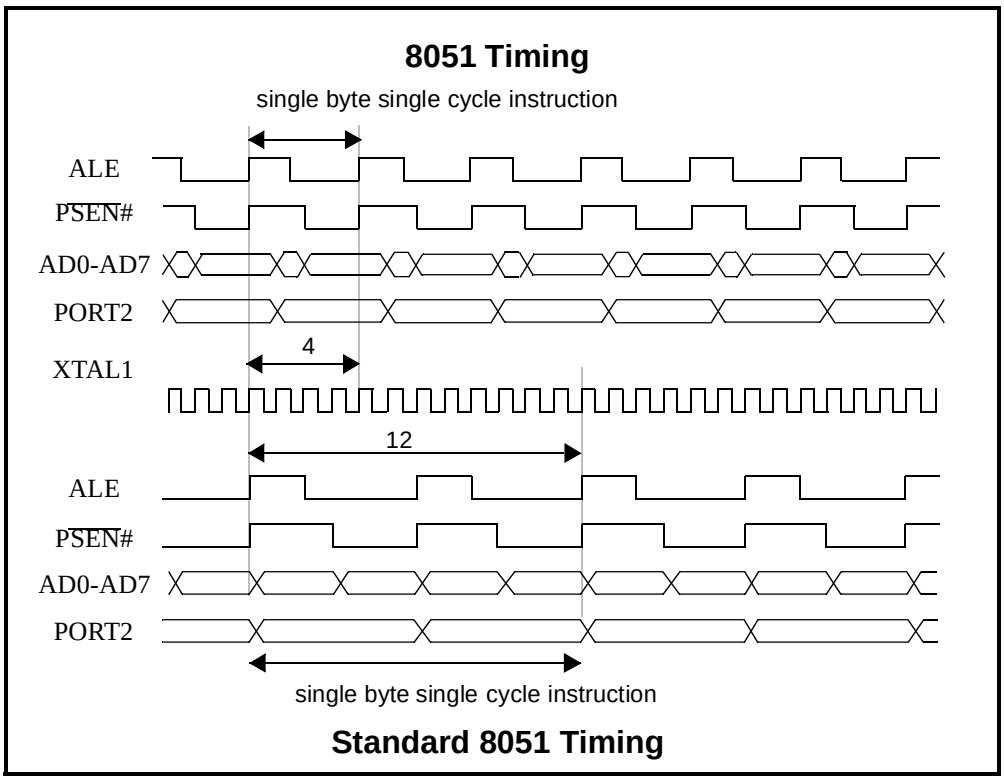


*Figure 16-2. 8051/Standard 8051 Timing Comparison*

## 16.4 Software Compatibility

The 8051 core is object code compatible with the industry standard 8051 micro-controller. That is, object code compiled with an industry standard 8051 compiler or assembler executes on the 8051 core and is functionally equivalent. However, because the 8051 core uses a different instruction timing than the standard 8051, existing code with timing loops may require modification.

The “Instruction Set” in Table 16-2 lists the number of instruction cycles required to perform each instruction on the 8051 core. The 8051 instruction cycle timing and number of instruction cycles required for each instruction are compatible with the Dallas Semiconductor DS80C320.

## 16.5 803x/805x Feature Comparison

Table 16-2 provides a feature-by-feature comparison of the 8051 core and several common 803x/805x configurations.

**Table 16-2. Features of 8051 Core & Common 803x/805x Configurations**

| Feature | Intel | | | | Dallas DS80C320 | Anchor 8051 |
|---|---|---|---|---|---|---|
| | 8031 | 8051 | 80C32 | 80C52 | | |
| Clocks per instruction cycle | 12 | 12 | 12 | 12 | 4 | 4 |
| Program / Data Memory | - | 4 KB ROM | - | 8 KB ROM | - | 8 K RAM |
| Internal RAM | 128 bytes | 128 bytes | 256 bytes | 256 bytes | 256 bytes | 256 bytes |
| Data Pointers | 1 | 1 | 1 | 1 | 2 | 2 |
| Serial Ports | 1 | 1 | 1 | 1 | 2 | 2 |
| 16-bit Timers | 2 | 2 | 3 | 3 | 3 | 3 |
| Interrupt sources (total of int. and ext.) | 5 | 5 | 6 | 6 | 13 | 13 |
| Stretch memory cycles | no | no | no | no | yes | yes |

## 16.6 8051 Core/DS80C320 Differences

The 8051 core is similar to the DS80C320 in terms of hardware features and instruction cycle timing. However, there are some important differences between the 8051 core and the DS80C320.

### 16.6.1 Serial Ports

The 8051 core does not implement serial port framing error detection and does not implement slave address comparison for multiprocessor communications. Therefore, the 8051 core also does not implement the following SFRs: SADDR0, SADDR1, SADEN0, and SADEN1.

### 16.6.2 Timer 2

The 8051 core does not implement Timer 2 downcounting mode or the downcount enable bit (TMOD2, Bit 0). Also, the 8051 core does not implement Timer 2 output enable (T2OE) bit (TMOD2, Bit 1). Therefore, the TMOD2 SFR is also not implemented in the 8051 core.

Also, the 8051 core Timer 2 overflow output is active for one clock cycle. In the DS80C320, the Timer 2 overflow output is a square wave with a 50% duty cycle.

*It is possible to float the T2OUT pin by setting OEB.7=0 and PORTBCFG.7=0. This selects the PORTB (not T2OUT) signal, and turns off its output buffer.*

### 16.6.3 Timed Access Protection

The 8051 core does not implement timed access protection and therefore, does not implement the TA SFR.

### 16.6.4 Watchdog Timer

The EZ-USB FX/8051 does not implement a watchdog timer. It also does not implement I/O ports 0-3. Instead, it uses ports A-E.

# Chapter 17. 8051 Architectural Overview

## 17.1 Introduction

This chapter provides a technical overview and description of the 8051 core architecture.

### 17.1.1 Memory Organization

Memory organization in the 8051 core is similar to that of the industry standard 8051. There are three distinct memory areas: registers, program memory, and data memory.

#### 17.1.1.1 Registers

Register memory is implemented inside the 8051 core. The 8051 accesses registers in two regions using direct addressing, providing the fastest available 8051 data access. The two directly addressable regions are 128 general purpose registers at addresses 00-7F, and 128 bytes of Special Function Registers (SFRs) at 80-FF. The SFR address space, which is not fully populated, contains 8051 control and status registers, plus added EZ-USB FX control and status registers.

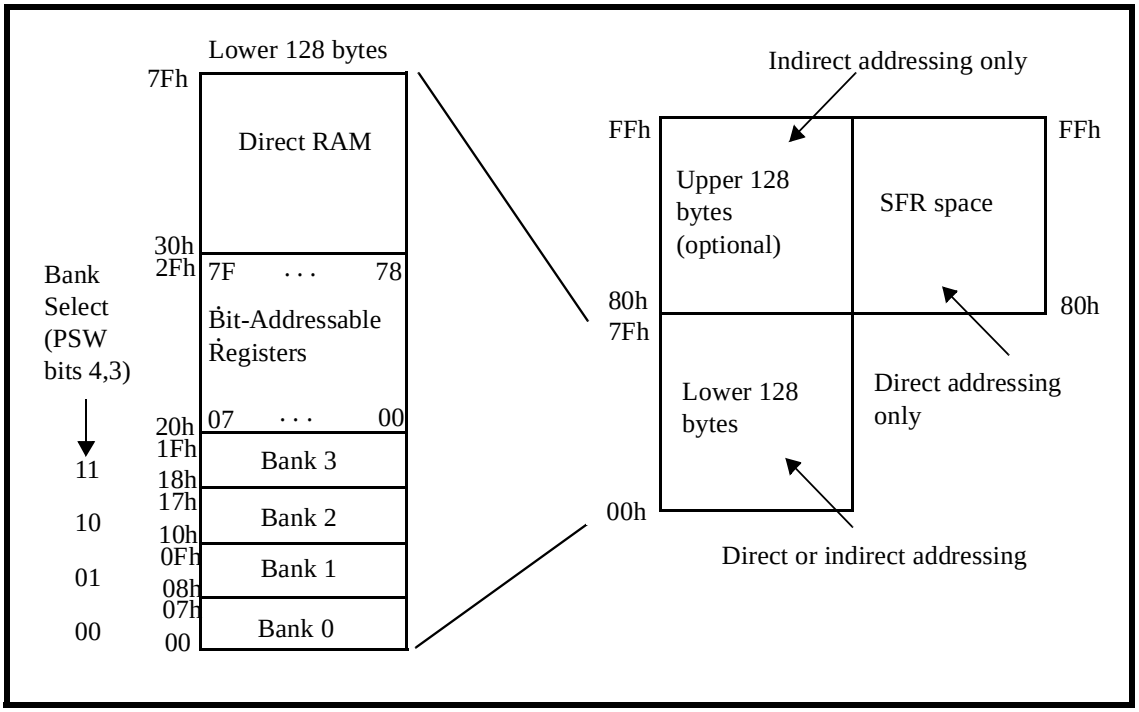


*Figure 17-1. Internal RAM Organization*

Some examples of direct addressing are:

- MOV A,22H ; load accumulator from register at address 22
- MOV A,IOE ; read the EZ-USB FX PORTE pins (added SFR)
- MOV IOD,A ; write the PORTD Bits (added SFR)

An additional 128 registers overlap the SFRs at addresses 80-FF. The 8051 keeps these separate from the SFRs by using a different addressing mode, 8-bit indirect, to access them. For example, to read the register at location 90(hex):

- MOV R0,#90H ;point to register RAM at 90(hex)
- MOV A,@R0 ;read it using 8-bit indirect addressing

The 8051 uses two registers, R0 and R1, to hold the 8-bit index. This addressing mode may also access register memory from 0-127, although it is faster and more efficient to use the direct addressing available in this lower region.

Since the 8051 stack is internally accessed using indirect addressing, it is a good idea to put the stack in the upper 128 bytes of register memory, which is addressable using indirect addressing only. This frees the lower 128 register bytes for use by the more efficient direct addressing.

### 17.1.1.2 Program Memory

The 8051 has separate address spaces for program and data memory. Program memory can only be read, not written. The read strobe for program memory is PSEN (Program Store Enable). The 8051 generates PSEN strobes for two conditions, instruction fetches and the MOVC (move code memory into the accumulator) instruction.

### 17.1.1.3 Data Memory

Data memory occupies a separate address space from program memory. Data memory can be read or written, using the RD and WR strobes. Up to 64 KB of data memory can be added to the EZ-USB FX versions that bring out the 8051 address and data bus pins. As the next section explains, a portion of this *external* data memory is actually implemented inside the FX chip.

### 17.1.1.4 EZ-USB FX Program/Data Memory

The EZ-USB FX family contains internal RAM, which in most systems provides all the memory for a single-chip solution. Therefore, this internal RAM must serve both as 8051 program and data memory. To accomplish this, the 8051 reads internal RAM using the logical OR of the PSEN and RD strobes. It is the responsibility of the system designer to ensure that the program and data memory spaces do not overlap. This is done using linker directives that place the code and data modules.

It is possible to add external program and data memory to the EZ-USB FX parts that provide the 8051 address and data bus pins. To avoid conflict with the internal combined program/data memory, the EZ-USB FX logic gates the memory strobes to be inactive when the 8051 accesses internal memory. These strobes include the RD#, WR#, CS#, and OE# pins. Because of this internal gating, a 64-KB memory (data and/or program) can be added without requiring external logic to inhibit access to the bottom 8 KB that are inside the FX part. Note that the PSEN and RD signals are available on separate pins, so the program and data spaces are not combined as they are inside the FX part.

The EA (external access) pin allows *all* external memory to be program memory. When EA is tied high, the 8051 reads the internal RAM using only the RD strobe—the combining of RD and PSEN is disabled. With EA=1, the internal RAM becomes data memory only, and program memory starts at 0000 in external memory. The other effect of tying the EA pin high is that the 8051 powers up running (not in RESET), ready to run the external code.

### 17.1.1.5 Accessing Data Memory

The 8051 reads and writes data memory using the MOVX instruction. Either an 8-bit or a 16-bit index (address pointer) can be used. 8-bit addressing uses either R0 or R1 to supply the lower address byte, and the MPAGE Register (SFR address 92H) to supply the high address byte. 16-bit addressing uses the 16-bit data pointer (DPTR) to supply the full address.

EZ-USB FX registers exist in the upper portion of internal memory. The 8051 accesses them using the MOVX instruction. A limited set of EZ-USB FX registers are in the SFR address space to provide fastest possible access.

### 17.1.2 Instruction Set

All 8051 instructions are binary code compatible and perform the same functions as they do with the industry standard 8051. The effects of these instructions on bits, flags, and other status functions is identical to the industry standard 8051. However, the timing of the instructions is different, both in terms of number of clock cycles per instruction cycle and timing within the instruction cycle.

Table 17-2 lists the 8051 instruction set and the number of instruction cycles required to complete each instruction. Table 17-1 defines the symbols and mnemonics used in Table 17-2.

**Table 17-1. Legend for Instruction Set Table**

| Symbol | Function |
|---|---|
| A | Accumulator |
| Rn | Register R7–R0 |
| direct | Internal register address |
| @Ri | Internal register pointed to by R0 or R1 |
| rel | Two's complement offset byte |
| bit | Direct bit address |
| #data | 8-bit constant |
| #data 16 | 16-bit constant |
| addr 16 | 16-bit destination address |
| addr 11 | 11-bit destination address |

## Table 17-2. 8051 Instruction Set

| Mnemonic | Description | Byte | Instr. Cycles | Hex Code |
|---|---|---|---|---|
| **Arithmetic** | | | | |
| ADD A, Rn | Add register to A | 1 | 1 | 28-2F |
| ADD A, direct | Add direct byte to A | 2 | 2 | 25 |
| ADD A, @Ri | Add data memory to A | 1 | 1 | 26-27 |
| ADDC A, #data | Add immediate to A | 2 | 2 | 24 |
| ADDC A, Rn | Add register to A with carry | 1 | 1 | 38-3F |
| ADDC A, direct | Add direct byte to A with carry | 2 | 2 | 35 |
| ADDC A, @Ri | Add data memory to A with carry | 1 | 1 | 36-37 |
| ADDC A, #data | Add immediate to A with carry | 2 | 2 | 34 |
| SUBB A, Rn | Subtract register from A with borrow | 1 | 1 | 98-9F |
| SUBB A, direct | Subtract direct byte from A with borrow | 2 | 2 | 95 |
| SUBB A, @Ri | Subtract data memory from A with borrow | 1 | 1 | 96-97 |
| SUBB A, #data | Subtract immediate from A with borrow | 2 | 2 | 94 |
| INC A | increment A | 1 | 1 | 04 |
| INC Rn | Increment register | 1 | 1 | 08-0F |
| INC direct | Increment direct byte | 2 | 2 | 05 |
| INC @ Ri | Increment data memory | 1 | 1 | 06-07 |
| DEC A | Decrement A | 1 | 1 | 14 |
| DEC Rn | Decrement Register | 1 | 1 | 18-1F |
| DEC direct | Decrement direct byte | 2 | 2 | 15 |
| DEC @Ri | Decrement data memory | 1 | 1 | 16-17 |
| INC DPTR | Increment data pointer | 1 | 3 | A3 |
| MUL AB | Multiply A by B | 1 | 5 | A4 |
| DIV AB | Divide A by B | 1 | 5 | 84 |
| DA A | Decimal adjust A | 1 | 1 | D4 |
| **Logical** | | | | |
| ANL, Rn | AND register to A | 1 | 1 | 58-5F |
| ANL A, direct | AND direct byte to A | 2 | 2 | 55 |
| ANL A, @Ri | AND data memory to A | 1 | 1 | 56-57 |
| ANL A, #data | AND immediate to A | 2 | 2 | 54 |
| ANL direct, A | AND A to direct byte | 2 | 2 | 52 |
| ANL direct, #data | AND immediate data to direct byte | 3 | 3 | 53 |
| ORL A, Rn | OR register to A | 1 | 1 | 48-4F |
| ORL A, direct | OR direct byte to A | 2 | 2 | 45 |
| ORL A, @Ri | OR data memory to A | 1 | 1 | 46-47 |
| ORL A, #data | OR immediate to A | 2 | 2 | 44 |
| ORL direct, A | OR A to direct byte | 2 | 2 | 42 |

**Table 17-2. 8051 Instruction Set**

| Mnemonic | Description | Byte | Instr. Cycles | Hex Code |
|---|---|---|---|---|
| ORL direct, #data | OR immediate data to direct byte | 3 | 3 | 43 |
| XORL A, Rn | Exclusive-OR register to A | 1 | 1 | 68-6F |
| XORL A, direct | Exclusive-OR direct byte to A | 2 | 2 | 65 |
| XORL A, @Ri | Exclusive-OR data memory to A | 1 | 1 | 66-67 |
| XORL A, #data | Exclusive-OR immediate to A | 2 | 2 | 64 |
| XORL direct, A | Exclusive-OR A to direct byte | 2 | 2 | 62 |
| XORL direct, #data | Exclusive-OR immediate data to direct byte | 3 | 3 | 63 |
| CLR A | Clear A | 1 | 1 | E4 |
| CPL A | Complement A | 1 | 1 | F4 |
| SWAP A | Swap nibbles of a | 1 | 1 | C4 |
| RL A | Rotate A left | 1 | 1 | 23 |
| RLC A | Rotate A left through carry | 1 | 1 | 33 |
| RRA | Rotate A right | 1 | 1 | 03 |
| RRC A | Rotate A right through carry | 1 | 1 | 13 |
| **Data Transfer** | | | | |
| MOV A, Rn | Move register to A | 1 | 1 | E8-EF |
| MOV A, direct | Move direct byte to A | 2 | 2 | E5 |
| MOV A, @Ri | Move data memory to A | 1 | 1 | E6-E7 |
| MOV A, #data | Move immediate to A | 2 | 2 | 74 |
| MOV Rn, A | Move A to register | 1 | 1 | F8-FF |
| MOV Rn, direct | Move direct byte to register | 2 | 2 | A8-AF |
| MOV Rn, #data | Move immediate to register | 2 | 2 | 78-7F |
| MOV direct, A | Move A to direct byte | 2 | 2 | F5 |
| MOV direct, Rn | Move register to direct byte | 2 | 2 | 88-8F |
| MOV direct, direct | Move direct byte to direct byte | 3 | 3 | 85 |
| MOV direct, @Ri | Move data memory to direct byte | 2 | 2 | 86-87 |
| MOV direct, #data | Move immediate to direct byte | 3 | 3 | 75 |
| MOV @Ri, A | MOV A to data memory | 1 | 1 | F6-F7 |
| MOV @Ri, direct | Move direct byte to data memory | 2 | 2 | A6-A7 |
| MOV @Ri, #data | Move immediate to data memory | 2 | 2 | 76-77 |
| MOV DPTR, #data | Move immediate to data pointer | 3 | 3 | 90 |
| MOVC A, @A+DPTR | Move code byte relative DPTR to A | 1 | 3 | 93 |
| MOVC A, @A+PC | Move code byte relative PC to A | 1 | 3 | 83 |
| MOVX A, @Ri | Move external data (A8) to A | 1 | 2-9* | E2-E3 |
| MOVX A, @DPTR | Move external data (A16) to A | 1 | 2-9* | E0 |
| MOVX @Ri, A | Move A to external data (A8) | 1 | 2-9* | F2-F3 |
| MOVX @DPTR, A | Move A to external data (A16) | 1 | 2-9* | F0 |
| PUSH direct | Push direct byte onto stack | 2 | 2 | C0 |

