NetChip
Technology, Inc.

335 Pioneer Way
Mountain View, California 94041
Tel (650) 526-1490
Fax (650) 526-1494
e-mail: sales@netchip.com
Internet: www.netchip.com

NET2280 PCI USB 2.0 High Speed Peripheral Controller

Rev 1A Device

Doc #: 605-0177-0120
Revision: 1.2
Date: March 17, 2003

Revision History

Revision	Issue Date	Comments
1.0	May 5, 2002	Initial Release
1.1	November 11, 2002	Update power consumption values
1.2	March 17, 2003	Rev 1A release

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

NET2280 PCI/USB 2.0 Controller

1 INTRODUCTION 10

1.1 FEATURES 10
1.2 FEATURE OVERVIEW 10
1.3 OPERATION OVERVIEW 13
1.3.1 Applications 13
1.3.2 Initialization 13
1.3.3 PCI Interface 13
1.3.4 USB Interface 13
1.3.5 Interrupts 14
1.4 NET2280 BLOCK DIAGRAM 14
1.5 NET2280 TYPICAL ADAPTER MODE BLOCK DIAGRAM 15
1.6 NET2280 TYPICAL HOST MODE BLOCK DIAGRAM 15
1.7 EXAMPLE CONNECTIONS TO NET2280 16
1.7.1 Example Part Numbers 17
1.7.2 General PCB Layout Guidelines 17
1.7.2.1 USB Differential Signals 17
1.7.2.2 Analog VDD (power) 17
1.7.2.3 Analog VSS (ground) 18
1.7.2.4 Decoupling Capacitors 18
1.7.2.5 EMI Noise Suppression 18
1.8 TERMINOLOGY 19

2 PIN DESCRIPTION 20

2.1 DIGITAL POWER AND GROUND (26 PINS) 20
2.2 USB TRANSCEIVER (15 PINS) 21
2.3 CLOCKS, RESET, MISC (27 PINS) 22
2.4 PCI BUS (52 PINS) 24
2.5 PHYSICAL PIN ASSIGNMENT 27

3 RESET AND INITIALIZATION 28

3.1 OVERVIEW 28
3.2 RESET# PIN 28
3.3 PCI RST# PIN 28
3.4 ROOT PORT RESET 28
3.5 SOFT RESETS 28
3.6 RESET SUMMARY 29
3.7 INITIALIZATION SUMMARY 29

4 EEPROM 30

4.1 OVERVIEW 30
4.2 EEPROM DATA FORMAT 30
4.3 INITIALIZATION 31
4.4 EEPROM RANDOM READ/WRITE ACCESS 31
4.4.1 EEPROM Opcodes 31
4.4.2 EEPROM Low-Level Access Routines 32
4.4.3 EEPROM Read Status Routine 33
4.4.4 EEPROM Write Data Routine 33
4.4.5 EEPROM Read Data Routine 33

5 8051 CPU 34

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

5.1 OVERVIEW....34
5.2 8051 MEMORY MAP....34
5.2.1 Program Space (64 Kbytes)....34
5.2.2 External Data Space (64 Kbytes)....34
5.2.3 Internal Data Space (256 bytes)....34
5.2.3.1 Internal RAM (256 Bytes)....34
5.2.3.2 Special Function Registers....35
5.2.4 PCI Master Cycles....36
5.3 8051 INTERRUPTS....37

6 PCI INTERFACE....38

6.1 OVERVIEW....38
6.2 CONFIGURATION TRANSACTIONS....38
6.3 INITIATOR TRANSACTIONS....39
6.4 TARGET TRANSACTIONS....39
6.5 BUS ARBITRATION....40
6.5.1 Overview....40
6.5.2 External Arbiter Mode....40
6.5.3 Internal Arbiter Mode....40
6.5.4 Arbitration Parking....40

7 USB FUNCTIONAL DESCRIPTION....41

7.1 USB INTERFACE....41
7.2 USB PROTOCOL....41
7.2.1 Tokens....41
7.2.2 Packets....41
7.2.3 Transaction....42
7.2.4 Transfer....42
7.3 AUTOMATIC RETRIES....42
7.3.1 Out Transactions....42
7.3.2 In Transactions....42
7.4 PING FLOW CONTROL....42
7.5 PACKET SIZES....42
7.6 USB ENDPOINTS....43
7.6.1 Control Endpoint - Endpoint 0....43
7.6.1.1 Control Write Transfer....43
7.6.1.2 Control Write Transfer Details....44
7.6.1.3 Control Read Transfer....45
7.6.1.4 Control Read Transfer Details....45
7.6.1.5 Auto-Enumerate....47
7.6.2 Isochronous Endpoints....47
7.6.2.1 Isochronous Out Transactions....48
7.6.2.2 High Bandwidth Isochronous OUT Transactions....48
7.6.2.3 Isochronous In Transactions....49
7.6.2.4 High Bandwidth Isochronous IN Transactions....49
7.6.3 Bulk Endpoints....50
7.6.3.1 Bulk Out Transactions....50
7.6.3.2 Bulk In Endpoints....51
7.6.4 Interrupt Endpoints....52
7.6.4.1 Interrupt Out Transactions....52
7.6.4.2 Interrupt In Endpoints....52
7.6.4.3 High Bandwidth INTERRUPT Endpoints....52
7.6.5 Dedicated Endpoints....53
7.6.5.1 CFGOUT Endpoint....53
7.6.5.2 CFGIN Endpoint....54

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.5.3 PCIOUT Endpoint 55
7.6.5.4 PCIIN Endpoint 56
7.6.5.5 STATIN Endpoint....56
7.7 FIFOs....57
7.7.1 IN Endpoint FIFOs....57
7.7.2 OUT Endpoint FIFOs....58
7.8 USB TEST MODES....59

8 DMA CONTROLLER....60

8.1 OVERVIEW....60
8.2 SINGLE TRANSFER MODE....60
8.2.1 OUT Endpoints....60
8.2.2 IN Endpoints....60
8.3 SCATTER/GATHER MODE....61
8.3.1 Valid Bit....61
8.3.2 Clear Count Enable....61
8.3.3 Direction....61
8.3.4 Done Interrupt Enable....62
8.3.5 End of Chain....62
8.4 OUT TRANSFER DMA COMPLETION....62
8.5 DMA ABORT....63
8.6 DMA PAUSE....63
8.7 PCI UNALIGNED WRITE TRANSFERS....63
8.7.1 Restrictions....63
8.8 PCI UNALIGNED READ TRANSFERS....64
8.8.1 Restrictions....64

9 INTERRUPT AND STATUS REGISTER OPERATION....65

9.1 OVERVIEW....65
9.2 INTERRUPT STATUS REGISTERS (IRQSTAT0 AND IRQSTAT1)....65
9.3 ENDPOINT RESPONSE REGISTERS (EP_RSP)....65
9.4 ENDPOINT STATUS REGISTER (EP_STAT)....65

10 POWER MANAGEMENT....66

10.1 OVERVIEW....66
10.2 USB POWER CONFIGURATIONS....66
10.2.1 Self-Powered Device....66
10.3 USB SUSPEND MODE....66
10.4 PCI POWER MANAGEMENT....67
10.4.1 Power States....67
10.5 USB SUSPEND/RESUME FOR PCI HOST MODE....67
10.5.1 Suspend Sequence....67
10.5.2 Host Initiated Wake-Up....68
10.5.3 Device-Remote Wake-Up....68
10.5.4 Resume Interrupt....68
10.6 USB SUSPEND/RESUME FOR PCI ADAPTER MODE....68
10.6.1 Suspend Sequence with 8051 held in reset....68
10.6.2 Host Initiated Wake-Up with 8051 held in reset....68
10.6.3 Suspend Sequence with 8051 operating....68
10.6.4 Host Initiated Wake-Up with 8051 operating....69
10.6.5 PME Isolation....69
10.7 NET2280 LOW-POWER MODES....70
10.7.1 USB Suspend (Unplugged from USB)....70

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10.7.2 Power-On Standby....70
10.8 CLKRUN# PIN....70

11 CONFIGURATION REGISTERS....71

11.1 REGISTER DESCRIPTION....71
11.2 ACCESS DESIGNATORS....71
11.3 REGISTER SUMMARY....71
11.3.1 PCI Configuration Registers....72
11.3.2 Main Control Registers....73
11.3.3 USB Control Registers....73
11.3.4 PCI Control Registers....74
11.3.5 DMA Control Registers....74
11.3.6 Dedicated Endpoint Registers....74
11.3.7 Configurable Endpoint / FIFO Registers....75
11.3.8 Indexed Registers....76
11.4 PCI CONFIGURATION REGISTERS....77
11.4.1 (Address 00h; PCIVENDID) PCI Vendor ID....77
11.4.2 (Address 02h; PCIDEVID) PCI Device ID....77
11.4.3 (Address 04h; PCICMD) PCI Command....77
11.4.4 (Address 06h; PCISTAT) PCI Status....78
11.4.5 (Address 08h; PCIDEVREV) PCI Device Revision ID....78
11.4.6 (Address 09h; PCICLASS) PCI Class Code....78
11.4.7 (Address 0Ch; PCICACHESIZE) PCI Cache Line Size....79
11.4.8 (Address 0Dh; PCILATENCY) PCI Bus Latency Timer....79
11.4.9 (Address 0Eh; PCIHEADER) PCI Header Type....79
11.4.10 (Address 0Fh; PCIBIST) PCI Built-in Self Test....79
11.4.11 (Address 10h; PCIBASE0) PCI Base Address 0....79
11.4.12 (Address 14h; PCIBASE1) PCI Base Address 1....80
11.4.13 (Address 18h; PCIBASE2) PCI Base Address 2....80
11.4.14 (Address 1Ch, 20h, 24h; PCIBASE3, 4, 5) PCI Base Address 3-5....80
11.4.15 (Address 28h; CARDBUS); PCI CardBus CIS Pointer....80
11.4.16 (Address 2Ch; PCISUBVID); PCI Subsystem Vendor ID....81
11.4.17 (Address 2Eh; PCISUBID); PCI Subsystem ID....81
11.4.18 (Address 30h; PCIROMBASE) PCI Expansion ROM Base Address....81
11.4.19 (Address 34h; PCICAPPTR); PCI Capabilities Pointer....81
11.4.20 (Address 3Ch; PCIINTLINE); PCI Interrupt Line....81
11.4.21 (Address 3Dh; PCIINTPIN); PCI Interrupt Pin....81
11.4.22 (Address 3Eh; PCIMINGNT); PCI Minimum Grant....81
11.4.23 (Address 3Fh; PCIMINLAT); PCI Minimum Latency....81
11.4.24 (Address 40h; PWRMNGID); Power Management Capability ID....82
11.4.25 (Address 41h; PWRMNGNEXT); Power Management Next Pointer....82
11.4.26 (Address 42h; PWRMNGCAP); Power Management Capabilities....82
11.4.27 (Address 44h; PWRMNGCSR); Power Management Control/Status....83
11.4.28 (Address 46h; PWRMNGBRIDGE); Power Management Bridge Support....83
11.4.29 (Address 47h; PWRMNGDATA) Power Management Data....83
11.5 MAIN CONTROL REGISTERS....84
11.5.1 (Address 00h; DEVINIT); Device Initialization....84
11.5.2 (Address 04h; EECTL); EEPROM Control....85
11.5.3 (Address 08h; EECLKFREQ); EEPROM Clock Frequency....85
11.5.4 (Address 10h; PCIIRQENB0) PCI Interrupt Request Enable 0....86
11.5.5 (Address 14h; PCIIRQENB1) PCI Interrupt Request Enable 1....86
11.5.6 (Address 18h; CPUIRQENB0) CPU Interrupt Request Enable 0....87
11.5.7 (Address 1Ch; CPUIRQENB1) CPU Interrupt Request Enable 1....88