## Table 17-2. 8051 Instruction Set

| Mnemonic | Description | Byte | Instr. Cycles | Hex Code |
|---|---|---|---|---|
| POP direct | Pop direct byte from stack | 2 | 2 | D0 |
| XCH A, Rn | Exchange A and register | 1 | 1 | C8-CF |
| XCH A, direct | Exchange A and direct byte | 2 | 2 | C5 |
| XCH A, @Ri | Exchange A and data memory | 1 | 1 | C6-C7 |
| XCHD A, @Ri | Exchange A and data memory nibble | 1 | 1 | D6-D7 |
| ** Number of cycles is user-selectable. See Section 17.1.5. "Stretch Memory Cycles (Wait States)".* | | | | |
| **Boolean** | | | | |
| CLR C | Clear carry | 1 | 1 | C3 |
| CLR bit | Clear direct bit | 2 | 2 | C2 |
| SETB C | Set carry | 1 | 1 | D3 |
| SETB bit | Set direct bit | 2 | 2 | D2 |
| CPL C | Complement carry | 1 | 1 | B3 |
| CPL bit | Complement direct bit | 2 | 2 | B2 |
| ANL C, bit | AND direct bit to carry | 2 | 2 | 82 |
| ANL C, /bit | AND direct bit inverse to carry | 2 | 2 | B0 |
| ORL C, bit | OR direct bit to carry | 2 | 2 | 72 |
| ORL C, /bit | OR direct bit inverse to carry | 2 | 2 | A0 |
| MOV C, bit | Move direct bit to carry | 2 | 2 | A2 |
| MOV bit, C | Move carry to direct bit | 2 | 2 | 92 |
| Branching | | | | |
| ACALL addr11 | Absolute call to subroutine | 2 | 3 | 11-F1 |
| LCALL addr 16 | Long call to subroutine | 3 | 4 | 12 |
| RET | Return from subroutine | 1 | 4 | 22 |
| RETI | Return from interrupt | 1 | 4 | 32 |
| AJMP addr 11 | Absolute jump unconditional | 2 | 3 | 01-E1 |
| LJMP addr 16 | Long jump unconditional | 3 | 4 | 02 |
| SJMP rel | Short jump (relative address) | 2 | 3 | 80 |
| JC rel | Jump on carry = 1 | 2 | 3 | 40 |
| JNC rel | Jump on carry = 0 | 2 | 3 | 50 |
| JB bit, rel | Jump on direct bit = 1 | 3 | 4 | 20 |
| 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 |

**Table 17-2. 8051 Instruction Set**

| Mnemonic | Description | Byte | Instr. Cycles | Hex Code |
|---|---|---|---|---|
| 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. | | | | |

### *17.1.3 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 17-2 to calculate the timing of software loops. The bytes column indicates the number of memory 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.

### 17.1.4 CPU Timing

As previously stated, an 8051 core instruction cycle consists of 4 *CLKOUT* cycles. Each *CLKOUT* cycle forms a CPU cycle. Therefore, an instruction cycle consists of 4 CPU cycles: C1, C2, C3, and C4, as illustrated in *Figure 17-2*. 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 *CLKOUT* at the end of C3.

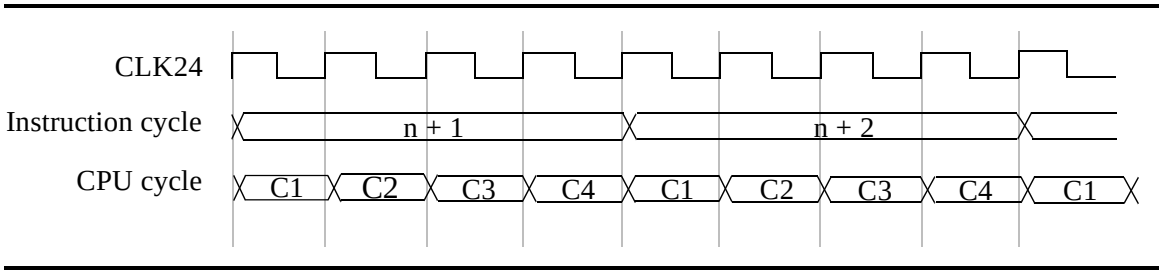


*Figure 17-2. CPU Timing for Single-Cycle Instruction*

### 17.1.5 Stretch Memory Cycles (Wait States)

The stretch memory cycle feature enables application software to adjust the speed of *data memory* (not code 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.

By default, the stretch value resets to one (three cycle MOVX). For full-speed data memory access, the software must set the stretch value to zero. The stretch value affects only data memory access (not program memory).

The stretch value affects the width of the read/write strobe and all related timing. Using a higher stretch value results in a wider read/write strobe, which allows the memory or peripheral more time to respond.

Table 17-3 lists the data memory access speeds for stretch values zero through seven. MD2–0 are the three LSBs of the Clock Control Register (CKCON.2–0).

**Table 17-3. Data Memory Stretch Values**

| MD2 | MD1 | MD0 | Memory Cycles | Read/Write Strobe Width (Clocks) | Strobe Width @ 24MHz | Strobe Width @ 48MHz |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 2 | 83.3 ns | 41.65 ns |
| 0 | 0 | 1 | 3 (default) | 4 | 166.7 ns | 83.35 ns |
| 0 | 1 | 0 | 4 | 8 | 333.3 ns | 166.66 ns |
| 0 | 1 | 1 | 5 | 12 | 500 ns | 250 ns |
| 1 | 0 | 0 | 6 | 16 | 666.7 ns | 333.35 ns |
| 1 | 0 | 1 | 7 | 20 | 833.3 ns | 416.65 ns |
| 1 | 1 | 0 | 8 | 24 | 1000 ns | 500 ns |
| 1 | 1 | 1 | 9 | 28 | 1166.7 ns | 583.35 ns |

### *17.1.6 Dual Data Pointers*

The 8051core employs dual data pointers to accelerate data memory block moves. The standard 8051 data pointer (DPTR) is a 16-bit value used to address external data RAM or peripherals. The 8051 maintains the standard data pointer as DPTR0 at SFR locations 82h (DPL0) and 83h (DPH0). It is not necessary to modify existing code to use DPTR0.

The 8051 core adds a second data pointer (DPTR1) at SFR locations 84h (DPL1) and 85h (DPH1). The SEL Bit in the DPTR Select Register, DPS (SFR 86h), selects the active pointer. When SEL = 0, instructions that use the DPTR will use DPL0 and DPH0. When SEL = 1, instructions that use the DPTR will use DPL1 and DPH1. SEL is the bit 0 of SFR location 86h. No other bits of SFR location 86h are used.

All DPTR-related instructions use the data pointer selected by the SEL Bit. To switch the active pointer, toggle the SEL Bit. The fastest way to do so is to use the increment instruction (`INC DPS`). This requires only one instruction to switch from a source address to a destination address, saving application code from having to save source and destination addresses when doing a block move.

Using dual data pointers provides significantly increased efficiency when moving large blocks of data.

The SFR locations related to the dual data pointers are:

| | | |
|---|---|---|
| 82h | DPL0 | DPTR0 low byte |
| 83h | DPH0 | DPTR0 high byte |
| 84h | DPL1 | DPTR1 low byte |
| 85h | DPH1 | DPTR1 high byte |
| 86h | DPS | DPTR Select (Bit 0) |

### *17.1.7 Special Function Registers*

The Special Function Registers (SFRs) control several of the features of the 8051 and EZ-USB FX. Most of the 8051 core SFRs are identical to the standard 8051 SFRs. However, there are additional SFRs that control features that are not available in the standard 8051, plus some EZ-USB FX features.

Table 17-4 lists the 8051 core SFRs and indicates which SFRs are not included in the standard 8051 SFR space. See Section 4.12. *"SFR Addressing"* for the EZ-USB FX added SFRs.

In Table 17-5, SFR Bit positions that contain a 0 or a 1 cannot be written to and, when read, always return the value shown (0 or 1). SFR Bit positions that contain "-" are available but not used. Table 17-5 lists the reset values for the SFRs.

The following SFRs are related to CPU operation and program execution:

| | | |
|---|---|---|
| 81h | SP | Stack Pointer |
| D0h | PSW | Program Status Word () |
| E0h | ACC | Accumulator Register |
| F0h | B | B Register |

Table 17-6 lists the functions of the bits in the PSW SFR. Detailed descriptions of the remaining SFRs appear with the associated hardware descriptions in *Chapter 4. "EZ-USB FX Input/Output"* of this manual.

**Table 17-4. Special Function Registers**

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Addr |
|---|---|---|---|---|---|---|---|---|---|
| SP | | | | | | | | | 81h |
| DPL0 | | | | | | | | | 82h |
| DPH0 | | | | | | | | | 83h |
| DPL1(1) | | | | | | | | | 84h |
| DPH1(1) | | | | | | | | | 85h |
| DPS(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SEL | 86h |
| PCON | SMOD0 | - | 1 | 1 | GF1 | GF0 | STOP | IDLE | 87h |
| TCON | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 | 88h |
| TMOD | GATE | C/T | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 | 89h |
| TL0 | | | | | | | | | 8Ah |
| TL1 | | | | | | | | | 8Bh |
| TH0 | | | | | | | | | 8Ch |
| TH1 | | | | | | | | | 8Dh |
| CKCON(1) | - | - | T2M | T1M | T0M | MD2 | MD1 | MD0 | 8Eh |
| SPC_FNC(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | WRS | 8Fh |
| EXIF(1) | IE5 | IE4 | I2CINT | USBINT | 1 | 0 | 0 | 0 | 91h |
| MPAGE(1) | | | | | | | | | 92h |
| SCON0 | SM0_0 | SM1_0 | SM2_0 | REN_0 | TB8_0 | RB8_0 | TI_0 | RI_0 | 98h |
| SBUF0 | | | | | | | | | 99h |
| IE | EA | ES1 | ET2 | ES0 | ET1 | EX1 | ET0 | EX0 | A8h |
| IP | 1 | PS1 | PT2 | PS0 | PT1 | PX1 | PT0 | PX0 | B8h |
| SCON1(1) | SM0_1 | SM1_1 | SM2_1 | REN_1 | TB8_1 | RB8_1 | TI_1 | RI_1 | C0h |
| SBUF1(1) | | | | | | | | | C1h |
| T2CON | TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/$\overline{T2}$ | CP/$\overline{RL2}$ | C8h |
| RCAP2L | | | | | | | | | CAh |
| RCAP2H | | | | | | | | | CBh |
| TL2 | | | | | | | | | CCh |
| TH2 | | | | | | | | | CDh |
| PSW | CY | AC | F0 | RS1 | RS0 | OV | F1 | P | D0h |
| EICON(1) | SMOD1 | 1 | ERESI | RESI | INT6 | 0 | 0 | 0 | D8h |
| ACC | | | | | | | | | E0H |
| EIE(1) | 1 | 1 | 1 | EWDI | EX5 | EX4 | EI2C | EUSB | E8h |
| B | | | | | | | | | F0h |
| EIP(1) | 1 | 1 | 1 | PX6 | PX5 | PX4 | PI2C | PUSB | F8h |

(1) Not part of standard 8051 architecture.

## Table 17-5. Special Function Register Reset Values

| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Addr |
|---|---|---|---|---|---|---|---|---|---|
| SP | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 81h |
| DPL0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 82h |
| DPH0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 83h |
| DPL1(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 84h |
| DPH1(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 85h |
| DPS(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 86h |
| PCON | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 87h |
| TCON | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 88h |
| TMOD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 89h |
| TL0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8Ah |
| TL1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8Bh |
| TH0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8Ch |
| TH1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8Dh |
| CKCON(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 8Eh |
| SPC_FNC(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8Fh |
| EXIF(1) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 91h |
| MPAGE(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 92h |
| SCON0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 98h |
| SBUF0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 99h |
| IE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A8h |
| IP | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | B8h |
| SCON1(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | C0h |
| SBUF1(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | C1h |
| T2CON | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | C8h |
| RCAP2L | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CAh |
| RCAP2H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CBh |
| TL2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CCh |
| TH2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CDh |
| PSW | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D0h |
| EICON(1) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | D8h |
| ACC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E0H |
| EIE(1) | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | E8h |
| B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | F0h |
| EIP(1) | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | F8h |

(1) Not part of standard 8051 architecture.

**Table 17-6. PSW Register - SFR D0h**

| Bit | Function |
|---|---|
| PSW.7 | **CY** - Carry flag. This is the **unsigned** carry bit. The CY flag is set when an arithmetic operation results in a carry from bit 7 to bit 8, and cleared otherwise. In other words, it acts as a virtual bit 8. The CY flag is cleared on multiplication and division. |
| PSW.6 | **AC** - Auxiliary carry flag. Set to 1 when the last arithmetic operation resulted in a carry into (during addition) or borrow from (during subtraction) the high order nibble, otherwise cleared to 0 by all arithmetic operations. |
| PSW.5 | **F0** - User flag 0. Bit-addressable, general purpose flag for software control. |
| PSW.4 | **RS1** - Register bank select bit 1. used with RS0 to select a register bank in internal RAM. |
| PSW.3 | **RS0** - Register bank select bit 0, decoded as:<br>RS1RS0 Banks Selected<br>00 Register bank 0, addresses 00h-07h<br>01 Register bank 1, addresses 08h-0Fh<br>10 Register bank 2, addresses 10h-17h<br>11 Register bank 3, addresses 18h-1Fh |
| PSW.2 | **OV** - Overflow flag. This is the **signed** carry bit. The OV flag is set when a positive sum exceeds 7fh, or a negative sum (in two's compliment notation) exceeds 80h. On a multiply, if OV = 1, the result of the multiply is greater than FFh. On a divide, OV = 1 on a divide by 0. |
| PSW.1 | **F1** - User flag 1. Bit-addressable, general purpose flag for software control. |
| PSW.0 | **P** - Parity flag. Set to 1 when the modulo-2 sum of the 8 bits in the accumulator is 1 (odd parity), cleared to 0 on even parity. |

# Chapter 18. 8051 Hardware Description

## 18.1 Introduction

This chapter provides technical data about the 8051 core hardware operation and timing. The topics are:

- Timers/Counters
- Serial Interface
- Interrupts
- 8051 Reset
- Power Saving Modes.

## 18.2 Timers/Counters

The 8051 core includes three timer/counters (Timer 0, Timer 1, and Timer 2). Each timer/counter can operate as either a timer with a clock rate based on the *CLKOUT* pin or as an event counter clocked by the *T0* pin (Timer 0), *T1* pin (Timer 1), or the *T2* pin (Timer 2).

Each timer/counter consists of a 16-bit register that is accessible to software as two SFRs:

- Timer 0 — TL0 and TH0
- Timer 1 — TL1 and TH1
- Timer 2 — TL2 and TH2.

### 18.2.1 803x/805x Compatibility

The implementation of the timers/counters is similar to that of the Dallas Semiconductor DS80C320. Table 18-7 summarizes the differences in timer/counter implementation between the Intel 8051, the Dallas Semiconductor DS80C320, and the 8051 core.

**Table 18-7. Timer/Counter Implementation Comparison**

| Feature | Intel 8051 | Dallas DS80C320 | 8051 |
|---|---|---|---|
| Number of timers | 2 | 3 | 3 |
| Timer 0/1 overflow available as output signals | not implemented | not implemented | T0OUT, T1OUT (one CLKOUT pulse) |
| Timer 2 output enable | n/a | implemented | not directly implemented |
| Timer 2 downcount enable | n/a | implemented | not implemented |
| Timer 2 overflow available as output signal | n/a | implemented | T2OUT (one CLKOUT pulse) |

### 18.2.2 Timers 0 and 1

Timers 0 and 1 each operate in four modes, as controlled through the TMOD SFR (Table 18-8) and the TCON SFR (Table 18-9). The four modes are:

- 13-bit timer/counter (mode 0)
- 16-bit timer/counter (mode 1)
- 8-bit counter with auto-reload (mode 2)
- Two 8-bit counters (mode 3, Timer 0 only)

#### 18.2.2.1 Mode 0

Mode 0 operation, illustrated in *Figure 18-3*, is the same for Timer 0 and Timer 1. In mode 0, the timer is configured as a 13-bit counter that uses bits 0-4 of TL0 (or TL1) and all 8 bits of TH0 (or TH1). The timer enable bit (TR0/TR1) in the TCON SFR starts the timer. The C/$\overline{T}$ Bit selects the timer/counter clock source, CLKOUT or the T0/T1 pins.

The timer counts transitions from the selected source as long as the GATE Bit is 0, or the GATE Bit is 1 and the corresponding interrupt pin (INT0# or INT1#) is 1.

When the 13-bit count increments from 1FFFh (all ones), the counter rolls over to all zeros, the TF0 (or TF1) Bit is set in the TCON SFR, and the T0OUT (or T1OUT) pin goes high for one clock cycle.

The upper 3 bits of TL0 (or TL1) are indeterminate in mode 0 and must be masked when the software evaluates the register.

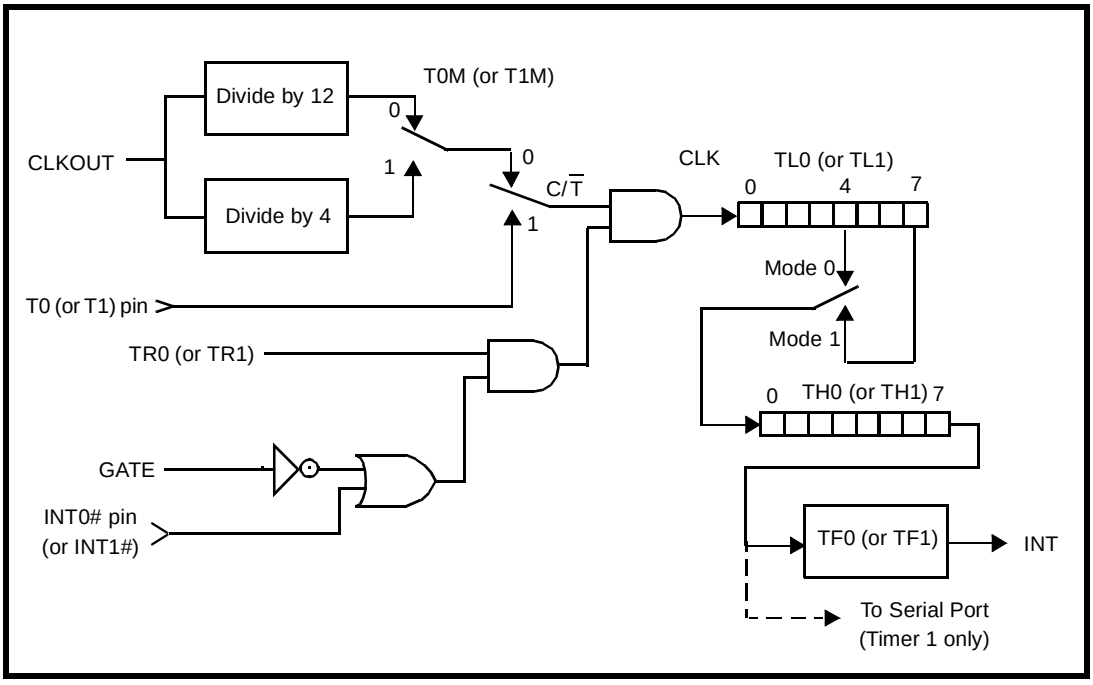


*Figure 18-3. Timer 0/1 - Modes 0 and 1*

### 18.2.2.2 Mode 1

Mode 1 operation is the same for Timer 0 and Timer 1. In mode 1, the timer is configured as a 16-bit counter. As illustrated in *Figure 18-3*, all 8 bits of the LSB Register (TL0 or TL1) are used. The counter rolls over to all zeros when the count increments from FFFFh. Otherwise, mode 1 operation is the same as mode 0.

**Table 18-8. TMOD Register — SFR 89h**

| Bit | Function |
|---|---|
| TMOD.7 | **GATE** - Timer 1 gate control. When GATE = 1, Timer 1 will clock only when INT1# = 1 and TR1 (TCON.6) = 1. When GATE = 0, Timer 1 will clock only when TR1 = 1, regardless of the state of INT1#. |
| TMOD.6 | **C/$\overline{T}$** - Counter/Timer select. When C/$\overline{T}$ = 0, Timer 1 is clocked by CLKOUT/4 or CLKOUT/12, depending on the state of T1M (CKCON.4). When C/$\overline{T}$ = 1, Timer 1 is clocked by the T1 pin. |
| TMOD.5 | **M1** - Timer 1 mode select bit 1. |
| TMOD.4 | **M0** - Timer 1 mode select bit 0, decoded as:<br>M1 M0 Mode<br>0 0 Mode 0 : 13-bit counter<br>0 1 Mode 1 : 16-bit counter<br>1 0 Mode 2 : 8-bit counter with auto-reload<br>1 1 Mode 3 : Timer 1 stopped |
| TMOD.3 | **GATE** - Timer 0 gate control, When GATE = 1, Timer 0 will clock only when INT0 = 1 and TR0 (TCON.4) = 1. When GATE = 0, Timer 0 will clock only when TR0 = 1, regardless of the state of INT0. |
| TMOD.2 | **C/$\overline{T}$** - Counter/Timer select. When C/$\overline{T}$ = 0, Timer 0 is clocked by CLKOUT/4 or CLKOUT/12, depending on the state of T0M (CKCON.3). When C/$\overline{T}$ = 1, Timer 0 is clocked by the T0 pin. |
| TMOD.1 | **M1** - Timer 0 mode select bit 1. |
| TMOD.0 | **M0** - Timer 0 mode select bit 0, decoded as:<br>M1 M0 Mode<br>0 0 Mode 0 : 13-bit counter<br>0 1 Mode 1 : 16-bit counter<br>1 0 Mode 2 : 8-bit counter with auto-reload<br>1 1 Mode 3 : Two 8-bit counters |

**Table 18-9. TCON Register — SRF 88h**

| Bit | Function |
|---|---|
| TCON.7 | **TF1** - Timer 1 overflow flag. Set to 1 when the Timer 1 count overflows and cleared when the processor vectors to the interrupt service routine. |
| TCON.6 | **TR1** - Timer 1 run control. Set to 1 to enable counting on Timer 1. |
| TCON.5 | **TF0** - Timer 0 overflow flag. Set to 1 when the Timer 0 count overflows and cleared when the processor vectors to the interrupt service routine. |
| TCON.4 | **TR0** - Timer 0 run control. Set to 1 to enable counting on Timer 0. |
| TCON.3 | **IE1** - Interrupt 1 edge detect. If external interrupt 1 is configured to be edge-sensitive (IT1 = 1), IE1 is set by hardware when a negative edge is detected on the INT1 pin and is automatically cleared when the CPU vectors to the corresponding interrupt service routine. In this case, IE1 can also be cleared by software. If external interrupt 1 is configured to be level-sensitive (IT1 = 0), IE1 is set when the INT1# pin is 0 and cleared when the INT1# pin is 1. In level-sensitive mode, software cannot write to IE1. |
| TCON.2 | **IT1** - Interrupt 1 type select. INT1 is detected on falling edge when IT1 = 1; INT1 is detected as a low level when IT1 = 0. |
| TCON.1 | **IE0** - Interrupt 0 edge detect. If external interrupt 0 is configured to be edge-sensitive (IT0 = 1), IE0 is set by hardware when a negative edge is detected on the INT0 pin and is automatically cleared when the CPU vectors to the corresponding interrupt service routine. In this case, IE0 can also be cleared by software. If external interrupt 0 is configured to be level-sensitive (IT0 = 0), IE0 is set when the INT0# pin is 0 and cleared when the INT0# pin is 1. In level-sensitive mode, software cannot write to IE0. |
| TCON.0 | **IT0** - Interrupt 0 type select. INT0 is detected on falling edge when IT0 = 1; INT0 is detected as a low level when IT0 = 0. |

### 18.2.2.3 Mode 2

Mode 2 operation is the same for Timer 0 and Timer 1. In mode 2, the timer is configured as an 8-bit counter, with automatic reload of the start value. The LSB Register (TL0 or TL1) is the counter and the MSB Register (TH0 or TH1) stores the reload value.

As illustrated in *Figure 18-4*, mode 2 counter control is the same as for mode 0 and mode 1. However, in mode 2, when TL*n* increments from FFh, the value stored in TH*n* is reloaded into TL*n*.

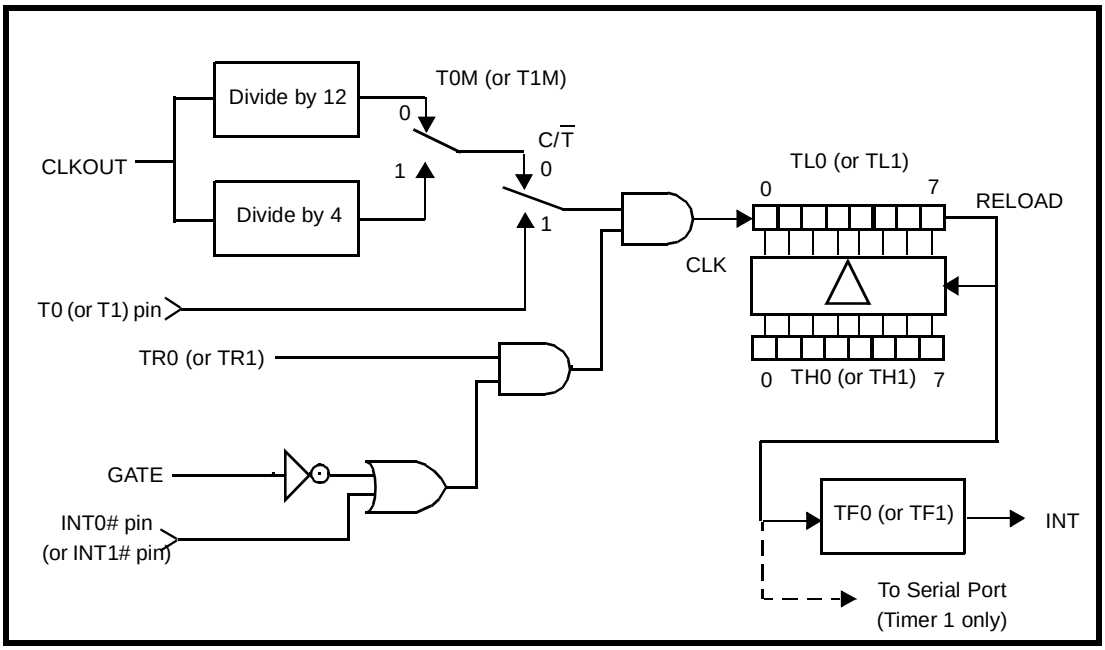


*Figure 18-4. Timer 0/1 - Mode 2*

### 18.2.2.4 Mode 3

In mode 3, Timer 0 operates as two 8-bit counters and Timer 1 stops counting and holds its value.

As shown in *Figure 18-5*, TL0 is configured as an 8-bit counter controlled by the normal Timer 0 control bits. TL0 can either count CLKOUT cycles (divided by 4 or by 12) or high-to-low transitions on *T0*, as determined by the C/$\overline{T}$ Bit. The GATE function can be used to give counter enable control to the INT0# pin.

TH0 functions as an independent 8-bit counter. However, TH0 can only count CLKOUT cycles (divided by 4 or by 12). The Timer 1 control and flag bits (TR1 and TF1) are used as the control and flag bits for TH0.

When Timer 0 is in mode 3, Timer 1 has limited usage because Timer 0 uses the Timer 1 control bit (TR1) and interrupt flag (TF1). Timer 1 can still be used for baud rate generation and the Timer 1 count values are still available in the TL1 and TH1 Registers.

Control of Timer 1 when Timer 0 is in mode 3 is through the Timer 1 mode bits. To turn Timer 1 on, set Timer 1 to mode 0, 1, or 2. To turn Timer 1 off, set it to mode 3. The Timer 1 C/$\overline{T}$ Bit and T1M Bit are still available to Timer 1. Therefore, Timer 1 can count CLKOUT/4, CLKOUT/12, or high-to-low transitions on the T1 pin. The Timer 1 GATE function is also available when Timer 0 is in mode 3.

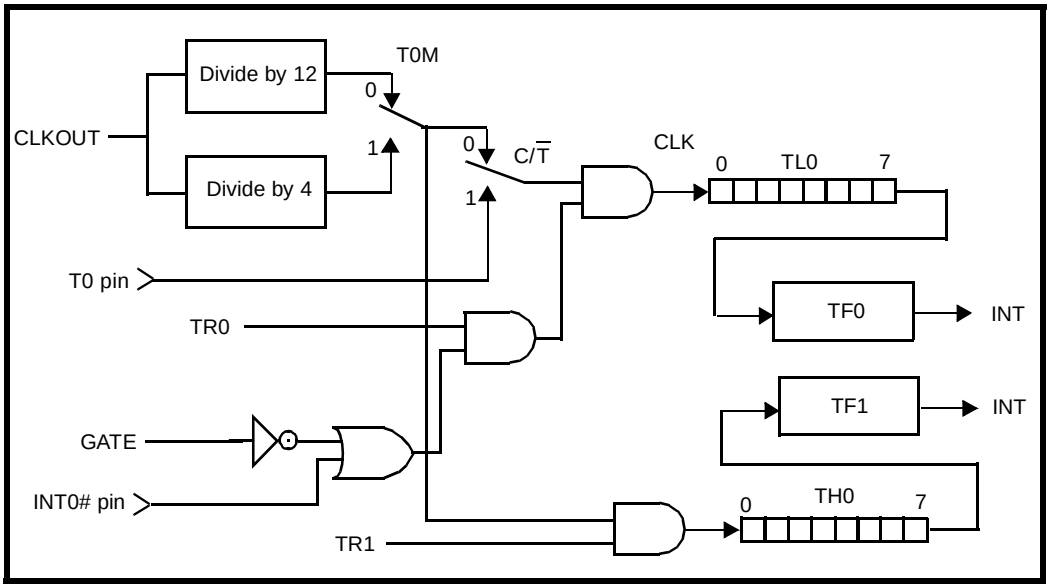


*Figure 18-5. Timer 0 - Mode 3*

### 18.2.3 Timer Rate Control

The default timer clock scheme for the 8051 timers is 12 CLKOUT cycles per increment, the same as in the standard 8051. However, in the 8051, the instruction cycle is 4 CLKOUT cycles.

Using the default rate (12 clocks per timer increment) allows existing application code with real-time dependencies, such as baud rate, to operate properly. However, applications that require fast timing can set the timers to increment every 4 CLKOUT cycles by setting bits in the Clock Control Register (CKCON) at SFR location 8Eh. (See Table 18-10).

The CKCON Bits that control the timer clock rates are:

| CKCON Bit | Counter/Timer |
|---|---|
| 5 | Timer 2 |
| 4 | Timer 1 |
| 3 | Timer 0 |

When a CKCON Register Bit is set to 1, the associated counter increments at 4-CLKOUT intervals. When a CKCON Bit is cleared, the associated counter increments at 12-CLKOUT intervals. The timer controls are independent of each other. The default setting for all three timers is 0 (12-CLKOUT intervals). These bits have no effect in counter mode.

**Table 18-10. CKCON Register — SRF 8Eh**

| Bit | Function |
|---|---|
| CKCON.7,6 | Reserved |
| CKCON.5 | **T2M** - Timer 2 clock select. When T2M = 0, Timer 2 uses CLKOUT/12 (for compatibility with 80C32); when T2M = 1, Timer 2 uses CLKOUT/4. This bit has no effect when Timer 2 is configured for baud rate generation. |
| CKCON.4 | **T1M** - Timer 1 clock select. When T1M = 0, Timer 1 uses CLKOUT/12 (for compatibility with 80C32); when T1M = 1, Timer 1 uses CLKOUT/4. |
| CKCON.3 | **T0M** - Timer 0 clock select. When T0M = 0, Timer 0 uses CLKOUT/12 (for compatibility with 80C32); when T0M = 1, Timer 0 uses CLKOUT/4. |
| CKCON.2-0 | **MD2, MD1, MD0** - Control the number of cycles to be used for external MOVX instructions. |

### *18.2.4 Timer 2*

Timer 2 runs only in 16-bit mode and offers several capabilities not available with Timers 0 and 1. The modes available with Timer 2 are:

- 16-bit timer/counter
- 16-bit timer with capture
- 16-bit auto-reload timer/counter
- Baud rate generator.

The SFRs associated with Timer 2 are:

- T2CON — SFR C8h (Table 18-12)
- RCAP2L — SFR CAh - Used to capture the TL2 value when Timer 2 is configured for capture mode, or as the LSB of the 16-bit reload value when Timer 2 is configured for auto-reload mode.
- RCAP2H — SFR CBh - Used to capture the TH2 value when Timer 2 is configured for capture mode, or as the MSB of the 16-bit reload value when Timer 2 is configured for auto-reload mode.
- TL2 - SFR CCh — Lower 8 bits of the 16-bit count.
- TH2 - SFR CDh — Upper 8 bits of the 16-bit count.

### 18.2.4.1 Timer 2 Mode Control

Table 18-11 summarizes how the SFR Bits determine the Timer 2 mode.

**Table 18-11. Timer 2 Mode Control Summary**

| RCLK | TCLK | CP/$\overline{RL2}$ | TR2 | Mode |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 16-bit timer/counter with capture |
| 0 | 0 | 0 | 1 | 16-bit timer/counter with auto-reload |
| 1 | X | X | 1 | Baud rate generator |
| X | 1 | X | 1 | Baud rate generator |
| X | X | X | 0 | Off |
| X = Don't care. | | | | |

## *18.2.5 16-Bit Timer/Counter Mode*

*Figure 18-6* illustrates how Timer 2 operates in timer/counter mode with the optional capture feature. The C/$\overline{T2}$ Bit determines whether the 16-bit counter counts CLKOUT cycles (divided by 4 or 12), or high-to-low transitions on the T2 pin. The TR2 Bit enables the counter. When the count increments from FFFFh, the TF2 flag is set, and the T2OUT pin goes high for one CLKOUT cycle.

**Table 18-12. T2CON Register — SFR C8h**

| Bit | Function |
|---|---|
| T2CON.7 | **TF2** - Timer 2 overflow flag. Hardware will set TF2 when the Timer 2 overflows from FFFFh. TF2 must be cleared to 0 by the software. TF2 will only be set to a 1 if RCLK and TCLK are both cleared to 0. Writing a 1 to TF2 forces a Timer 2 interrupt if enabled. |
| T2CON.6 | **EXF2** - Timer 2 external flag. Hardware will set EXF2 when a reload or capture is caused by a high-to-low transition on the T2EX pin, and EXEN2 is set. EXF2 must be cleared to 0 by the software. Writing a 1 to EXF2 forces a Timer 2 interrupt if enabled. |
| T2CON.5 | **RCLK** - Receive clock flag. Determines whether Timer 1 or Timer 2 is used for Serial Port 0 timing of received data in serial mode 1 or 3. RCLK =1 selects Timer 2 overflow as the receive clock. RCLK =0 selects Timer 1 overflow as the receive clock. |
| T2CON.4 | **TCLK** - Transmit clock flag. Determines whether Timer 1 or Timer 2 is used for Serial Port 0 timing of transmit data in serial mode 1 or 3. RCLK =1 selects Timer 2 overflow as the transmit clock. RCLK =0 selects Timer 1 overflow as the transmit clock. |
| T2CON.3 | **EXEN2** - Timer 2 external enable. EXEN2 = 1 enables capture or reload to occur as a result of a high-to-low transition on the T2EX pin, if Timer 2 is not generating baud rates for the serial port. EXEN2 = 0 causes Timer 2 to ignore all external events on the T2EX pin. |
| T2CON.2 | **TR2** - Timer 2 run control flag. TR2 = 1 starts Timer 2. TR2 = 0 stops Timer 2. |
| T2CON.1 | **C/$\overline{T2}$** - Counter/timer select. C/$\overline{T2}$ = 0 selects a timer function for Timer 2. C/$\overline{T2}$ = 1 selects a counter of falling transitions on the T2 pin. When used as a timer, Timer 2 runs at 4 clocks per tick or 12 clocks per tick as programmed by CKCON.5, in all modes except baud rate generator mode. When used in baud rate generator mode, Timer 2 runs at 2 clocks per tick, independent of the state of CKCON.5. |
| T2CON.0 | **CP/$\overline{RL2}$** - Capture/reload flag. When CP/$\overline{RL2}$ = 1, Timer 2 captures occur on high-to-low transitions of the T2EX pin, if EXEN2 = 1. When CP/$\overline{RL2}$ = 0, auto-reloads occur when Timer 2 overflows or when high-to-low transitions occur on the T2EX pin, if EXEN2 = 1. If either RCLK or TCLK is set to 1, CP/$\overline{RL2}$ will not function and Timer 2 will operate in auto-reload mode following each overflow. |

### 18.2.5.1 6-Bit Timer/Counter Mode with Capture

The Timer 2 capture mode (*Figure 18-6*) is the same as the 16-bit timer/counter mode, with the addition of the capture registers and control signals.

The CP/$\overline{RL2}$ Bit in the T2CON SFR enables the capture feature. When CP/$\overline{RL2}$ = 1, a high-to-low transition on the T2EX pin when EXEN2 = 1 causes the Timer 2 value to be loaded into the capture registers RCAP2L and RCAP2H.