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.8 (Address 24h; USBIRQENB1) USB Interrupt Request Enable 1 89
11.5.9 (Address 28h; IRQSTAT0) Interrupt Request Status 0 90
11.5.10 (Address 2Ch; IRQSTAT1) Interrupt Request Status 1....90
11.5.11 (Address 30h; IDXADDR) Indexed Register Address 92
11.5.12 (Address 34h; IDXDATA) Indexed Register Data....92
11.5.13 (Address 38h; FIFOCTL) FIFO Control....92
11.5.14 (Address 40h; MEMADDR) FIFO Memory Address....93
11.5.15 (Address 44h; MEMDATA0) FIFO Memory Data 0 93
11.5.16 (Address 48h; MEMDATA1) FIFO Memory Data 1 93
11.5.17 (Address 50h; GPIOCTL) General Purpose I/O Control....94
11.5.18 (Address 54h; GPIOSTAT) General Purpose I/O Status....94
11.6 USB CONTROL REGISTERS....95
11.6.1 (Address 80h; STDRSP) Standard Response Control 95
11.6.2 (Address 84h; PRODVENDID) Product/Vendor ID 96
11.6.3 (Address 88h; RELNUM) Device Release Number 96
11.6.4 (Address 8Ch; USBCTL) USB Control....97
11.6.5 (Address 90h; USBSTAT) USB Status 98
11.6.6 (Address 94h; XCVRDIAG) Transceiver Diagnostic Control 98
11.6.7 (Address 98h; SETUP0123) Setup Bytes 0, 1, 2, 3....99
11.6.8 (Address 9Ch; SETUP4567) Setup Bytes 4, 5, 6, 7 99
11.6.9 (Address A4h; OURADDR) Our USB Address....100
11.6.10 (Address A8h; OURCONFIG) Our USB Configuration....100
11.7 PCI MASTER CONTROL REGISTERS....101
11.7.1 (Address 100h; PCIMSTCTL); PCI Master Control....101
11.7.2 (Address 104h; PCIMSTADDR); PCI Master Address....102
11.7.3 (Address 108h; PCIMSTDATA); PCI Master Data....102
11.7.4 (Address 10Ch; PCIMSTSTAT); PCI Master Status 102
11.8 DMA CONTROLLER REGISTERS 103
11.8.1 (DMACTL) DMA Control....103
11.8.2 (DMASTAT) DMA Status....104
11.8.3 (DMACOUNT) DMA Byte Count 104
11.8.4 (DMAADDR) DMA Address....104
11.8.5 (DMADESC) DMA Descriptor 104
11.9 DEDICATED ENDPOINT REGISTERS 105
11.9.1 (DEP_CFG) Dedicated Endpoint Configuration 105
11.9.2 (DEP_RSP) Dedicated Endpoint Response....105
11.10 CONFIGURABLE ENDPOINT / FIFO REGISTERS 106
11.10.1 (EP_CFG) Endpoint Configuration....106
11.10.2 (EP_RSP) Endpoint Response 107
11.10.3 (EP_IRQENB) Endpoint Interrupt Enable 108
11.10.4 (EP_STAT) Endpoint Status 109
11.10.5 (EP_AVAIL) Endpoint Available Count 110
11.10.6 (EP_DATA) Endpoint Data 110
11.11 INDEXED REGISTERS....111
11.11.1 (Index 00h; DIAG) Diagnostic Control....111
11.11.2 (Index 01h; PKTLEN) Packet Length....111
11.11.3 (Index 02h; FRAME) Frame Counter....111
11.11.4 (Index 03h; CHIPREV) Chip Revision 111
11.11.5 (Index 06h; HS_MAXPOWER) High-Speed Maximum Power....112
11.11.6 (Index 07h; FS_MAXPOWER) Full-Speed Maximum Power....112
11.11.7 (Index 08h; HS_INTPOLL_RATE) High-Speed Interrupt Polling Rate....112
11.11.8 (Index 09h; FS_INTPOLL_RATE) Full-Speed Interrupt Polling Rate....112
11.11.9 (Index 0Ah; HS_NAK_RATE) High-Speed NAK Rate 112

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11.10 (Index 0Bh; SCRATCH) Scratchpad 112
11.11.11 (Index 20h; EP_A_HS_MAXPKT) High-Speed Max Packet Size 113
11.11.12 (Index 21h; EP_A_FS_MAXPKT) Full-Speed Max Packet Size 113
11.11.13 (Index 30h; EP_B_HS_MAXPKT) High-Speed Max Packet Size 113
11.11.14 (Index 31h; EP_B_FS_MAXPKT) Full-Speed Max Packet Size 113
11.11.15 (Index 40h; EP_C_HS_MAXPKT) High-Speed Max Packet Size 114
11.11.16 (Index 41h; EP_C_FS_MAXPKT) Full-Speed Max Packet Size 114
11.11.17 (Index 50h; EP_D_HS_MAXPKT) High-Speed Max Packet Size 114
11.11.18 (Index 51h; EP_D_FS_MAXPKT) Full-Speed Max Packet Size 114
11.11.19 (Index 60h; EP_E_HS_MAXPKT) High-Speed Max Packet Size 115
11.11.20 (Index 61h; EP_E_FS_MAXPKT) Full-Speed Max Packet Size 115
11.11.21 (Index 70h; EP_F_HS_MAXPKT) High-Speed Max Packet Size 115
11.11.22 (Index 71h; EP_F_FS_MAXPKT) Full-Speed Max Packet Size 115
11.11.23 (Index 84h; STATIN_HS_INTPOLL_RATE) High-Speed Interrupt Polling Rate 116
11.11.24 (Index 85h; STATIN_FS_INTPOLL_RATE) Full-Speed Interrupt Polling Rate 116

12 USB STANDARD DEVICE REQUESTS 117

12.1 CONTROL READ TRANSFERS 118
12.1.1 Get Device Status 118
12.1.2 Get Interface Status 118
12.1.3 Get Endpoint Status 118
12.1.4 Get Device Descriptor (18 Bytes) 118
12.1.5 Get Device Qualifier (10 Bytes) 119
12.1.6 Get Other_Speed_Configuration Descriptor 119
12.1.7 Get Configuration Descriptor 120
12.1.7.1 Local-Enumerate 120
12.1.7.2 Auto-Enumerate 130
12.1.8 Get String Descriptor 0 136
12.1.9 Get String Descriptor 1 136
12.1.10 Get String Descriptor 2 136
12.1.11 Get String Descriptor 3 136
12.1.12 Get Configuration 136
12.1.13 Get Interface 136
12.2 CONTROL WRITE TRANSFERS 137
12.2.1 Set Address 137
12.2.2 Set Configuration 137
12.2.3 Set Interface 137
12.2.4 Device Clear Feature 137
12.2.5 Device Set Feature 138
12.2.6 Endpoint Clear Feature 138
12.2.7 Endpoint Set Feature 138

13 ELECTRICAL SPECIFICATIONS 139

13.1 ABSOLUTE MAXIMUM RATINGS 139
13.2 RECOMMENDED OPERATING CONDITIONS 139
13.3 DC SPECIFICATIONS 140
13.3.1 Core DC Specifications 140
13.3.1.1 Disconnected from USB 140
13.3.1.2 Connected to USB (High-Speed/Un-configured or Configured) 140
13.3.1.3 Active (High-Speed) 140
13.3.1.4 Connected to USB (Full-Speed/Un-configured or Configured) 140
13.3.1.5 Active (Full-Speed) 140
13.3.1.6 Suspended (Connected to USB) 140
13.3.1.7 Suspended (Disconnected from USB) 141

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.3.2 USB Full-Speed DC Specifications 141
13.3.3 USB High-Speed DC Specifications 142
13.3.4 PCI Bus DC Specification 143
13.4 AC SPECIFICATIONS 144
13.4.1 USB Full-Speed Port AC Specifications 144
13.4.2 USB High-Speed Port AC Specifications 144
13.4.3 USB Full-Speed Port AC Waveforms 145
13.4.4 USB Port AC/DC Specification Notes 147
13.4.5 PCI Bus AC Specifications 148
14 MECHANICAL DRAWING 149

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1 Introduction

1.1 Features

- USB Specification Version 2.0 Compliant (high and full speed)
- PCI Specification Version 2.3 Compliant
- PCI Bus Power Management Interface 1.1 Compliant
- Bridges between a PCI bus and a USB host
- Operates as PCI master or target
- Four channel scatter/gather DMA controller
- Integrated 8051 CPU with 8 Kbyte SRAM Program/Data memory
- Supports USB Full (12 Mb/sec) and High (480 Mb/sec) speeds
- USB Auto-Enumerate Modes
- Twelve Endpoints
  - Endpoint 0 for device control and status
  - Five Dedicated Endpoints for register and PCI accesses
  - Six Configurable Endpoints can be Isochronous, Bulk, or Interrupt, as well as In or Out
- Endpoints A, B, C, D share a 4K FIFO memory array
- Endpoints 0, E, F each have a 64 byte FIFO
- Automatic retry of failed packets
- Diagnostic register allows forced USB errors
- Software controlled disconnect allows re-enumeration
- Atomic operation to set and clear status bits simplifies software
- Serial EEPROM interface for initializing configuration registers and 8051 firmware
- CMOS Technology in 120 Pin Plastic TQFP Package
- 30 MHz oscillator with internal phase-lock loop multiplier
- 2.5V, 3.3V operating voltages, 5V tolerant PCI bus

1.2 Feature Overview

The NET2280 PCI/USB 2.0 high-speed peripheral controller allows Control, Isochronous, Bulk and Interrupt transfers between a PCI (Peripheral Component Interconnect) bus and a USB (Universal Serial Bus). The NET2280 supports the connection between a USB host computer and an intelligent peripheral such as a printer, scanner, communication device, or an embedded system that uses the PCI bus.

The seven main modules of the NET2280 are the USB Transceiver, Serial Interface Engine, USB Protocol Controller, endpoint FIFOs, PCI Bus Interface, 8051 CPU, and the Configuration Registers.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

USB Transceiver:
- Supports Full-Speed (12 MHz) or High-Speed (480 MHz) operation
- Serial data transmitter and receiver
- Parallel data interface to SIE
- Single parallel data clock output with on-chip PLL to generate higher speed serial data clocks
- Data and clock recovery from USB serial data stream
- SYNC/EOP generation and checking
- Bit-stuffing/unstuffing; bit stuff error detection
- Logic to facilitate Resume signaling
- Logic to facilitate Wake Up and Suspend detection
- Ability to switch between Full-Speed and High-Speed terminations/signaling

Serial Interface Engine (SIE):
- Interface between USB Protocol Controller and USB transceiver
- CRC generator and checker
- Packet Identifier (PID) decoder
- Forced error conditions
- USB 2.0 Test Modes

USB Protocol Controller
- Host to device communication
- Automatic retry of failed packets
- Automatic programmable response to Standard Requests during enumeration
- Up to 6 Isochronous, Bulk, or Interrupt endpoints, each with a data FIFO
- Five dedicated endpoints (2 IN, 2 OUT, 1 INTERRUPT) allow USB host access to configuration registers and to the PCI bus
- Configurable Control Endpoint 0
- Interface to FIFOs
- Software controlled disconnect allows device enumeration
- Software control of USB suspend and root port reset detection
- Software control of device remote wakeup
- Software control of root port wakeup