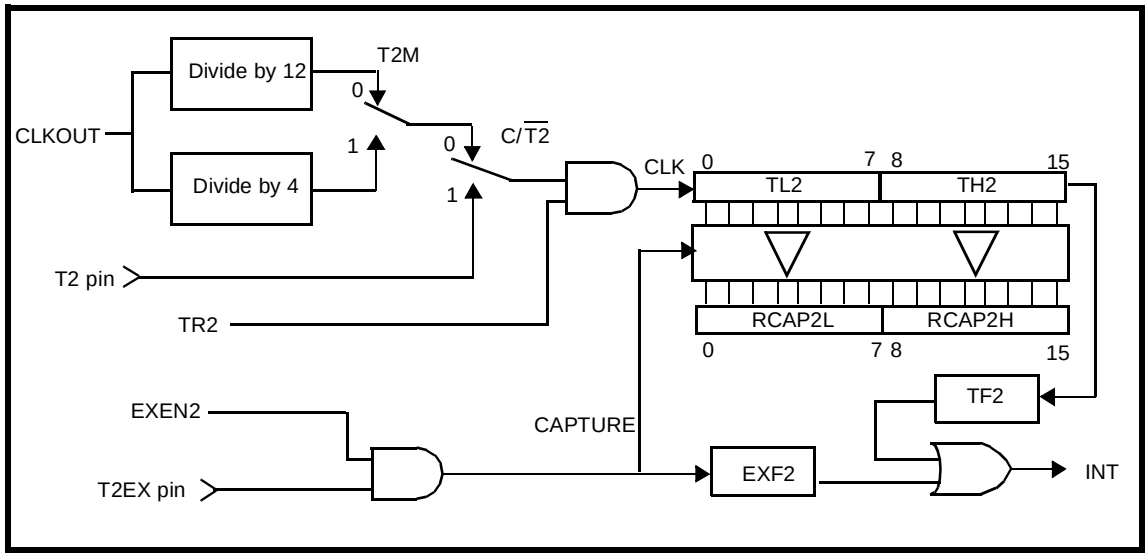


*Figure 18-6. Timer 2 - Timer/Counter with Capture*

### 18.2.6 16-Bit Timer/Counter Mode with Auto-Reload

When CP/$\overline{RL2}$ = 0, Timer 2 is configured for the auto-reload mode illustrated in *Figure 18-7* Control of counter input is the same as for the other 16-bit counter modes. When the count increments from FFFFh, Timer 2 sets the TF2 flag and the starting value is reloaded into TL2 and TH2. The software must preload the starting value into the RCAP2L and RCAP2H Registers.

When Timer 2 is in auto-reload mode, a reload can be forced by a high-to-low transition on the T2EX pin, if enabled by EXEN2 = 1.

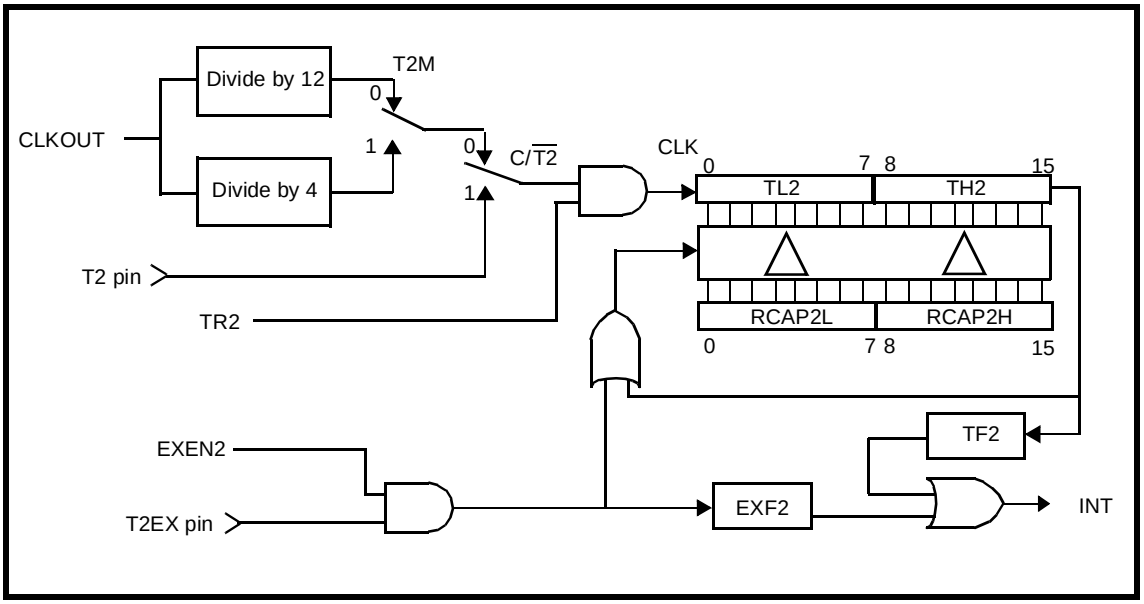


*Figure 18-7. Timer 2 - Timer/Counter with Auto Reload*

### *18.2.7 Baud Rate Generator Mode*

Setting either RCLK or TCLK to 1 configures Timer 2 to generate baud rates for Serial Port 0 in serial mode 1 or 3. In baud rate generator mode, Timer 2 functions in auto-reload mode. However, instead of setting the TF2 flag, the counter overflow is used to generate a shift clock for the serial port function. As in normal auto-reload mode, the overflow also causes the pre-loaded start value in the RCAP2L and RCAP2H Registers to be reloaded into the TL2 and TH2 Registers.

When either TCLK = 1 or RCLK = 1, Timer 2 is forced into auto-reload operation, regardless of the state of the CP/RL2 Bit.

When operating as a baud rate generator, Timer 2 does not set the TF2 Bit. In this mode, a Timer 2 interrupt can only be generated by a high-to-low transition on the T2EX pin setting the EXF2 Bit, and only if enabled by EXEN2 = 1.

The counter time base in baud rate generator mode is CLKOUT/2. To use an external clock source, set C/T2 to 1 and apply the desired clock source to the T2 pin.

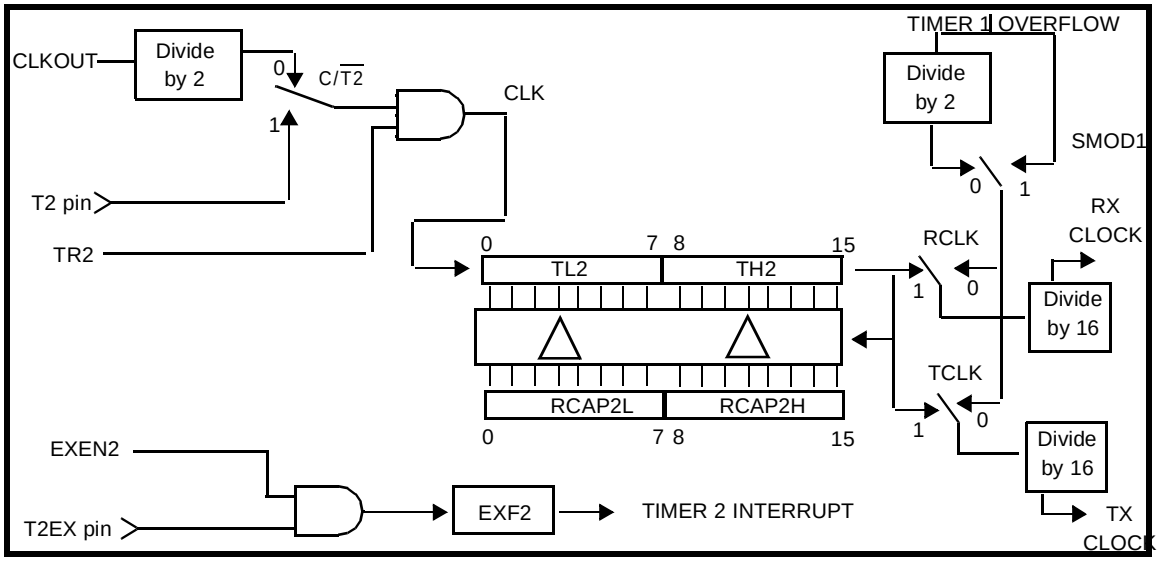


*Figure 18-8. Timer 2 - Baud Rate Generator Mode*

## 18.3 Serial Interface

The 8051 core provides two serial ports. Serial Port 0 is identical in operation to the standard 8051 serial port. Serial Port 1 is identical to Serial Port 0, except that Timer 2 cannot be used as the baud rate generator for Serial Port 1.

Each serial port can operate in synchronous or asynchronous mode. In synchronous mode, 8051 generates the serial clock and the serial port operates in half-duplex mode. In asynchronous mode, the serial port operates in full-duplex mode. In all modes, 8051 buffers received data in a holding register, enabling the UART to receive an incoming byte before the software has read the previous value.

Each serial port can operate in one of four modes, as outlined in Table 18-13.

**Table 18-13. Serial Port Modes**

| Mode | Sync/ Async | Baud Clock | Data Bits | Start/Stop | 9th Bit Function |
|---|---|---|---|---|---|
| 0 | Sync | CLKOUT/4 or CLKOUT/12 | 8 | None | None |
| 1 | Async | Timer 1 or Timer 2[1] | 8 | 1 start, 1 stop | None |
| 2 | Async | CLKOUT/32 or CLKOUT/64 | 9 | 1 start, 1 stop | 0, 1, parity |
| 3 | Async | Timer 1 or Timer 2[1] | 9 | 1 start, 1 stop | 0, 1, parity |
| (1) Timer 2 available for Serial Port 0 only. | | | | | |

The SFRs associated with the serial ports are:

- SCON0 - SFR 98h — Serial Port 0 control (Table 18-14).
- SBUF0 - SFR 99h — Serial Port 0 buffer.
- SCON1 - SFR C0h — Serial Port 1 control (Table 18-15).
- SBUF1 - SFR C1h — Serial Port 1 buffer.

### *18.3.1 803x/805x Compatibility*

The implementation of the serial interface is similar to that of the Intel 8052.

### *18.3.2 Mode 0*

Serial mode 0 provides synchronous, half-duplex serial communication. For Serial Port 0, serial data output occurs on the RXD0OUT pin, serial data is received on the RXD0 pin, and the TXD0 pin provides the shift clock for both transmit and receive. For Serial Port 1, the corresponding pins are RXD1OUT, RXD1, and TXD1.

The serial mode 0 baud rate is either CLKOUT/12 or CLKOUT/4, depending on the state of the SM2_0 Bit (or SM2_1 for Serial Port 1). When SM2_0 = 0, the baud rate is CLKOUT/12, when SM2_0 = 1, the baud rate is CLKOUT/4.

Mode 0 operation is identical to the standard 8051. Data transmission begins when an instruction writes to the SBUF0 (or SBUF1) SFR. The UART shifts the data, LSB first, at the selected baud rate, until the 8-bit value has been shifted out.

Mode 0 data reception begins when the REN_0 (or REN_1) Bit is set and the RI_0 (or RI_1) Bit is cleared in the corresponding SCON SFR. The shift clock is activated and the UART shifts data in on each rising edge of the shift clock until 8 bits have been received. One machine cycle after the 8th bit is shifted in, the RI_0 (or RI_1) Bit is set and reception stops until the software clears the RI Bit.

*Figure 18-9* through *Figure 18-12* illustrate Serial Port Mode 0 transmit and receive timing for both low-speed (CLKOUT/12) and high-speed (CLKOUT/4) operation.

**Table 18-14. SCON0 Register — SFR 98h**

| Bit | Function |
|---|---|
| SCON0.7 | **SM0_0** - Serial Port 0 mode bit 0. |
| SCON0.6 | **SM1_0** - Serial Port 0 mode bit 1, decoded as:<br>SM0_0 SM1_0 Mode<br>0 0 0<br>0 1 1<br>1 0 2<br>1 1 3 |
| SCON0.5 | **SM2_0** - Multiprocessor communication enable. In modes 2 and 3, this bit enables the multi-processor communication feature. If SM2_0 = 1 in mode 2 or 3, then RI_0 will not be activated if the received 9th bit is 0.<br><br>If SM2_0=1 in mode 1, then RI_0 will only be activated if a valid stop is received. In mode 0, SM2_0 establishes the baud rate: when SM2_0=0, the baud rate is CLKOUT/12; when SM2_0=1, the baud rate is CLKOUT/4. |
| SCON0.4 | **REN_0** - Receive enable. When REN_0=1, reception is enabled. |
| SCON0.3 | **TB8_0** - Defines the state of the 9th data bit transmitted in modes 2 and 3. |
| SCON0.2 | **RB8_0** - In modes 2 and 3, RB8_0 indicates the state of the 9th bit received. In mode 1, RB8_0 indicates the state of the received stop bit. In mode 0, RB8_0 is not used. |
| SCON0.1 | **TI_0** - Transmit interrupt flag. indicates that the transmit data word has been shifted out. In mode 0, TI_0 is set at the end of the 8th data bit. In all other modes, TI_0 is set when the stop bit is placed on the TXD0 pin. **TI_0 must be cleared by firmware.** |
| SCON0.0 | **RI_0** - Receive interrupt flag. Indicates that serial data word has been received. In mode 0, RI_0 is set at the end of the 8th data bit. In mode 1, RI_0 is set after the last sample of the incoming stop bit, subject to the state of SM2_0. In modes 2 and 3, RI_0 is set at the end of the last sample of RB8_0. **RI_0 must be cleared by firmware.** |

**Table 18-15. SCON1 Register — SFR C0h**

| Bit | Function |
|---|---|
| SCON1.7 | **SM0_1** - Serial Port 1 mode bit 0. |
| SCON1.6 | **SM1_1** - Serial Port 1 mode bit 1, decoded as:<br>SM0_1 SM1_1 Mode<br>0 0 0<br>0 1 1<br>1 0 2<br>1 1 3 |
| SCON1.5 | **SM2_1** - Multiprocessor communication enable. In modes 2 and 3, this bit enables the multiprocessor communication feature. If SM2_1 = 1 in mode 2 or 3, then RI_1 will not be activated if the received 9th bit is 0.<br><br>If SM2_1=1 in mode 1, then RI_1 will only be activated if a valid stop is received. In mode 0, SM2_1 establishes the baud rate: when SM2_1=0, the baud rate is CLKOUT/12; when SM2_1=1, the baud rate is CLKOUT/4. |
| SCON1.4 | **REN_1** - Receive enable. When REN_1=1, reception is enabled. |
| SCON1.3 | **TB8_1** - Defines the state of the 9th data bit transmitted in modes 2 and 3. |
| SCON1.2 | **RB8_1** - In modes 2 and 3, RB8_0 indicates the state of the 9th bit received. In mode 1, RB8_1 indicates the state of the received stop bit. In mode 0, RB8_1 is not used. |
| SCON1.1 | **TI_1** - Transmit interrupt flag. indicates that the transmit data word has been shifted out. In mode 0, TI_1 is set at the end of the 8th data bit. In all other modes, TI_1 is set when the stop bit is placed on the TXD0 pin. TI_1 must be cleared by the software. |
| SCON1.0 | **RI_1** - Receive interrupt flag. Indicates that serial data word has been received. In mode 0, RI_1 is set at the end of the 8th data bit. In mode 1, RI_1 is set after the last sample of the incoming stop bit, subject to the state of SM2_1. In modes 2 and 3, RI_1 is set at the end of the last sample of RB8_1. RI_1 must be cleared by the software. |