Endpoint FIFOs
- Endpoints A, B, C, D share 4Kbyte FIFO memory array
- Three FIFO configurations available for endpoints A, B, C, D
  - Endpoints A, B, C, D each have 1 Kbyte FIFO
  - Endpoints A, B each have 2 Kbyte FIFO (endpoints C and D not used)
  - Endpoint A has 2 Kbyte FIFO, endpoints B,C each have 1 Kbyte FIFO (endpoint D not used)
- Endpoints 0, E, F each have 64 byte FIFOs

PCI Bus Interface
- PCI Version 2.3 compliant 32-bit, 33 MHz PCI interface
- Master controller allows DMA, USB, or 8051 access to PCI target devices
- Target controller allows access to all chip resources via the PCI bus
- Four Channel DMA Controller allows USB packets to be transferred directly to/from the PCI bus
- PCI Interrupt generated for various USB events
- Arbiter supports 1 external and 9 internal bus masters
- Support of Power Management registers and PME# pin

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

8051 CPU

- 8K SRAM Unified Program/Data memory
- 256 byte internal RAM for CPU
- Access to PCI bus
- Access to Configuration registers
- Debug port
- Serial EEPROM interface for loading firmware into Program memory

Configuration Registers

- All registers (except Indexed) are directly accessible for fast access
- Registers are accessible from the PCI and USB buses, and from the 8051 CPU
- Any of the configuration registers can be initialized from the Serial EEPROM
- PCI Configuration Registers required for all PCI devices
- Main Control Registers for common functions
- USB Control Registers for the USB Protocol Controller Module
- PCI Control Registers for configuring PCI master controller
- DMA Control Registers for configuring the DMA controllers
- Endpoint Control Registers for each of the 12 endpoints
- Indexed Registers for infrequently accessed functions

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.3 Operation Overview

1.3.1 Applications

The NET2280 can be used in two different groups of PCI applications: Embedded Systems and Adapters. In a typical embedded system, a CPU with a PCI bus interface is used to configure and manage one or more PCI device controllers. The NET2280 provides this PCI system with USB 2.0 capability, allowing application data to be transferred between PCI device controllers and a USB host. The NET2280 is configured in PCI Adapter mode to support this configuration.

The Adapter group of applications allows a PCI device controller that was typically connected to a motherboard PCI slot to instead connect to a USB host through the NET2280. Since the NET2280 has an internal CPU that handles device initialization and the USB protocol, no external CPU is required. The NET2280 is configured in PCI Host mode to support this configuration.

1.3.2 Initialization

When the NET2280 is first reset, the EEPROM controller loads the 8051 program RAM with up to 8 Kbytes of firmware. It can also load any number of configuration registers. Loading 8K bytes from a 2MHz SPI EEPROM takes about 35 msec. The 8051 firmware contains the code to initialize all of the PCI and USB configuration registers, and the code necessary to support a USB enumeration. After the firmware has completed initialization, the NET2280 is enabled for enumeration. After enumeration, updated firmware can be downloaded to the NET2280. The 8051 can then re-program the configuration registers, disconnect from the host, and then re-enumerate.

1.3.3 PCI Interface

The PCI interface is enabled to respond to configuration or memory cycles after the EEPROM or 8051 has programmed the appropriate PCI configuration registers. In an embedded system with a PCI capable CPU, this external CPU acts as the Central Resource Function and configures all of the PCI devices, including the NET2280.

In a system with no external PCI capable CPU, the NET2280 provides the Central Resource Functions (arbitration, configuration transactions, IDSEL, etc.). The 8051 performs Type 0 or Type 1 configuration transactions to configure other devices on the PCI bus. The 8051 can also perform memory or I/O transactions to other PCI devices. These transactions will execute relatively slowly, because the 8051 needs to queue up 4 bytes of data and an address offset before initiating the transaction.

Once the PCI devices have been configured and initialized, data transfers can take place between the device PCI controller (scanner, printer, ADSL, etc.) and the NET2280. The NET2280 DMA controller, the device DMA controller, the NET2280 8051, or an external PCI capable CPU or DMA controller can perform these transfers.

1.3.4 USB Interface

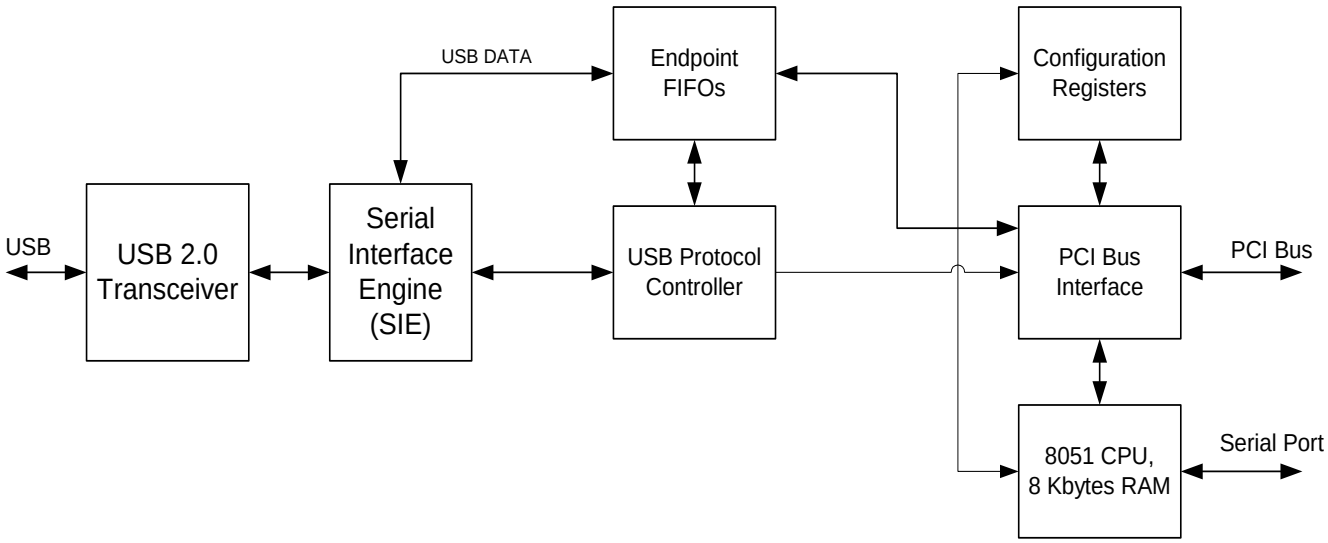
The NET2280 supports operation with either a full-speed (12 Mbits/sec) or high-speed (480 Mbits/sec) host. The NET2280 provides six configurable endpoints and five dedicated endpoints, in addition to endpoint 0. Each configurable endpoint has a FIFO associated with it. Data is written to or read from the FIFOs via the PCI bus. A USB host can read or write the 8051 program memory, initiate PCI transactions, and access the configuration registers using the dedicated endpoints.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.3.5 Interrupts