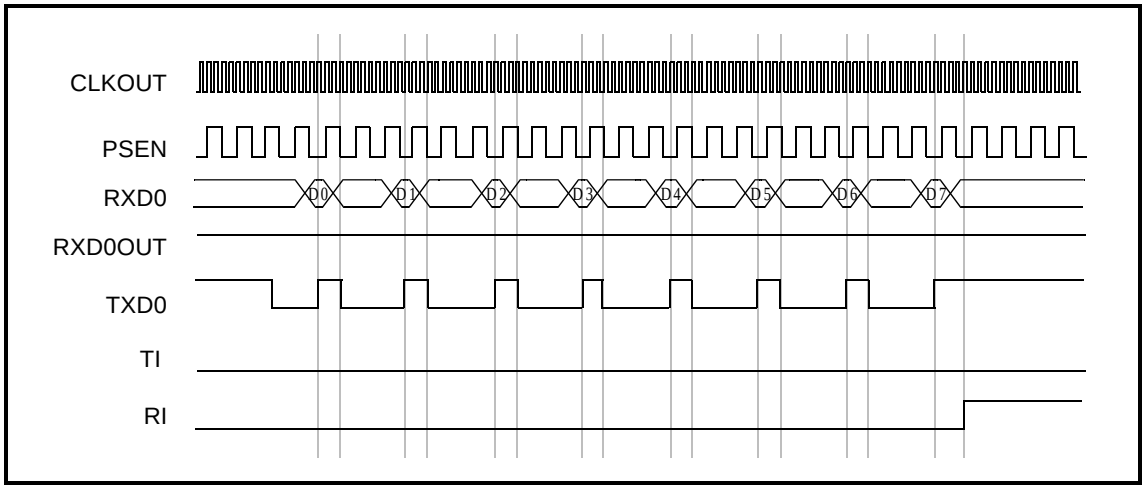


*Figure 18-9. Serial Port Mode 0 Receive Timing - Low Speed Operation*

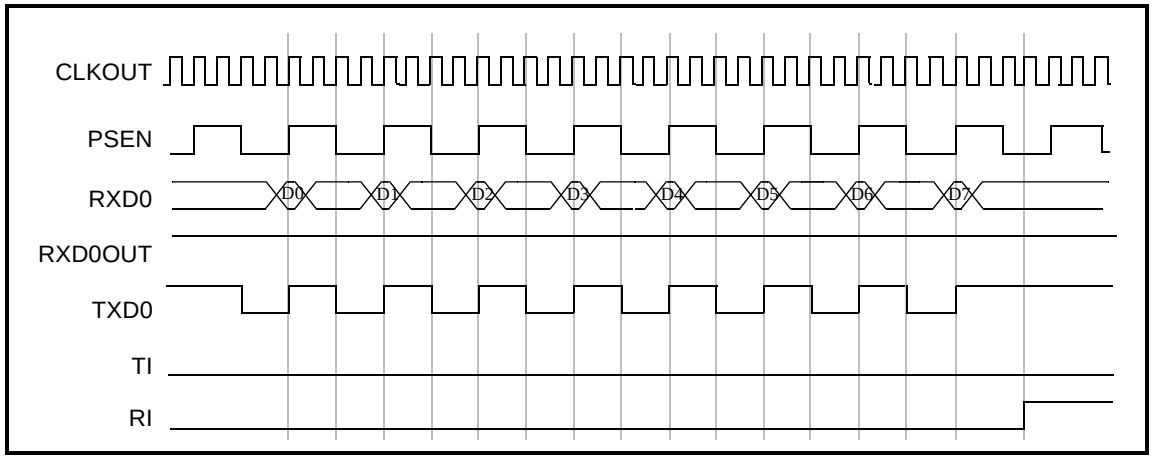


*Figure 18-10. Serial Port Mode 0 Receive Timing - High Speed Operation*

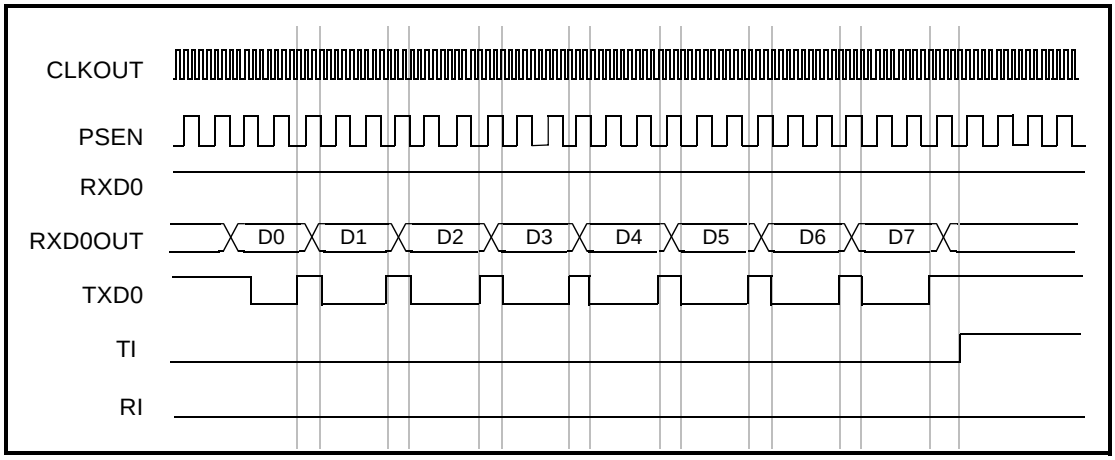


*Figure 18-11. Serial Port Mode 0 Transmit Timing - Low Speed Operation*

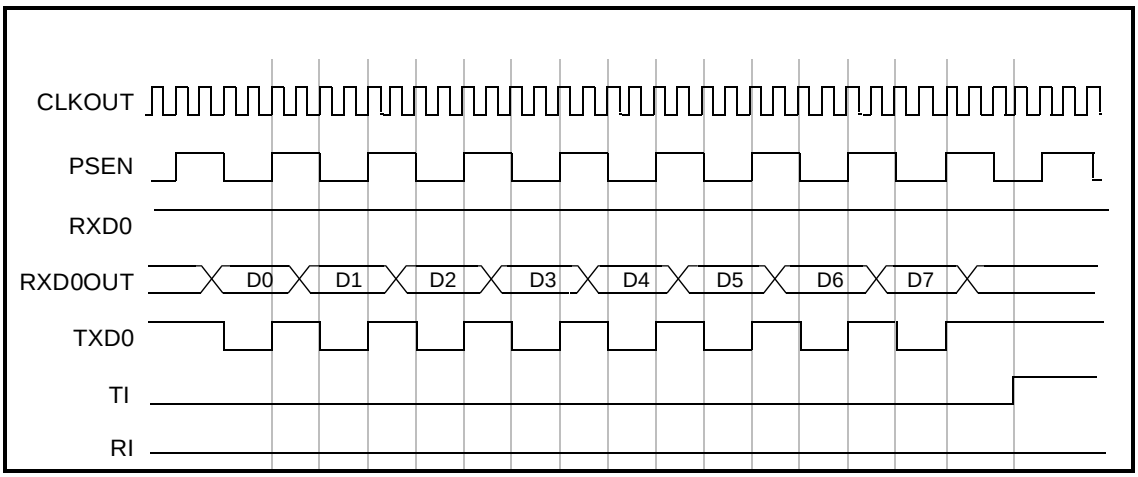


*Figure 18-12. Serial Port Mode 0 Transmit Timing - High Speed Operation*

### *18.3.3 Mode 1*

Mode 1 provides standard asynchronous, full-duplex communication, using a total of 10 bits: 1 start bit, 8 data bits, and 1 stop bit. For receive operations, the stop bit is stored in RB8_0 (or RB8_1). Data bits are received and transmitted LSB first.

#### 18.3.3.1 Mode 1 Baud Rate

The mode 1 baud rate is a function of timer overflow. Serial Port 0 can use either Timer 1 or Timer 2 to generate baud rates. Serial Port 1 can only use Timer 1. The two serial ports can run at the same baud rate if they both use Timer 1, or different baud rates if Serial Port 0 uses Timer 2 and Serial Port 1 uses Timer 1.

Each time the timer increments from its maximum count (FFh for Timer 1 or FFFFh for Timer 2), a clock is sent to the baud rate circuit. The clock is then divided by 16 to generate the baud rate.

When using Timer 1, the SMOD0 (or SMOD1) Bit selects whether or not to divide the Timer 1 roll-over rate by 2. Therefore, when using Timer 1, the baud rate is determined by the equation:

$$\text{Baud Rate} = \frac{2^{\text{SMODx}}}{32} \times \text{Timer 1 Overflow}$$

SMOD0 is SFR Bit PCON.7; SMOD1 is SFR Bit EICON.7.

When using Timer 2, the baud rate is determined by the equation:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow}}{16}$$

To use Timer 1 as the baud rate generator, it is best to use Timer 1 mode 2 (8-bit counter with auto-reload), although any counter mode can be used. The Timer 1 reload is stored in the TH1 Register, which makes the complete formula for Timer 1:

$$\text{Baud Rate} = \frac{2^{\text{SMODx}}}{32} \times \frac{\text{CLKOUT}}{12 \times (256 - \text{TH1})}$$

The 12 in the denominator in the above equation can be changed to 4 by setting the T1M Bit in the CKCON SFR. To derive the required TH1 value from a known baud rate (when TM1 = 0), use the equation:

$$\text{TH1} = 256 - \frac{2^{\text{SMODx}} \times \text{CLKOUT}}{384 \times \text{Baud Rate}}$$

You can also achieve very low serial port baud rates from Timer 1 by enabling the Timer 1 interrupt, configuring Timer 1 to mode 1, and using the Timer 1 interrupt to initiate a 16-bit software reload. Table 18-16 lists sample reload values for a variety of common serial port baud rates.

*More accurate baud rates are achieved by using Timer 2 as the baud rate generator (next section).*

**Table 18-16. Timer 1 Reload Values for Common Serial Port Mode 1 Baud Rates**

| Nominal Rate | 24 MHz Divisor | Reload Value | Actual Rate | Error |
|---|---|---|---|---|
| 57600 | 6 | FA | 62500 | 8.5% |
| 38400 | 10 | F6 | 37500 | -2.3% |
| 28800 | 13 | F3 | 28846 | +0.16% |
| 19200 | 20 | EC | 18750 | -2.3% |
| 9600 | 39 | D9 | 9615 | +0.16% |
| 4800 | 78 | B2 | 4807 | +0.15% |
| 2400 | 156 | 64 | 2403 | +.13% |
| Settings: SMOD =1, C/$\overline{T}$=0, Timer1 mode=2, TIM=1<br>***Note:*** Using rates that are off by 2.3% or more will not work in all systems. | | | | |

To use Timer 2 as the baud rate generator, configure Timer 2 in auto-reload mode and set the TCLK and/or RCLK Bits in the T2CON SFR. TCLK selects Timer 2 as the baud rate generator for the transmitter; RCLK selects Timer 2 as the baud rate generator for the receiver. The 16-bit reload value for Timer 2 is stored in the RCAP2L and RCA2H SFRs, which makes the equation for the Timer 2 baud rate:

$$\text{Baud Rate} = \frac{\text{CLKOUT}}{32 \times (65536 - \text{RCAP2H,RCAP2L})}$$

where RCAP2H,RCAP2L is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned number.

The 32 in the denominator is the result of CLKOUT being divided by 2 and the Timer 2 overflow being divided by 16. Setting TCLK or RCLK to 1 automatically causes CLKOUT to be divided by 2, as shown in *Figure 18-8*, instead of the 4 or 12 as determined by the T2M Bit in the CKCON SFR.

To derive the required RCAP2H and RCAP2L values from a known baud rate, use the equation:

$$\text{RCAP2H,RCAP2L} = 65536 - \frac{\text{CLKOUT}}{32 \times \text{Baud Rate}}$$

When either RCLK or TCLK is set, the TF2 flag is not set on a Timer 2 roll over, and the T2EX reload trigger is disabled.

**Table 18-17. Timer 2 Reload Values for Common Serial Port Mode 1 Baud Rates**

| Nominal Rate | C/T2 | Divisor | Reload Val | Actual Rate | Error |
|---|---|---|---|---|---|
| 57600 | 0 | 13 | F3 | 57692.31 | 0.16% |
| 38400 | 0 | 20 | EC | 37500 | -2.34% |
| 28800 | 0 | 26 | E6 | 28846.15 | 0.16% |
| 19200 | 0 | 39 | D9 | 19230.77 | 0.16% |
| 9600 | 0 | 78 | B2 | 9615.385 | 0.16% |
| 4800 | 0 | 156 | 64 | 4807.692 | 0.16% |
| 2400 | 0 | 312 | FEC8 | 2403.846 | 0.16% |
| ***Note:*** using rates that are off by 2.3% or more will not work in all systems. | | | | | |

### 18.3.3.2 Mode 1 Transmit

*Figure 18-13* illustrates the mode 1 transmit timing. In mode 1, the UART begins transmitting after the first roll over of the divide-by-16 counter after the software writes to the SBUF0 (or SBUF1) Register. The UART transmits data on the TXD0 (or TXD1) pin in the following order: start bit, 8 data bits (LSB first), stop bit. The TI_0 (or TI_1) Bit is set 2 CLKOUT cycles after the stop bit is transmitted.

## *18.3.4 Mode 1 Receive*

*Figure 18-14* illustrates the mode 1 receive timing. Reception begins at the falling edge of a start bit received on the RXD0 (or RXD1) pin, when enabled by the REN_0 (or REN_1) Bit. For this purpose, the RXD0 (or RXD1) pin is sampled 16 times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receive clock is reset to align the counter roll over to the bit boundaries.

For noise rejection, the serial port establishes the content of each received bit by a majority decision of 3 consecutive samples in the middle of each bit time. This is especially true for the start bit. If the falling edge on the RXD0 (or RXD1) pin is not verified by a majority decision of 3 consecutive samples (low), then the serial port stops reception and waits for another falling edge on the RXD0 (or RXD1) pin.

At the middle of the stop bit time, the serial port checks for the following conditions:

- RI_0 (or RI_1) = 0, and
- If SM2_0 (or SM2_1) = 1, the state of the stop bit is 1.
  (If SM2_0 (or SM2_1) = 0, the state of the stop bit doesn't matter.)

If the above conditions are met, the serial port then writes the received byte to the SBUF0 (or SBUF1) Register, loads the stop bit into RB8_0 (or RB8_1), and sets the RI_0 (or RI_1) Bit. If the above conditions are not met, the received data is lost, the SBUF Register and RB8 Bit are not loaded, and the RI Bit is not set.

After the middle of the stop bit time, the serial port waits for another high-to-low transition on the (RXD0 or RXD1) pin.

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

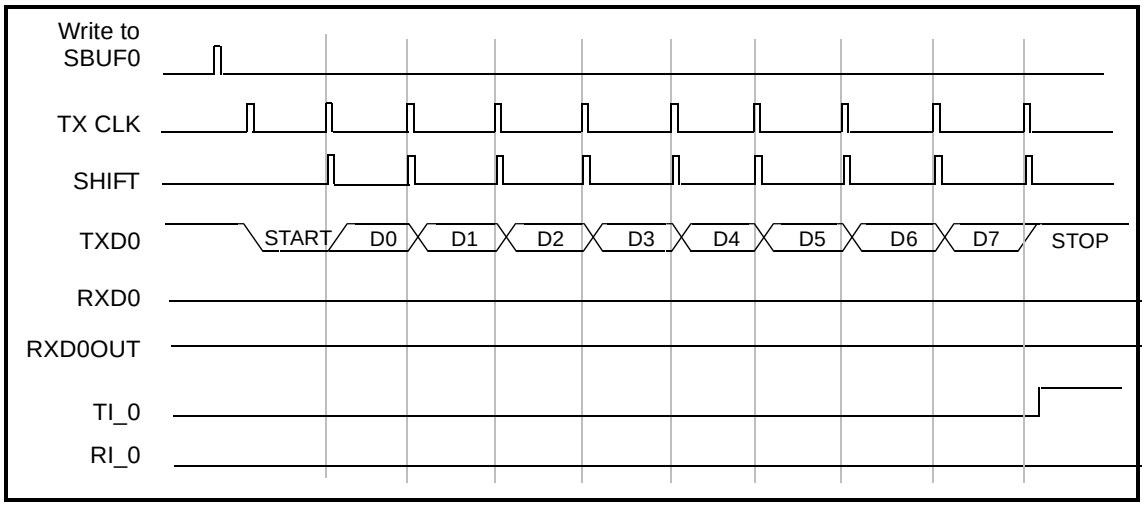


*Figure 18-13. Serial Port 0 Mode 1 Transmit Timing*

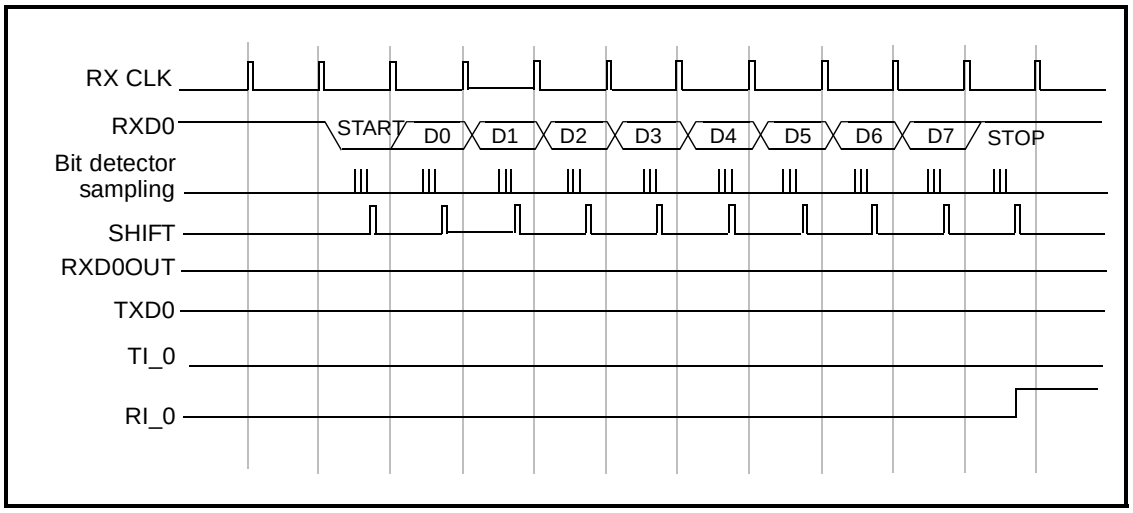


*Figure 18-14. Serial Port 0 Mode 1 Receive Timing*

### *18.3.5 Mode 2*

Mode 2 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. For transmission, the 9th bit is determined by the value in TB8_0 (or TB8_1). To use the 9th bit as a parity bit, move the value of the P Bit (SFR PSW.0) to TB8_0 (or TB8_1).

The mode 2 baud rate is either CLKOUT/32 or CLKOUT/64, as determined by the SMOD0 (or SMOD1) Bit. The formula for the mode 2 baud rate is:

$$\text{Baud Rate} = \frac{2^{\text{SMODx}} \times \text{CLKOUT}}{64}$$

Mode 2 operation is identical to the standard 8051.

#### 18.3.5.1 Mode 2 Transmit

*Figure 18-15* illustrates the mode 2 transmit timing. Transmission begins after the first roll over of the divide-by-16 counter following a software write to SBUF0 (or SBUF1). The UART shifts data out on the TXD0 (or TXD1) pin in the following order: start bit, data bits (LSB first), 9th bit, stop bit. The TI_0 (or TI_1) Bit is set when the stop bit is placed on the TXD0 (or TXD1) pin.

### 18.3.5.2 Mode 2 Receive

*Figure 18-16* illustrates the mode 2 receive timing. Reception begins at the falling edge of a start bit received on the RXD0 (or RXD1) pin, when enabled by the REN_0 (or REN_1) Bit. For this purpose, the RXD0 (or RXD1) pin is sampled 16 times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receive clock is reset to align the counter roll over to the bit boundaries.

For noise rejection, the serial port establishes the content of each received bit by a majority decision of 3 consecutive samples in the middle of each bit time. This is especially true for the start bit. If the falling edge on the RXD0 (or RXD1) pin is not verified by a majority decision of 3 consecutive samples (low), then the serial port stops reception and waits for another falling edge on the RXD0 (or RXD1) pin.

At the middle of the stop bit time, the serial port checks for the following conditions:

- RI_0 (or RI_1) = 0, and
- If SM2_0 (or SM2_1) = 1, the state of the stop bit is 1.
  (If SM2_0 (or SM2_1) = 0, the state of the stop bit doesn't matter.)

If the above conditions are met, the serial port then writes the received byte to the SBUF0 (or SBUF1) Register, loads the stop bit into RB8_0 (or RB8_1), and sets the RI_0 (or RI_1) Bit. If the above conditions are not met, the received data is lost, the SBUF Register and RB8 Bit are not loaded, and the RI Bit is not set. After the middle of the stop bit time, the serial port waits for another high-to-low transition on the RXD0 (or RXD1) pin.

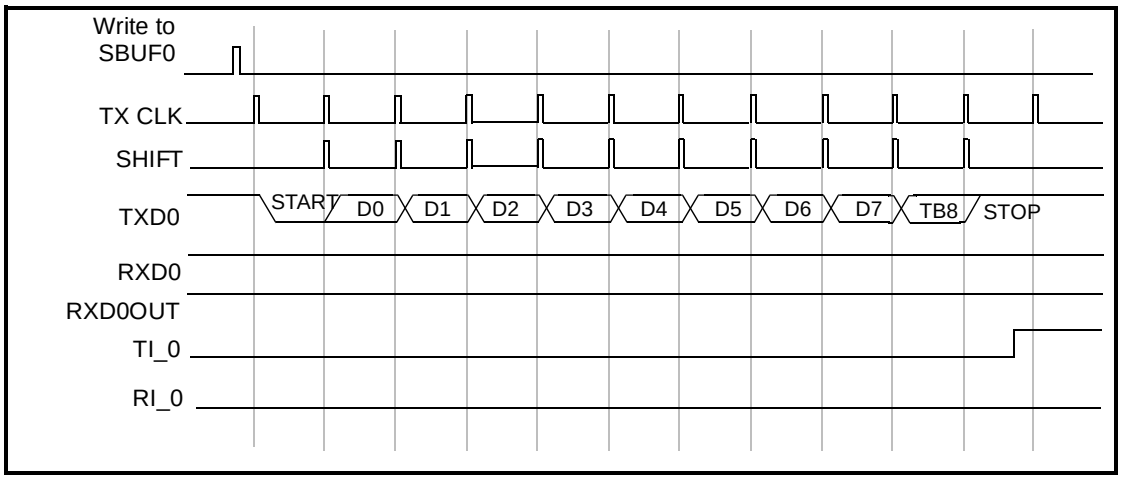


*Figure 18-15. Serial Port 0 Mode 2 Transmit Timing*

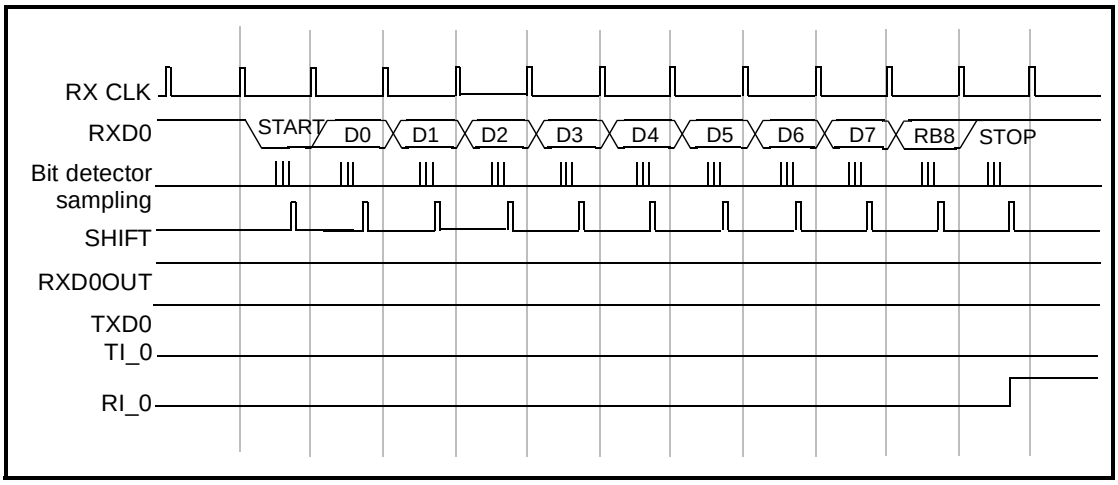


*Figure 18-16. Serial Port 0 Mode 2 Receive Timing*

### *18.3.6 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 18-17* illustrates the mode 3 transmit timing.

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

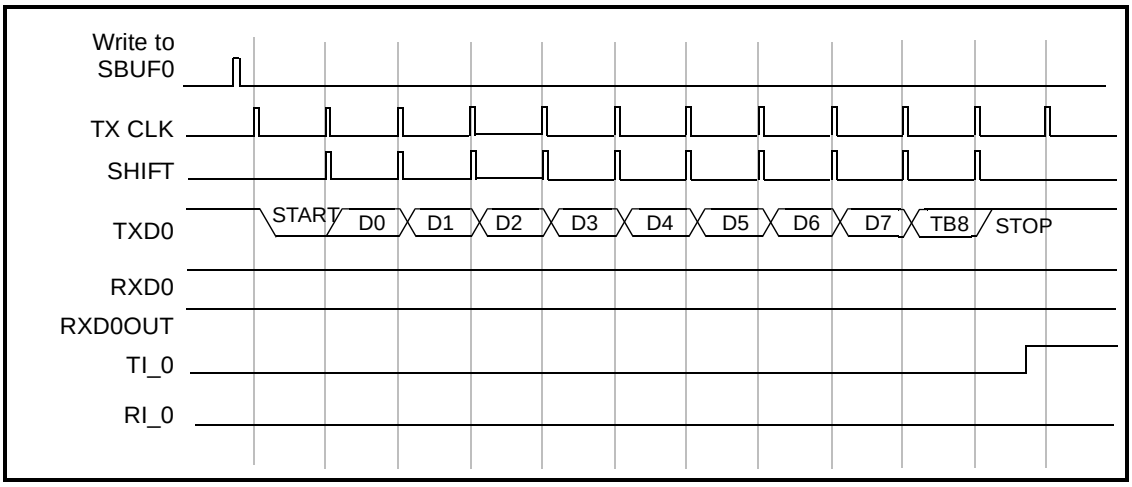


*Figure 18-17. Serial Port 0 Mode 3 Transmit Timing*

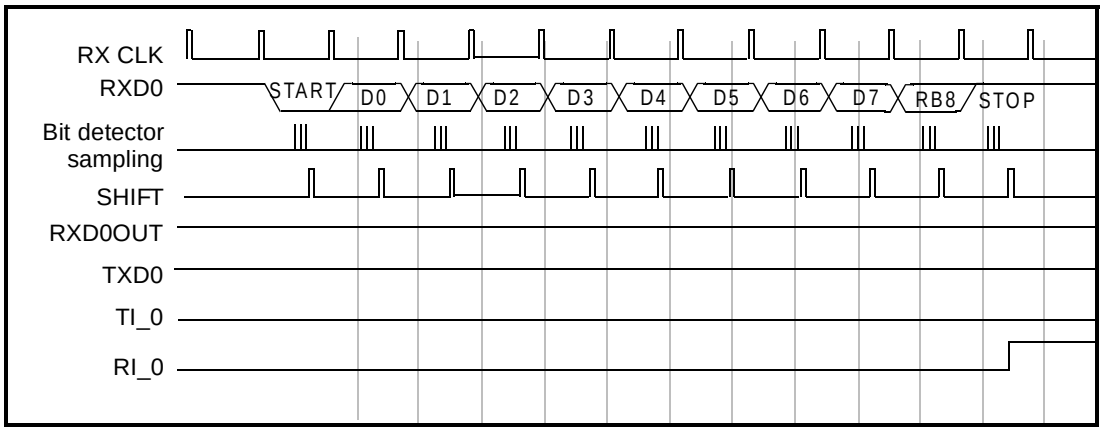


*Figure 18-18. Serial Port 0 Mode 3 Receive Timing*

### 18.3.7 Multiprocessor Communications

The multiprocessor communication feature is enabled in modes 2 and 3 when the SM2 Bit is set in the SCON SFR for a serial port (SM2_0 for Serial Port 0, SM2_1 for Serial Port 1). In multiprocessor communication mode, the 9th bit received is stored in RB8_0 (or RB8_1) and, after the stop bit is received, the serial port interrupt is activated only if RB8_0 (or RB8_1) = 1.

A typical use for the multiprocessor communication feature is when a master wants to send a block of data to one of several slaves. The master first transmits an address byte that identifies the target slave. When transmitting an address byte, the master sets the 9th bit to 1; for data bytes, the 9th bit is 0.

With SM2_0 (or SM2_1) = 1, no slave will be interrupted by a data byte. However, an address byte interrupts all slaves so that each slave can examine the received address byte to determine whether that slave is being addressed. Address decoding must be done by software during the interrupt service routine. The addressed slave clears its SM2_0 (or SM2_1) Bit and prepares to receive the data bytes. The slaves that are not being addressed leave the SM2_0 (or SM2_1) Bit set and ignore the incoming data bytes.

### 18.3.8 Interrupt SFRs

The following SFRs are associated with interrupt control:

- IE - SFR A8h (Table 18-18)
- IP - SFR B8h (Table 18-19)
- EXIF - SFR 91h (Table 18-20)
- EICON - SFR D8h (Table 18-21)
- EIE - SFR E8h (Table 18-22)
- EIP - SFR F8h (Table 18-23).

The IE and IP SFRs provide interrupt enable and priority control for the standard interrupt unit, as with the standard 8051. Additionally, these SFRs provide control bits for the Serial Port 1 interrupt. These bits (ES1 and PS1) are available only when the extended interrupt unit is implemented (ext_intr=1). Otherwise, they are read as 0.

Bits ES0, ES1, ET2, PS0, PS1, and PT2 are present, but not used, when the corresponding module is not implemented.

The EXIF, EICON, EIE and EIP Registers provide flags, enable control, and priority control for the optional extended interrupt unit.

**Table 18-18. IE Register — SFR A8h**

| Bit | Function |
|---|---|
| IE.7 | **EA** - Global interrupt enable. Controls masking of all interrupts except USB wakeup (resume). EA = 0 disables all interrupts except USB wakeup. When EA = 1, interrupts are enabled or masked by their individual enable bits. |
| IE.6 | **ES1** - Enable Serial Port 1 interrupt. ES1 = 0 disables Serial port 1 interrupts (TI_1 and RI_1). ES1 = 1 enables interrupts generated by the TI_1 or TI_1 flag. |
| IE.5 | **ET2** - Enable Timer 2 interrupt. ET2 = 0 disables Timer 2 interrupt (TF2). ET2=1 enables interrupts generated by the TF2 or EXF2 flag. |
| IE.4 | **ES0** - Enable Serial Port 0 interrupt. ES0 = 0 disables Serial Port 0 interrupts (TI_0 and RI_0). ES0=1 enables interrupts generated by the TI_0 or RI_0 flag. |
| IE.3 | **ET1** - Enable Timer 1 interrupt. ET1 = 0 disables Timer 1 interrupt (TF1). ET1=1 enables interrupts generated by the TF1 flag. |
| IE.2 | **EX1** - Enable external interrupt 1. EX1 = 0 disables external interrupt 1 (INT1). EX1=1 enables interrupts generated by the INT1# pin. |
| IE.1 | **ET0** - Enable Timer 0 interrupt. ET0 = 0 disables Timer 0 interrupt (TF0). ET0=1 enables interrupts generated by the TF0 flag. |
| IE.0 | **EX0** - Enable external interrupt 0. EX0 = 0 disables external interrupt 0 (INT0). EX0=1 enables interrupts generated by the INT0# pin. |

**Table 18-19. IP Register — SFR B8h**

| Bit | Function |
|---|---|
| IP.7 | **Reserved**. Read as 1. |
| IP.6 | **PS1** - Serial Port 1 interrupt priority control. PS1=0 sets Serial Port 1 interrupt (TI_1 or RI_1) to low priority. PS1=1 sets Serial port 1 interrupt to high priority. |
| IP.5 | **PT2** - Timer 2 interrupt priority control. PT2=0 sets Timer 2 interrupt (TF2) to low priority. PT2=1 sets Timer 2 interrupt to high priority. |
| IP.4 | **PS0** - Serial Port 0 interrupt priority control. PS0=0 sets Serial Port 0 interrupt (TI_0 or RI_0) to low priority. PS0=1 sets Serial Port 0 interrupt to high priority. |
| IP.3 | **PT2** - Timer 1 interrupt priority control. PT1 = 0 sets Timer 1 interrupt (TF1) to low priority. PT1=1 sets Timer 1 interrupt to high priority. |
| IP.2 | **PX1** - External interrupt 1 priority control. PX 1= 0 sets external interrupt 1 (INT1) to low priority. PT1 = 1 sets external interrupt 1 to high priority. |
| IP.1 | **PT0** - Timer 0 interrupt priority control. PT0 = 0 sets Timer 0 interrupt (TF0) to low priority. PT0=1 sets Timer 0 interrupt to high priority. |
| IP.0 | **PX0** - External interrupt 0 priority control. PX0 = 0 sets external interrupt 0 (INT0) to low priority. PX0=1 sets external interrupt 0 to high priority. |

**Table 18-20. EXIF Register — SFR 91h**

| Bit | Function |
|---|---|
| EXIF.7 | **IE5** - External interrupt 5 flag. IE 5= 1 indicates a falling edge was detected at the INT5# pin. IE5 must be cleared by software. Setting IE5 in software generates an interrupt, if enabled. |
| EXIF.6 | **IE4** - External interrupt 4 flag. IE4 indicates a rising edge was detected at the INT4 pin. IE4 must be cleared by software. Setting IE4 in software generates an interrupt, if enabled. |
| EXIF.5 | **I2CINT** - External interrupt 3 flag. The “INT3” interrupt is internally connected to the EZ-USB FX $I^2C$ controller and renamed “I2CINT”. I2CINT = 1 indicates an $I^2C$ interrupt. I2CINT must be cleared by software. Setting I2CINT in software generates an interrupt, if enabled. |
| EXIF.4 | **USBINT** - External interrupt 2 flag. The “INT2” interrupt is internally connected to the EZ-USB FX interrupt and renamed “USBINT”. USBINT = 1 indicates an USB interrupt. USBINT must be cleared by software. Setting USBINT in software generates an interrupt, if enabled. |
| EXIF.3 | **Reserved**. Read as 1. |
| EXIF.2-0 | **Reserved**. Read as 0. |

**Table 18-21. EICON Register — SFR D8h**

| Bit | Function |
|---|---|
| EICON.7 | **SMOD1** - Serial Port 1 baud rate doubler enable. When SMOD1 = 1 the baud rate for Serial Port is doubled. |
| EICON.6 | **Reserved**. Read as 1. |
| EICON.5 | **ERESI** - Enable resume interrupt. ERESI = 0 disables resume interrupt (RESI). ERESI = 1 enables interrupts generated by the resume event. |
| EICON.4 | **RESI** - Wakeup interrupt flag. EICON.4 = 1 indicates a negative transition was detected at the WAKEUP# pin, or that USB has activity resumed from the suspended state. EICON.4 = 1 must be cleared by software before exiting the interrupt service routine, otherwise the interrupt occurs again. Setting EICON.4=1 in software generates a wakeup interrupt, if enabled. |
| EICON.3 | **INT6** - External interrupt 6. When INT6 = 1, the INT6 pin has detected a low to high transition. INT6 will remain active until cleared by writing a 0 to this bit. Setting this bit in software generates an INT6 interrupt in enabled. |
| EICON.2-0 | **Reserved**. Read as 0. |

**Table 18-22. EIE Register — SFR E8h**

| Bit | Function |
|---|---|
| EIE.7-5 | Reserved. Read as 1. |
| EIE.4 | EX6 - Enable external interrupt 6. EX6 = 0 disables external interrupt 6 (INT6). EX6 = 1 enables interrupts generated by the INT6 pin. |
| EIE.3 | EX5 - Enable external interrupt 5. EX5 = 0 disables external interrupt 5 (INT5). EX5 = 1 enables interrupts generated by the INT5# pin. |
| EIE.2 | EX4 - Enable external interrupt 4. EX4 = 0 disables external interrupt 4 (INT4). EX4 = 1 enables interrupts generated by the INT4 pin. |
| EIE.1 | EI2C - Enable external interrupt 3. EI2C = 0 disables external interrupt 3 (INT3). EI2C = 1 enables interrupts generated by the $I^2C$ interface. |
| EIE.0 | EUSB - Enable USB interrupt. EUSB = 0 disables USB interrupts. EUSB = 1 enables interrupts generated by the USB Interface. |

**Table 18-23. EIP Register — SFR F8h**

| Bit | Function |
|---|---|
| EIP.7-5 | **Reserved**. Read as 1. |
| EIP.4 | **PX6** - External interrupt 6 priority control. PX6 = 0 sets external interrupt 6 (INT6) to low priority. PX6 = 1 sets external interrupt 6 to high priority. |
| EIP.3 | **PX5** - External interrupt 5 priority control. PX5 = 0 sets external interrupt 5 (INT5#) to low priority. PX5=1 sets external interrupt 5 to high priority. |
| EIP.2 | **PX4** - External interrupt 4 priority control. PX4 = 0 sets external interrupt 4 (INT4) to low priority. PX4=1 sets external interrupt 4 to high priority. |
| EIP.1 | **PI2C** - External interrupt 3 priority control. PI2C = 0 sets $I^2C$ interrupt to low priority. PI2C=1 sets $I^2C$ interrupt to high priority. |
| EIP.0 | **PUSB** - External interrupt 2 priority control. PUSB = 0 sets USB interrupt to low priority. PUSB=1 sets USB interrupt to high priority. |

## 18.4 Interrupt Processing

When an enabled interrupt occurs, the 8051 core vectors to the address of the interrupt service routine (ISR) associated with that interrupt, as listed in Table 18-24. The 8051 core executes the ISR to completion unless another interrupt of higher priority occurs. Each ISR ends with a `RETI` (return from interrupt) instruction. After executing the `RETI`, the CPU returns to the next instruction that would have been executed if the interrupt had not occurred.

An ISR can only be interrupted by a higher priority interrupt. That is, an ISR for a low-level interrupt can only be interrupted by high-level interrupt. An ISR for a high-level interrupt can only be interrupted by the resume interrupt.

The 8051 core always completes the instruction in progress before servicing an interrupt. If the instruction in progress is RETI, or a write access to any of the IP, IE, EIP, or EIE SFRs, the 8051 core completes one additional instruction before servicing the interrupt.

### 18.4.1 Interrupt Masking

The EA Bit in the IE SFR (IE.7) is a global enable for all interrupts except the USB wakeup (resume) interrupt. When EA = 1, each interrupt is enabled or masked by its individual enable bit. When EA = 0, all interrupts are masked, except the USB wakeup interrupt.

Table 18-25 provides a summary of interrupt sources, flags, enables, and priorities.

**Table 18-24. Interrupt Natural Vectors and Priorities**

| Interrupt | Description | Natural Priority | Interrupt Vector |
|---|---|---|---|
| RESUME | USB Wakeup (resume) interrupt | 0 | 33h |
| INT0 | External interrupt 0 | 1 | 03h |
| TF0 | Timer 0 interrupt | 2 | 0Bh |
| INT1 | External interrupt 1 | 3 | 13h |
| TF1 | Timer 1 interrupt | 4 | 1Bh |
| TI_0 or RI_0 | Serial port 0 interrupt | 5 | 23h |
| TF2 or EXF2 | Timer 2 interrupt | 6 | 2Bh |
| TI_1 or RI_1 | Serial port 1 interrupt | 7 | 3Bh |
| INT2 | USB interrupt | 8 | 43h |
| INT3 | $I^2C$ interrupt | 9 | 4Bh |
| INT4 | External interrupt 4 | 4 | 53h |
| INT5 | External interrupt 5 | 11 | 5Bh |
| INT6 | External interrupt 6 | 12 | 63H |

#### 18.4.1.1 Interrupt Priorities

There are two stages of interrupt priority assignment, interrupt level and natural priority. The interrupt level (highest, high, or low) takes precedence over natural priority. The USB wakeup interrupt, if enabled, always has highest priority and is the only interrupt that can have highest priority. All other interrupts can be assigned either high or low priority.

In addition to an assigned priority level (high or low), each interrupt also has a natural priority, as listed in Table 18-24. Simultaneous interrupts with the same priority level (for example, both high) are resolved according to their natural priority. For example, if INT0 and INT2 are both programmed as high priority, INT0 takes precedence due to its higher natural priority.

Once an interrupt is being serviced, only an interrupt of higher priority level can interrupt the service routine of the interrupt currently being serviced.