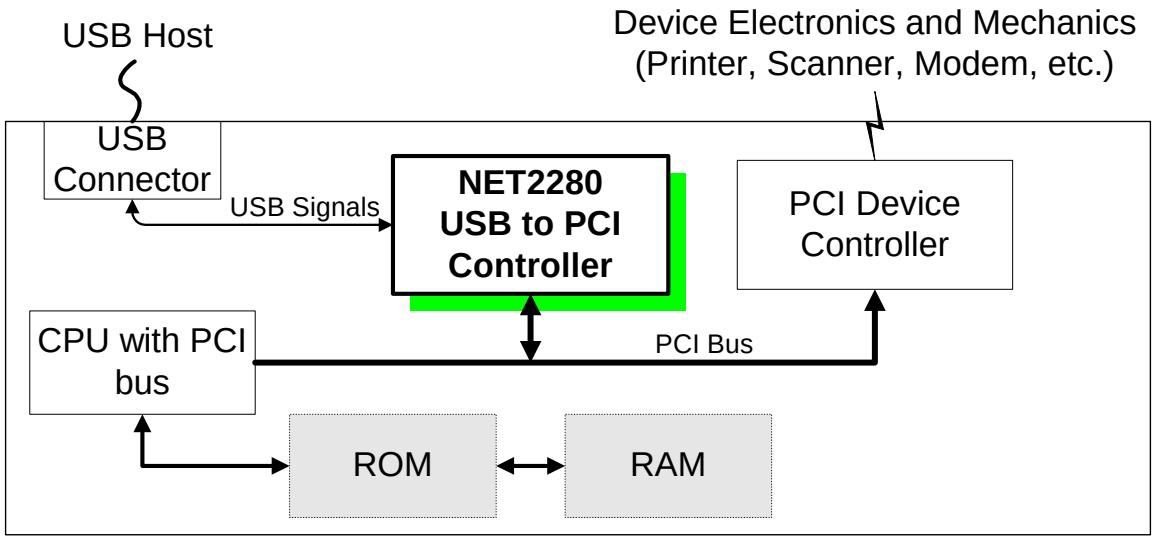
Interrupts from the various PCI and USB sources can be routed to either the 8051, the PCI bus, or the dedicated USB interrupt endpoint. The NET2280 either accepts (PCI Host mode) or generates (PCI Adapter mode) the PCI INTA# interrupt. In PCI Host mode, all internal and PCI interrupts should be routed to the either the 8051 or the USB interrupt endpoint. In PCI Adapter mode, interrupt sources can be routed to either the 8051, the PCI bus, or the dedicated USB interrupt endpoint.

1.4 NET2280 Block Diagram

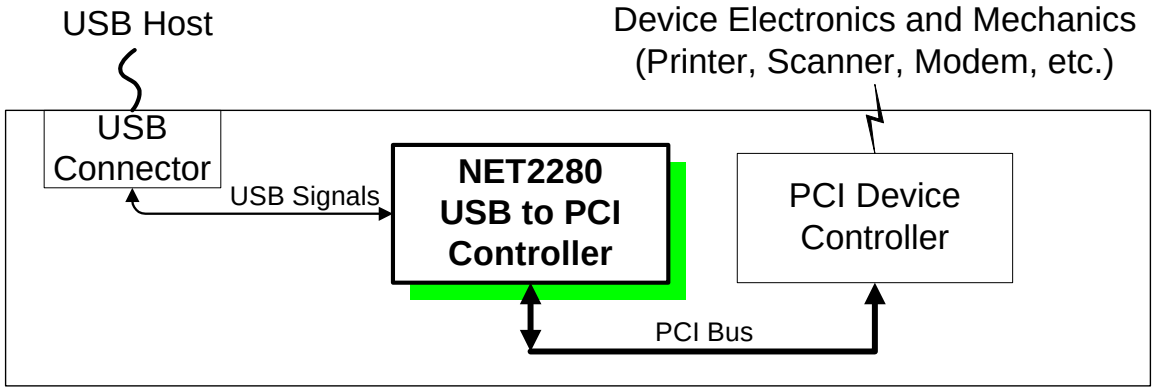


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.5 NET2280 Typical Adapter Mode Block Diagram