**Table 18-25. Interrupt Flags, Enables, and Priority Control**

| Interrupt | Description | Flag | Enable | Priority Control |
|---|---|---|---|---|
| RESUME | Resume interrupt | EICON.4 | EICON.5 | N/A |
| INT0 | External interrupt 0 | TCON.1 | IE.0 | IP.0 |
| TF0 | Timer 0 interrupt | TCON.5 | IE.1 | IP.1 |
| INT1 | External interrupt 1 | TCON.3 | IE.2 | IP.2 |
| TF1 | Timer 1 interrupt | TCON.7 | IE.3 | IP.3 |
| TI_0 or RI_0 | Serial port 0 transmit or receive | SCON0.0 (RI.0), SCON0.1 (Ti_0) | IE.4 | IP.4 |
| TF2 or EXF2 | Timer 2 interrupt | T2CON.7 (TF2), T2CON.6 (EXF2) | IE.5 | IP.5 |
| TI_1 or RI_1 | Serial port 1 transmit or receive | SCON1.0 (RI_1), SCON1.1 (TI_1) | IE.6 | IP.6 |
| USB (INT2) | USB interrupt | EXIF.4 | EIE.0 | EIP.0 |
| $I^2C$ (INT3) | $I^2C$ interrupt | EXIT.5 | EIE.1 | EIP.1 |
| INT4 | External interrupt 4 | EXIF.6 | EIE.2 | EIP.2 |
| INT5 | External interrupt 5 | EXIF.7 | EIE.3 | EIP.3 |
| INT6 | External INT 6 | EICON.3 | EIE.4 | EIP.4 |

## *18.4.2 Interrupt Sampling*

The internal timers and serial ports generate interrupts by setting their respective SFR interrupt flag bits. External interrupts are sampled once per instruction cycle.

INT0 and INT1 are both active low and can be programmed to be either edge-sensitive or level-sensitive, through the IT0 and IT1 Bits in the TCON SFR. For example, when IT0 = 0, INT0 is level-sensitive and the 8051 core sets the IE0 flag when the INT0# pin is sampled low. When IT0 = 1, INT0 is edge-sensitive and the 8051 sets the IE0 flag when the INT0# pin is sampled high then low on consecutive samples.

The remaining five interrupts (INT 4-6, USB & $I^2C$ Interrupts) are edge-sensitive only. INT6 and INT4 are active high and INT5 is active low.

To ensure that edge-sensitive interrupts are detected, the corresponding ports should be held high for 4 CLKOUT cycles and then low for 4 CLKOUT cycles. Level-sensitive interrupts are not latched and must remain active until serviced.

### 18.4.3 Interrupt Latency

Interrupt response time depends on the current state of the 8051. The fastest response time is 5 instruction cycles: 1 to detect the interrupt, and 4 to perform the LCALL to the ISR.

The maximum latency (13 instruction cycles) occurs when the 8051 is currently executing a RETI instruction followed by a MUL or DIV instruction. The 13 instruction cycles in this case are: 1 to detect the interrupt, 3 to complete the RETI, 5 to execute the DIV or MUL, and 4 to execute the LCALL to the ISR. For the maximum latency case, the response time is 13 x 4 = 52 CLKOUT cycles.

### 18.4.4 Single-Step Operation

The 8051 interrupt structure provides a way to perform single-step program execution. When exiting an ISR with an RETI instruction, the 8051 will always execute at least one instruction of the task program. Therefore, once an ISR is entered, it cannot be re-entered until at least one program instruction is executed.

To perform single-step execution, program one of the external interrupts (for example,INT0) to be level-sensitive and write an ISR for that interrupt the terminates as follows:

```
JNB  TCON.1,$   ; wait for high on INT0# pin
JB   TCON.1,$   ; wait for low on INT0# pin
RETI            ; return for ISR
```

The CPU enters the ISR when the INT0# pin goes low, then waits for a pulse on INT0#. Each time INT0# is pulsed, the CPU exits the ISR, executes one program instruction, then re-enters the ISR.

## 18.5 Reset

The 8051 RESET pin is internally connected to an EZ-USB FX register bit that is controllable through the USB host. See *Chapter 13. "EZ-USB FX Resets"* for details.

## 18.6 Power Saving Modes

### 18.6.1 Idle Mode

An instruction that sets the IDLE Bit (PCON.0) causes the 8051 to enter idle mode when that instruction completes. In idle mode, CPU processing is suspended, and internal registers maintain their current data. When the 8051 core is in idle, the USB core enters suspend mode and shuts down the 24 MHz oscillator. See *Chapter 14. "EZ-USB FX Power Management"* for a full description of the Suspend/Resume process.

**Table 18-26. 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. |

*If the EZ-USB FX WAKEUP# pin is tied low, setting PCON.0 high does not put the 8051 into IDLE state.*

# EZ-USB FX Register Summary

The following table is a summary of all the EZ-USB FX Registers.

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **FIFO A-IN** | | | | | | | | | | | |
| 7800 | AINDATA | Read Data from FIFO A | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | R | R, r = read-only, |
| 7801 | AINBC | Input FIFO A Byte Count | 0 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000000 | R | |
| 7802 | AINPF | FIFO A-IN Programmable Flag (internal bit) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00100000 | RW | Default: half empty |
| 7803 | AINPFPIN | FIFO A-IN Programmable Flag (external pin) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000000 | RW | LTGT = 0: Flag is true (1) if Bytes in FIFO <= BCNT |
| 7804 | | (reserved) | | | | | | | | | | | LTGT = 1: Flag is true (1) if Bytes in FIFO >= BCNT |
| | | **FIFO B-IN** | | | | | | | | | | | Default: empty |
| 7805 | BINDATA | Read Data from FIFO B | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | R | W, w = write-only |
| 7806 | BINBC | Input FIFO B Byte Count | 0 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000000 | R | |
| 7807 | BINPF | FIFO B-IN Programmable Flag (internal bit) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00100000 | RW | Default: half empty |
| 7808 | BINPFPIN | FIFO B-IN Programmable Flag (external pin) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000000 | RW | LTGT = 0: Flag is true (1) if Bytes in FIFO <= BCNT |
| 7809 | | (reserved) | | | | | | | | | | | LTGT = 1: Flag is true (1) if Bytes in FIFO >= BCNT |
| | | **FIFO A/B-IN Control** | | | | | | | | | | | Default: empty |
| 780A | ABINCS | Input FIFOS Toggle control and flags | INTOG | INSEL | AINPF | AINEF | AINFF | BINPF | BINEF | BINFF | 01110110 | bbrrrrrr | FF=Full Flag, EF=Empty Flag, PF=Programmable Flag |
| 780B | ABINIE | Input FIFO Interrupt Enables | 0 | 0 | AINPF | AINEF | AINFF | BINPF | BINEF | BINFF | 00000000 | RW | INSEL: 1=A-FIFO, 0=B-FIFO |
| 780C | ABINIRQ | Input FIFO Interrupt Requests | 0 | 0 | AINPF | AINEF | AINFF | BINPF | BINEF | BINFF | xxxxxxxx | RW | |
| 780D | | (reserved) | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | **FIFO A-OUT** | | | | | | | | | | | |
| 780E | AOUTDATA | Load Output FIFO A | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | W | |
| 780F | AOUTBC | Output FIFO A Byte Count | 0 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000000 | R | |
| 7810 | AOUTPF | FIFO A-OUT Programmable Flag (internal bit) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 10100000 | RW | "INT" suffix means internal, 8051-accessible bits. Default: half-full |
| 7811 | AOUTPFPIN | FIFO A-OUT Programmable Flag (external pin) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 11000000 | RW | Default: full |
| 7812 | | (reserved) | | | | | | | | | | | |
| | | **FIFO B-OUT** | | | | | | | | | | | |
| 7813 | BOUTDATA | Load Output FIFO B | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | W | RW = Read or Write, |
| 7814 | BOUTBC | Output FIFO B Byte Count | 0 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000000 | R | |
| 7815 | BOUTPF | FIFO B-OUT Programmable Flag (internal bit) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 10100000 | RW | "PIN" suffix means external pin flags. Default: half-full |
| 7816 | BOUTPFPIN | FIFO B-OUT Programmable Flag (external pin) | LTGT | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 11000000 | RW | Default: full |
| 7817 | | (reserved) | | | | | | | | | | | |
| | | **FIFO A/B OUT Control** | | | | | | | | | | | |
| 7818 | ABOUTCS | Output FIFOS Toggle control and flags | OUTTOG | OUTSEL | AOUTPF | AOUTEF | AOUTFF | BOUTPF | BOUTEF | BOUTFF | 01010010 | RW | PF=Programmable Flag, |
| 7819 | ABOUTIE | Output FIFO Interrupt Enables | 0 | 0 | AOUTPF | AOUTEF | AOUTFF | BOUTPF | BOUTEF | BOUTFF | 00000000 | RW | |
| 781A | ABOUTIRQ | Output FIFO Interrupt Requests | 0 | 0 | AOUTPF | AOUTEF | AOUTFF | BOUTPF | BOUTEF | BOUTFF | xxxxxxxx | RW | |
| 781B | | (reserved) | | | | | | | | | | | |
| | | **FIFO A/B Global Control** | | | | | | | | | | | |

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 781C | ABSETUP | FIFO Setup | 0 | 0 | ASYNC | DBLIN | 0 | OUTDLY | 0 | DBLOUT | 00000000 | RW | ASYNC=1: async FIFOS, DBL=1:double stuff,OUTDLY=1: clock delay |
| 781D | ABPOLAR | FIFO Control Signals Polarity | 0 | 0 | BOE | AOE | SLRD | SLWR | ASEL | BSEL | 00000000 | RW | 0=active LO, 1-active HI |
| 781E | ABFLUSH | Write (data=x) to reset all flags | x | x | x | x | x | x | x | x | xxxxxxxx | W | Flag reset: Empty=1, Full=0, PF=? |
| 781F | | (reserved) | | | | | | | | | | | |
| 7820 | | (reserved) | | | | | | | | | | | |
| 7821 | | (reserved) | | | | | | | | | | | |
| 7822 | | (reserved) | | | | | | | | | | | |
| 7823 | | (reserved) | | | | | | | | | | | |
| | | **GPIF** | | | | | | | | | | | |
| 7824 | WFSELECT | Waveform Selector | SINGLEWR 0-3 | | SINGLERD 0-3 | | FIFOWR 0-3 | | FIFORD 0-3 | | 11100100 | RW | Select waveform 0[00], 1[01], 2[10] or 3[11] |
| 7825 | IDLE_CS | GPIF Done, GPIF IDLE drive mode | DONE | 0 | 0 | 0 | 0 | 0 | 0 | IDLEDRV | 10000000 | RW | DONE=1: GPIF done (IRQ4). IDLEDRV=1:drive bus, 0:TS |
| 7826 | IDLE_CTLOUT | Inactive Bus, CTL states | 0 | 0 | CTL5 | CTL4 | CTL3 | CTL2 | CTL1 | CTL0 | 11111111 | RW | |
| 7827 | CTLOUTCFG | CTL OUT pin drive | TRICTL | 0 | CTL5 | CTL4 | CTL3 | CTL2 | CTL1 | CTL0 | 00000000 | RW | 0=CMOS, 1=open drn. |
| 7828 | | (reserved) | | | | | | | | | | | |
| 7829 | | (reserved) | | | | | | | | | | | |
| 782A | GPIFADRL | GPIF Address L | x | x | A5 | A4 | A3 | A2 | A1 | A0 | 00000000 | RW | |
| 782B | | (reserved) | | | | | | | | | | | |
| 782C | AINTC | FIFO A IN T.C. | FITC | FIFO A IN Transaction Count [6:0] | | | | | | | 00000001 | RW | FITC=1: Use FIFO flags |
| 782D | AOUTTC | FIFO A OUT T.C. | FITC | FIFO A OUT Transaction Count [6:0] | | | | | | | 00000001 | RW | FITC=0: Use Transac Count. |
| 782E | ATRIG | ***Write:*** write FIFO A. ***Read:*** start RD trans. | x | x | x | x | x | x | x | x | xxxxxxxx | RW | Note: read the data in ? |
| 782F | | (reserved) | | | | | | | | | | | |
| 7830 | BINTC | FIFO B IN T.C. | FITC | FIFO B IN Transaction Count [6:0] | | | | | | | 00000001 | RW | |
| 7831 | BOUTTC | FIFO B OUT T.C. | FITC | FIFO B OUT Transaction Count [6:0] | | | | | | | 00000001 | RW | |
| 7832 | BTRIG | ***Write***: write FIFO B***. Read:*** start RD trans. | x | x | x | x | x | x | x | x | xxxxxxxx | RW | ? |
| 7833 | | (reserved) | | | | | | | | | | | |
| 7834 | SGLDATH | GPIF Data H (16-bit mode only) | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | xxxxxxxx | RW | |
| 7835 | SGLDATLTRIG | Read or Write GPIF Data L & trigger rd transac | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW | Triggers a GPIF Read Waveform |
| 7836 | SGLDATLNTRIG | Read GPIF Data L, no rd transac trigger | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | R | No GPIF Waveform |
| 7837 | | (reserved) | | | | | | | | | | | |
| 7838 | READY | Internal RDY,Sync/Async, RDY pin states | INTRDY | SAS | RDY5 | RDY4 | RDY3 | RDY2 | RDY1 | RDY0 | 00xxxxxx | bbrrrrrr | SAS=1: synchrnous, 0:asynchronous |
| 7839 | ABORT | Abort GPIF cycles | x | x | x | x | x | x | x | x | xxxxxxxx | W | Go To GPIF IDLE state. Data is D.C. |
| 783A | | (reserved) | | | | | | | | | | | |
| 783B | GENIE | | 0 | 0 | 0 | 0 | 0 | DMADN | GPWF | GPDONE | 00000000 | RW | |
| 783C | GENIRQ | | 0 | 0 | 0 | 0 | 0 | DMADN | GPWF | GPDONE | 00000xxx | RW | |
| 783D | | (reserved) | | | | | | | | | | | |
| 783E | | (reserved) | | | | | | | | | | | |

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 783F | | (reserved) | | | | | | | | | | | |
| 7840 | | (reserved) | | | | | | | | | | | |
| | | **IO Ports D,E** | | | | | | | | | | | |
| 7841 | OUTD | Output Port D | OUTD7 | OUTD6 | OUTD5 | OUTD4 | OUTD3 | OUTD2 | OUTD1 | OUTD0 | xxxxxxxx | W | |
| 7842 | PINSD | Input Port D pins | PIND7 | PIND6 | PIND5 | PIND4 | PIND3 | PIND2 | PIND1 | PIND0 | xxxxxxxx | R | |
| 7843 | OED | Port D Output Enable | 0ED7 | 0ED6 | 0ED5 | 0ED4 | 0ED3 | 0ED2 | 0ED1 | 0ED0 | 00000000 | RW | |
| 7844 | | (reserved) | | | | | | | | | | | |
| 7845 | OUTE | Output Port E | OUTE7 | OUTE6 | OUTE5 | OUTE4 | OUTE3 | OUTE2 | OUTE1 | OUTE0 | xxxxxxxx | W | |
| 7846 | PINSE | Input Port E pins | PINE7 | PINE6 | PINE5 | PINE4 | PINE3 | PINE2 | PINE1 | PINE0 | xxxxxxxx | R | |
| 7847 | OEE | Port E Output Enable | OEE7 | OEE6 | OEE5 | OEE4 | OEE3 | OEE2 | OEE1 | OEE0 | 00000000 | RW | |
| 7848 | | (reserved) | | | | | | | | | | | |
| 7849 | PORTSETUP | Timer0 Clock source, Port-to-SFR mapping | 0 | 0 | 0 | 0 | 0 | 0 | T0CLK | SFRPORT | 00000000 | RW | T0CLK (0) Normal Timer0 clock; (1) CPU clock/13 |
| 784A | IFCONFIG | Select 8/16 bit data bus, confugure busses (IF) | *52ONE* | 0 | 0 | | GSTATE | BUS16 | IF1 | IF0 | 00000000 | brrrbbbb | SFRPORT (1) IO Ports SFR mapped, (0) not |
| 784B | PORTACF2 | Port A GPIF signals | 0 | 0 | SLRD | SLWR | 0 | 0 | 0 | 0 | 00000000 | RW | *52ONE*: Set to 1 for 52-pin part (drive internal RDY signals HI) |
| 784C | PORTCCF2 | Port C GPIF signals | CTL5 | CTL4 | CTL3 | CTL1 | RDY3 | 0 | RDY1 | RDY0 | 00000000 | RW | |
| 784D | | (reserved) | | | | | | | | | | | |
| 784E | | (reserved) | | | | | | | | | | | |
| | | **DMA Control** | | | | | | | | | | | |
| 784F | DMASRCH | DMA SourceH | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | 00000000 | RW | |
| 7850 | DMASRCL | DMA Source L | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | 00000000 | RW | |
| 7851 | DMADESTH | DMA Destination H | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | 00000000 | RW | |
| 7852 | DMADESTL | DMA Destination L | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | 00000000 | RW | |
| 7853 | | (reserved) | | | | | | | | | | | |
| 7854 | DMALEN | DMA Transfer Length | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 00000001 | RW | 0=256, 1=1…255=255 |
| 7855 | DMAGO | Start DMA Transfer | DONE | x | x | x | x | x | x | x | xxxxxxxx | rxxxxxxx | Write this register to start a DMA transfer |
| 7856 | | (reserved) | | | | | | | | | | | DSTR[2..0] set stretch values for external DMA transfers |
| 7857 | DMABURST | | x | x | x | DSTR2 | DSTR1 | DSTR0 | RB | WB | 00000100 | RW | RB/WB enable synchronous burst transfers on 8051 data bus |
| 7858 | DMAEXTFIFO | | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | xxxxxxxx | N/A | Use this DMA address to select 8051 A/D busses as external FIFO |
| 7859 | | (reserved) | | | | | | | | | | | |
| 785A | | (reserved) | | | | | | | | | | | Note: DSTRn are stretch values for DMA FRD# and FWR# signals |
| 785B | | (reserved) | | | | | | | | | | | |
| 785C | | (reserved) | | | | | | | | | | | |
| | | **Interrupt 4 Vector Control** | | | | | | | | | | | |
| 785D | INT4IVEC | Interrupt 4 Vector | 0 | 1 | I4V3 | I4V2 | I4V1 | I4V0 | 0 | 0 | 01000000 | R | See bottom of page 2 for vector coding |
| 785E | INT4SETUP | Interrupt 4 Setup | 0 | 0 | 0 | 0 | 0 | INT4SFC | INTERNAL | AV4EN | 00000000 | RW | INTERNAL: 0-INT4 from pin, 1-INT4 from FIFO unit |
| 785F-78FF | | (reserved) | | | | | | | | | | | |

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| 7900 | WFDESC(0) | **Waveform Descriptors (128 bytes)** | | | | | | | | | xxxxxxxx | RW | | | |
| 797F | WFDESC(127) | (last wafeform descriptor byte) | | | | | | | | | | | | | |
| 7980-7B3F | | (reserved) | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | **Endpoint 0-7 Data Buffers** | | | | | | | | | | | R=RD, W=WR, RW or b=BOTH | | |
| 7B40 | OUT7BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7B80 | IN7BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7BC0 | OUT6BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | [6] EP6OUT and EP7OUT pairable for double buffering | | |
| 7C00 | IN6BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | [5] EP6IN and EP7IN pairable for double buffering | | |
| 7C40 | OUT5BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7C80 | IN5BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7CC0 | OUT4BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | [4] EP4OUT and EP5OUT pairable for double buffering | | |
| 7D00 | IN4BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | [3] EP4IN and EP5IN pairable for double buffering | | |
| 7D40 | OUT3BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7D80 | IN3BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7DC0 | OUT2BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | [2] EP2-IN and EP3IN pairable for double buffering | | |
| 7E00 | IN2BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | [1] EP2-IN and EP3IN pairable for double buffering | | |
| 7E40 | OUT1BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7E80 | IN1BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7EC0 | OUT0BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7F00 | IN0BUF | (64 bytes) | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | | | |
| 7F40-7F5F | | (reserved) | | | | | | | | | | | | | |
| | | **Isochronous Data** | | | | | | | | | | | | | |
| 7F60 | OUT8DATA | Endpoint 8 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F61 | OUT9DATA | Endpoint 9 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F62 | OUT10DATA | Endpoint 10 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F63 | OUT11DATA | Endpoint 11 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F64 | OUT12DATA | Endpoint 12 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F65 | OUT13DATA | Endpoint 13 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F66 | OUT14DATA | Endpoint 14 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F67 | OUT15DATA | Endpoint 15 OUT Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F68 | IN8DATA | Endpoint 8 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| 7F69 | IN9DATA | Endpoint 9 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| 7F6A | IN10DATA | Endpoint 10 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| 7F6B | IN11DATA | Endpoint 11 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| 7F6C | IN12DATA | Endpoint 12 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7F6D | IN13DATA | Endpoint 13 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| 7F6E | IN14DATA | Endpoint 14 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| 7F6F | IN15DATA | Endpoint 15 IN Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | W | | | |
| | | **Isochronous Byte Counts** | | | | | | | | | | | | | |
| 7F70 | OUT8BCH | EP8 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F71 | OUT8BCL | EP8 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F72 | OUT9BCH | EP9 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F73 | OUT9BCL | EP9 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F74 | OUT10BCH | EP10 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F75 | OUT10BCL | EP10 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F76 | OUT11BCH | EP11 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F77 | OUT11BCL | EP11 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F78 | OUT12BCH | EP12 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F79 | OUT12BCL | EP12 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F7A | OUT13BCH | EP13 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F7B | OUT13BCL | EP13 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F7C | OUT14BCH | EP14 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F7D | OUT14BCL | EP14 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F7E | OUT15BCH | EP15 Out Byte Count H | 0 | 0 | 0 | 0 | 0 | 0 | d9 | d8 | xxxxxxxx | R | | | |
| 7F7F | OUT15BCL | EP15 Out Byte Count L | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | | | |
| 7F80-7F91 | | (reserved) | | | | | | | | | | | | | |
| | | **CPU Registers** | | | | | | | | | | | | | |
| 7F92 | CPUCS | Control & Status | rv3 | rv2 | rv1 | rv0 | 24/48 | CLKINV | CLKOE | 8051RES | 00000010 | rrrrrrbr | Bits 7:2 are chip rev number | | |
| 7F93 | PORTACFG | Port A Configuration | RxD1out | RxD0out | FRD | FWR | CS | OE | T1out | T0out | 00000000 | RW | CFG: 0=port (default), 1=alternate function | | |
| 7F94 | PORTBCFG | Port B Configuration | T2OUT | INT6 | INT5 | INT4 | TxD1 | RxD1 | T2EX | T2 | 00000000 | RW | | | |
| 7F95 | PORTCCFG | Port C Configuration | RD | WR | T1 | T0 | INT1 | INT0 | TxD0 | RxD0 | 00000000 | RW | | | |
| | | **Input-Output Port Registers** | | | | | | | | | | | | | |
| 7F96 | OUTA | Output Register A | OUTA7 | OUTA6 | OUTA5 | OUTA4 | OUTA3 | OUTA2 | OUTA1 | OUTA0 | 00000000 | RW | WR: output FF; RD: register outputs | | |
| 7F97 | OUTB | Output Register B | OUTB7 | OUTB6 | OUTB5 | OUTB4 | OUTB3 | OUTB2 | OUTB1 | OUTB0 | 00000000 | RW | | | |
| 7F98 | OUTC | Output Register C | OUTC7 | OUTC6 | OUTC5 | OUTC4 | OUTC3 | OUTC2 | OUTC1 | OUTC0 | 00000000 | RW | | | |
| 7F99 | PINSA | Port Pins A | PINA7 | PINA6 | PINA5 | PINA4 | PINA3 | PINA2 | PINA1 | PINA0 | xxxxxxxx | R | WR: no effect; RD: pin states (reg if OE=1, pin if OE=0) | | |
| 7F9A | PINSB | Port Pins B | PINB7 | PINB6 | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 | xxxxxxxx | R | | | |
| 7F9B | PINSC | Port Pins C | PINC7 | PINC6 | PINC5 | PINC4 | PINC3 | PINC2 | PINC1 | PINC0 | xxxxxxxx | R | | | |
| 7F9C | OEA | Output Enable A | OEA7 | OEA6 | OEA5 | OEA4 | OEA3 | OEA2 | OEA1 | OEA0 | 00000000 | RW | 1=output enabled | | |
| 7F9D | OEB | Output Enable B | OEB7 | OEB6 | OEB5 | OEB4 | OEB3 | OEB2 | OEB1 | OEB0 | 00000000 | RW | | | |
| 7F9E | OEC | Output Enable C | OEC7 | OEC6 | OEC5 | OEC4 | OEC3 | OEC2 | OEC1 | OEC0 | 00000000 | RW | RW pins enabled | | |
| 7F9F | | reserve | | | | | | | | | | | | | |

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Isochronous Control/Status Registers** | | | | | | | | | | | |
| 7FA0 | ISOERR | ISO OUT Endpoint Error | ISO15ERR | ISO14ERR | ISO13ERR | ISO12ERR | ISO11ERR | ISO10ERR | ISO9ERR | ISO8ERR | xxxxxxxx | R | CRC Error |
| 7FA1 | ISOCTL | Isochronous Control | * | * | * | * | PPSTAT | MBZ | MBZ | ISODISAB | 0000x000 | rrrrrbbb | |
| 7FA2 | ZBCOUT | Zero Byte Count bits | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | xxxxxxxx | R | |
| 7FA3 | | (reserved) | | | | | | | | | | | |
| 7FA4 | | (reserved) | | | | | | | | | | | |
| | | **I²C Registers** | | | | | | | | | | | |
| 7FA5† | I2CS | Control & Status | START | STOP | LASTRD | ID1 | ID0 | BERR | ACK | DONE | 000xx000 | bbbrrrrr | |
| 7FA6† | I2DAT | Data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FA7† | I2CMODE | I2C STOP interrupt enable | 0 | 0 | 0 | 0 | 0 | 0 | STOPIE | 400kHz | 00000000 | RW | |
| | | **Interrupts** | | | | | | | | | | | |
| 7FA8 | IVEC | Interrupt Vector | 0 | IV4 | IV3 | IV2 | IV1 | IV0 | 0 | 0 | 00000000 | R | |
| 7FA9† | IN07IRQ | EPIN Interrupt Request | IN7IR | IN6IR | IN5IR | IN4IR | IN3IR | IN2IR | IN1IR | IN0IR | 00000000 | RW | 1 = clear request, 0= no effect |
| 7FAA† | OUT07IRQ | EPOUT Interrupt Request | OUT7IR | OUT6IR | OUT5IR | OUT4IR | OUT3IR | OUT2IR | OUT1IR | OUT0IR | xxxxxxxx | RW | 1 = clear request, 0= no effect |
| 7FAB† | USBIRQ | USB Interrupt Request | 0 | 0 | IBNIR | URESIR | SUSPIR | SUTOKIR | SOFIR | SUDAVIR | xxxxxxxx | RW | 1 = clear request, 0= no effect |
| 7FAC† | IN07IEN | EP0-7IN Int Enables | IN7IEN | IN6IEN | IN5IEN | IN4IEN | IN3IEN | IN2IEN | IN1IEN | IN0IEN | 00000000 | RW | 1=enabled, 0=disabled |
| 7FAD† | OUT07IEN | EP0-7OUT Int Enables | OUT7IEN | OUT6IEN | OUT5IEN | OUT4IEN | OUT3IEN | OUT2IEN | OUT1IEN | OUT0IEN | 00000000 | RW | 1=enabled, 0=disabled |
| 7FAE† | USBIEN | USB Int Enables | 0 | 0 | IBNIE | URESIE | SUSPIE | SUTOKIE | SOFIE | SUDAVIE | 00000000 | RW | 1=enabled, 0=disabled |
| 7FAF | USBBAV | Breakpoint & Autovector | * | * | * | INT2SFC | BREAK | BPPULSE | BPEN | AVEN | xxx0xx00 | RW | Breakpoint and Autovector |
| 7FB0† | IBNIRQ | IBN Interrupt request | | EP6IN | EP5IN | EP4IN | EP3IN | EP2IN | EP1IN | EP0IN | xxxxxxxx | RW | |
| 7FB1† | IBNIE | IBN Interrupt Enable | | EP6IN | EP5IN | EP4IN | EP3IN | EP2IN | EP1IN | EP0IN | 00000000 | RW | |
| 7FB2 | BPADDRH | Breakpoint Address H | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | 00000000 | RW | |
| 7FB3 | BPADDRL | Breakpoint Address L | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | 00000000 | RW | |
| | | **Bulk Endpoints 0-7** | | | | | | | | | | | |
| 7FB4† | EP0CS | Control & Status | * | * | * | * | OUTBSY | INBSY | HSNAK | EP0STALL | 00001000 | rrrrrrbb | |
| 7FB5† | IN0BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FB6† | IN1CS | Control & Status | * | * | * | * | * | * | in1bsy | in1stl | 00000000 | rrrrrrbb | |
| 7FB7† | IN1BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FB8† | IN2CS | Control & Status | * | * | * | * | * | * | in2bsy | in2stl | 00000000 | rrrrrrbb | |
| 7FB9† | IN2BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FBA† | IN3CS | Control & Status | * | * | * | * | * | * | in3bsy | in3stl | 00000000 | rrrrrrbb | |
| 7FBB† | IN3BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FBC† | IN4CS | Control & Status | * | * | * | * | * | * | in4bsy | in4stl | 00000000 | rrrrrrbb | |
| 7FBD† | IN4BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FBE† | IN5CS | Control & Status | * | * | * | * | * | * | in5bsy | in5stl | 00000000 | rrrrrrbb | |
| 7FBF† | IN5BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7FC0† | IN6CS | Control & Status | * | * | * | * | * | * | in6bsy | in6stl | 00000000 | rrrrrrbb | |
| 7FC1† | IN6BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FC2† | IN7CS | Control & Status | * | * | * | * | * | * | in7bsy | in7stl | 00000000 | rrrrrrbb | |
| 7FC3† | IN7BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FC4 | | (reserved) | | | | | | | | | | | |
| 7FC5† | OUT0BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FC6† | OUT1CS | Control & Status | * | * | * | * | * | * | out1bsy | out1stl | 00000010 | rrrrrrrb | |
| 7FC7† | OUT1BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FC8† | OUT2CS | Control & Status | * | * | * | * | * | * | out2bsy | out2stl | 00000010 | rrrrrrrb | |
| 7FC9† | OUT2BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FCA† | OUT3CS | Control & Status | * | * | * | * | * | * | out3bsy | out3stl | 00000010 | rrrrrrrb | |
| 7FCB† | OUT3BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FCC† | OUT4CS | Control & Status | * | * | * | * | * | * | out4bsy | out4stl | 00000010 | rrrrrrrb | |
| 7FCD† | OU4TBC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FCE† | OUT5CS | Control & Status | * | * | * | * | * | * | out5bsy | out5stl | 00000010 | rrrrrrrb | |
| 7FCF† | OUT5BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FD0† | OUT6CS | Control & Status | * | * | * | * | * | * | out6bsy | out6stl | 00000010 | rrrrrrrb | |
| 7FD1† | OUT6BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| 7FD2† | OUT7CS | Control & Status | * | * | * | * | * | * | out7bsy | out7stl | 00000010 | rrrrrrrb | |
| 7FD3† | OUT7BC | Byte Count | * | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | RW | |
| | | **Global USB Registers** | | | | | | | | | | | |
| 7FD4† | SUDPTRH | Setup Data Ptr H | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | xxxxxxxx | RW | |
| 7FD5† | SUDPTRL | Setup Data Ptr L | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | xxxxxxxx | RW | |
| 7FD6† | USBCS | USB Control & Status | WakeSRC | * | * | * | DisCon | DiscOE | ReNum | SIGRSUME | 00000100 | brrrbbbb | |
| 7FD7† | TOGCTL | Toggle Control | Q | S | R | IO | 0 | EP2 | EP1 | EP0 | xxxxxxxx | rbbbbbbb | Endpoint Toggle bits |
| 7FD8 | USBFRAMEL | Frame Number L | FC7 | FC6 | FC5 | FC4 | FC3 | FC2 | FC1 | FC0 | xxxxxxxx | R | |
| 7FD9 | USBFRAMEH | Frame Number H | 0 | 0 | 0 | 0 | 0 | FC10 | FC9 | FC8 | xxxxxxxx | R | |
| 7FDA | | (reserved) | | | | | | | | | | | |
| 7FDB | FNADDR | Function Address | 0 | FA6 | FA5 | FA4 | FA3 | FA2 | FA1 | FA0 | xxxxxxxx | R | |
| 7FDC | | (reserved) | | | | | | | | | | | |
| 7FDD† | USBPAIR | Endpoint Control | ISOsend0 | * | PR6OUT | PR4OUT | PR2OUT | PR6IN | PR4IN | PR2IN | 0x000000 | RW | 1 = Pair endpoint (double-buffer) |
| 7FDE† | IN07VAL | Input Endpoint 0-7 valid | IN7VAL | IN6VAL | IN5VAL | IN4VAL | IN3VAL | IN2VAL | IN1VAL | 1 | 01010111 | RW | 1=valid, 0=not valid |
| 7FDF† | OUT07VAL | Output Endpoint 0-7 valid | OUT7VAL | OUT6VAL | OUT5VAL | OUT4VAL | OUT3VAL | OUT2VAL | OUT1VAL | 1 | 01010101 | RW | (a not-valid EP returns 'no response' instead of NAK) |
| 7FE0† | INISOVAL | Input EP 8-15 valid | IN15VAL | IN14VAL | IN13VAL | IN12VAL | IN11VAL | IN10VAL | IN9VAL | IN8VAL | 00000111 | RW | NOTE: EP0 is always valid |
| 7FE1† | OUTISOVAL | Output EP 8-15 valid | OUT15VAL | OUT14VAL | OUT13VAL | OUT12VAL | OUT11VAL | OUT10VAL | OUT9VAL | OUT8VAL | 00000111 | RW | |
| 7FE2 | FASTXFR | Fast Transfer Mode | FISO | FBLK | RPOL | RMOD1 | RMOD0 | WPOL | WMOD1 | WMOD0 | xxxxxxxx | RW | |

† **Read/write latency note:** These registers need the equivalent of 2 instruction clock cycles of time between performing the following instructions back-to-back: (1) write-write (2) write-read.

# EZ-USB FX Register Summary

| Addr | Name | Description | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Default | Access | Notes | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7FE3 | AUTOPTRH | Auto-Pointer H | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | xxxxxxxx | RW | | | |
| 7FE4 | AUTOPTRL | Auto-Pointer L | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | xxxxxxxx | RW | | | |
| 7FE5 | AUTODATA | Auto Pointer Data | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW | | | |
| 7FE6 | | (reserved) | | | | | | | | | | | | | |
| 7FE7 | | (reserved) | | | | | | | | | | | | | |
| | | **Setup Data** | | | | | | | | | | | | | |
| 7FE8 | SETUPDAT | 8 bytes of SETUP data | d7 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | xxxxxxxx | R | 3 LSB's of address are 000 | | |
| | | **Isochronous FIFO Sizes** | | | | | | | | | | | | | |
| 7FF0 | OUT8ADDR | Endpt 8 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF1 | OUT9ADDR | Endpt 9 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF2 | OUT10ADDR | Endpt 10 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF3 | OUT11ADDR | Endpt 11 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF4 | OUT12ADDR | Endpt 12 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF5 | OUT13ADDR | Endpt 13 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF6 | OUT14ADDR | Endpt 14 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF7 | OUT15ADDR | Endpt 15 OUT Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF8 | IN8ADDR | Endpt 8 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FF9 | IN9ADDR | Endpt 9 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FFA | IN19ADDR | Endpt 10 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FFB | IN11ADDR | Endpt 11 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FFC | IN12ADDR | Endpt 12 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FFD | IN13ADDR | Endpt 13 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FFE | IN14ADDR | Endpt 14 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| 7FFF | IN15ADDR | Endpt 15 IN Start Addr | A9 | A8 | A7 | A6 | A5 | A4 | 0 | 0 | xxxxxxxx | RW | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Interrupt 4 Sources and INT4IVEC values** | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | 20 | AINPF | **0** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | | | | | |
| | 21 | BINPF | **0** | **1** | 0 | 0 | 0 | 1 | 0 | 0 | | | | | |
| | 22 | AOUTPF | **0** | **1** | 0 | 0 | 1 | 0 | 0 | 0 | | | | | |
| | 23 | BOUTPF | **0** | **1** | 0 | 0 | 1 | 1 | 0 | 0 | | | | | |
| | 24 | AINEF | **0** | **1** | 0 | 1 | 0 | 0 | 0 | 0 | | | | | |
| | 25 | BINEF | **0** | **1** | 0 | 1 | 0 | 1 | 0 | 0 | | | | | |
| | 26 | AOUTEF | **0** | **1** | 0 | 1 | 1 | 0 | 0 | 0 | | | | | |
| | 27 | BOUTEF | **0** | **1** | 0 | 1 | 1 | 1 | 0 | 0 | | | | | |
| | 28 | AINFF | **0** | **1** | 1 | 0 | 0 | 0 | 0 | 0 | | | | | |
| | 29 | BINFF | **0** | **1** | 1 | 0 | 0 | 1 | 0 | 0 | | | | | |
| | 2A | AOUTFF | **0** | **1** | 1 | 0 | 1 | 0 | 0 | 0 | | | | | |
| | 2B | BOUTFF | **0** | **1** | 1 | 0 | 1 | 1 | 0 | 0 | | | | | |
| | 2C | GPIFDONE | **0** | **1** | 1 | 1 | 0 | 0 | 0 | 0 | | | | | |
| | 2D | GPIFWF | **0** | **1** | 1 | 1 | 0 | 1 | 0 | 0 | | | | | |
| | 2E | DMADONE | **0** | **1** | 1 | 1 | 1 | 0 | 0 | 0 | | | | | |
| | | | | | | | | | | | | | | | |
| | | **New Interrupt 2 source and vector** | | | | | | | | | | | | | |
| | 05 | IBN | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | | | Bulk IN token arrived and was NAK'd | | |