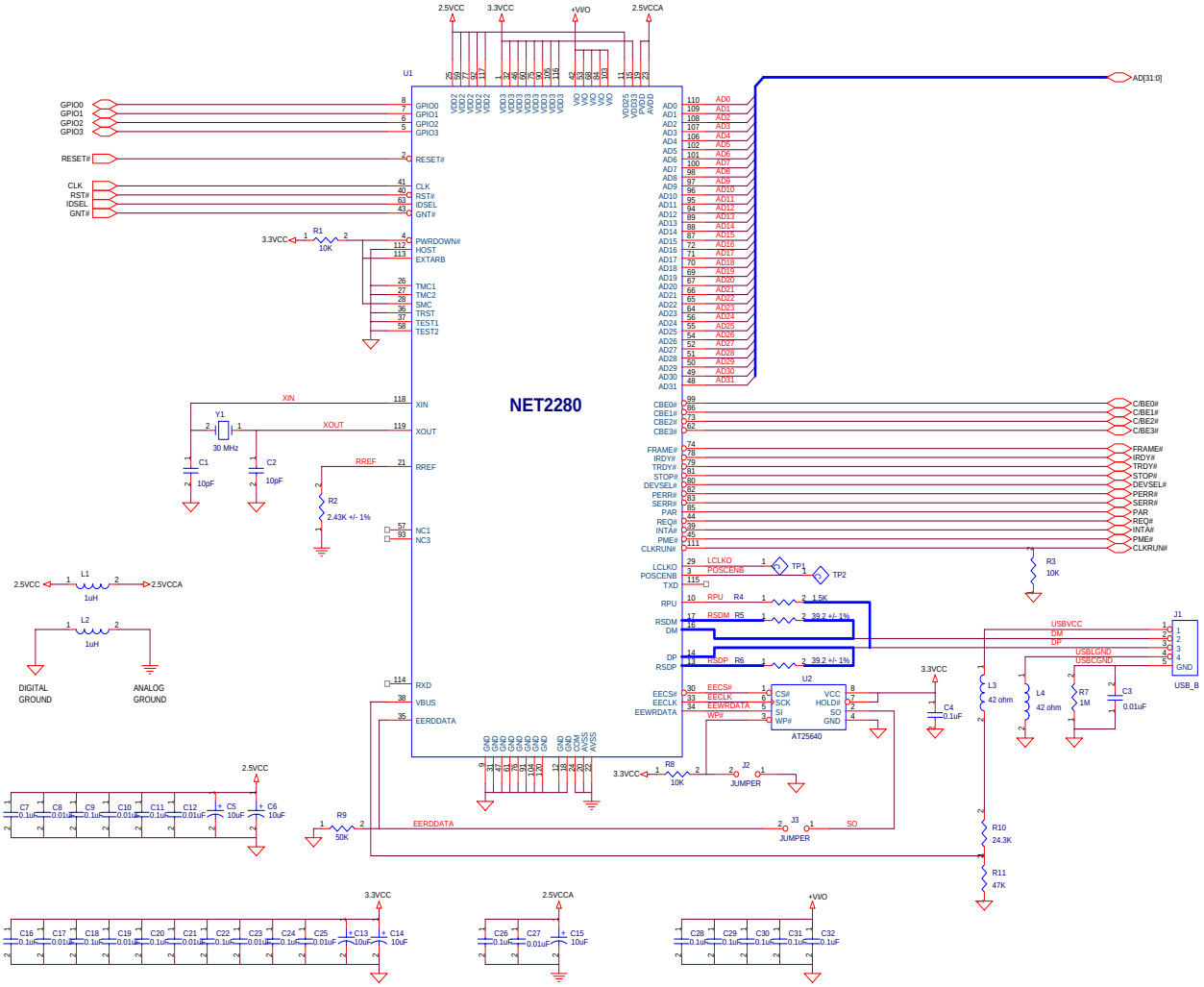


1.6 NET2280 Typical Host Mode Block Diagram



335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.7 Example connections to NET2280



335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.7.1 Example Part Numbers

Part	Manufacturer	Part Number	Website
30 MHz Fundamental, Series Resonant Crystal (Y1)	KDS	AT-49 30.000-16	http://www.kdsj.co.jp/english.html
Ferrite Beads (L3, L4)	Taiyo Yuden	FBMJ2125HS420-T	http://www.t-yuden.com/ferritebeads/index.cfm
1 µH Inductor, 10%, 0805 Pkg (L1, L2)	Taiyo Yuden	LK21251R0K	http://www.t-yuden.com/inductors/index.cfm
2.43K, 1% resistor, 0.1 Watt, 0603 Pkg (R2)	Panasonic	ERJ6ENF2431V	http://www.panasonic.com/industrial/components/pdf/002_er13_erj_2r_3r_6r_3e_6e_8e_14_12_dne.pdf
USB B Connector	Newnex	URB-1001	http://www.newnex.com

Note that the crystal should have a tolerance of +/- 0.005% (50 ppm) to guarantee a data rate of 480 Mbps +/- 500 ppm.

1.7.2 General PCB Layout Guidelines

USB2.0 high-speed 480 Mbits/sec data transfers utilize 400 mV differential signaling. This requires special Printed Circuit Board layout requirements.

The following guidelines must be followed to insure proper operation of the NET2280. It is strongly suggested that schematics and PCB layout be submitted to support@netchip.com for review prior to PCB fabrication.

1.7.2.1 USB Differential Signals

- Consult with board manufacturer for determining layer separation, trace width, and trace separation for maintaining differential impedance of 90 ohms and single ended impedance of 45 ohms.
- Maintain equal trace lengths for D+ and D-.
- Minimize number of vias and curves on D+ and D- traces.
- Use two 45 degree turns instead of one 90 degree turn.
- Minimize trace lengths shown in **bold** in the schematic in section 1.7.
- Prevent D+ and D- traces from crossing a power plane void. The same ground layer shall be kept next to the D+ and D- traces.
- Digital Ground (VSS) layer should be placed next to the layer where D+ and D- are routed.
- Avoid using studs or test points for observing USB signals.
- Maximize the distance of D+ and D- from other signals to prevent crosstalk.

1.7.2.2 Analog VDD (power)

- Analog power must be filtered from the digital power using the recommended circuit provided.
- Analog VDD and digital VDD must be connected via 1µH inductor.
- Analog VDD must be separated from digital VDD. If analog VDD and digital VDD are in the same layer, split the layer to accommodate the two power signals.
- AVDD and PVDD pins should be connected to analog VDD.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.7.2.3 Analog VSS (ground)

- Analog ground must be filtered from the digital ground using the recommended circuit provided.
- Analog VSS and digital VSS must be connected via a 1µH inductor.
- AVSS, PVSS, COM pins, and RREF's resistor should be connected to analog VSS.

1.7.2.4 Decoupling Capacitors

- At least one 0.1µF decoupling capacitor for every two pairs of digital/analog VDD and VSS should be located near the NET2280 device.
- At least one 0.01µF decoupling capacitor for every two pairs of digital/analog VDD and VSS should be located near the NET2280 device.
- Decoupling capacitors may be placed on the solder side of the PCB.
- At least one 10µF decoupling capacitor for every five 0.1µF decoupling capacitors.
- Use capacitors that have good quality at high frequency for low ESR, such as tantalum or ceramic capacitors. Do not use electrolytic capacitors.

1.7.2.5 EMI Noise Suppression

- A common-mode choke coil may suppress EMI noise effectively, although such a coil could affect USB 2.0 signal quality.
- Choose a good quality noise filter, if necessary.
- For a typical implementation, a choke is not required.
- Use good quality, shielded cables.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

1.8 Terminology

Byte. 8-bit quantity of data.

Word. 16-bit quantity of data.

Dword. 32-bit quantity of data.

Scalar. Multi-byte data element.

Big Endian. The most significant byte in a scalar is located at address 0.

Little Endian. The least significant byte in a scalar is located at address 0.

Clock cycle. One period of the PCI bus clock.

Agent. An entity that operates on the PCI bus.

Master (Initiator). Drives the address phase and transaction boundary (FRAME#). Initiates a transaction and drives data handshaking (IRDY#) with the target.

Target. Claims the transaction by asserting DEVSEL# and handshakes the transaction (TRDY#) with the initiator.

PCI Transaction. A read, write, read burst, or write burst operation on the PCI bus. It includes an address phase followed by one or more data phases.

PCI Transfer. During a *transfer*, data is moved from the source to the destination on the PCI bus. The assertion of TRDY# and IRDY# indicates a data *transfer*.

DAC. Dual address cycle. A PCI transaction where a 64-bit address is transferred across a 32-bit data path in two clock cycles.

Dedicated Endpoint. An endpoint that is used for a pre-defined task within the NET2280.

Configurable Endpoint. An endpoint that is available for general data transfers between the USB and PCI bus.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

2 Pin Description

Abbreviation	Description
I	Input
O	Output
I/O	Bi-Directional
PCI	PCI Compatible buffer
S	Schmitt Trigger
TS	Tri-State
STS	Sustained Tri-State, driven inactive one clock cycle before float
TP	Totem-Pole
OD	Open-Drain
PD	50K Pull-Down
PD2	100K Pull-Down
PU	50K Pull-Up
#	Active low

NOTE: Input pins that do not have an internal pull-up or pull-down resistor (designated by PU or PD in the 'Type' column above) must be driven externally when the NET2280 is in the suspended state.

2.1 Digital Power and Ground (26 pins)

Signal Name	Pins	Type	Description
VDD2	25, 59, 77, 92, 117	Power	**Core Supply Voltage.** Connect to the +2.5V supply.
VDD3	1, 32, 46, 60, 75, 90, 105, 116	Power	**I/O Supply Voltage.** Connect to the +3.3V supply.
GND	9, 31, 47, 61, 76, 91, 104, 120	GND	**Ground.** Connect to ground
VIO	42, 53, 68, 84, 103	Power	**PCI I/O Clamp Voltage.** For 3.3V PCI buses, connect this pin to 3.3V. For 5V PCI buses, connect this pin to 5V.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

2.2 USB Transceiver (15 pins)

Signal Name	Pins	Type	Description
DP	14	I/O	**High-Speed USB Data Port**. DP is the positive differential data signal of the high-speed USB data port. This pin connects directly to the USB connector.
DM	16	I/O	**High-Speed USB Data Port**. DM is the minus differential data signal of the high-speed USB data port. This pin connects directly to the USB connector.
RSDP	13	I/O	**Full-Speed USB Data Port**. RSDP is the positive differential data signal of the full-speed USB data port. This pin connects through a 39.2 ohm +/- 1% resistor to the USB connector.
RSDM	17	I/O	**Full-Speed USB Data Port**. RSDM is the minus differential data signal of the full-speed USB data port. This pin connects through a 39.2 ohm +/- 1% resistor to the USB connector.
RPU	10	O	**DP Pull Up Resistor.** Connect to DP pin through a 1.5K resistor.
RREF	21	I	**Reference Resistor.** Connect 2.43K +/- 1% resistor to ground.
VDD25	11	Power	**Supply Voltage.** Connect this pin to the +2.5V supply.
VDD33	15	Power	**Supply Voltage.** Connect this pin to the +3.3V supply.
PVDD	19	Power	**Supply Voltage.** Connect this pin to the +2.5V analog supply.
AVDD	23	Power	**Supply Voltage.** Connect this pin to the +2.5V analog supply.
GND	12, 18	GND	**Ground.** Connect these pins to ground.
AVSS	20, 22	GND	**Ground.** Connect these pins to analog ground.
COM (AVSS)	24	GND	**Ground.** Connect this pin to analog ground.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

2.3 Clocks, Reset, Misc (27 pins)

Signal Name	Pins	Type	Description
XIN	118	I	**Oscillator input.** Connect to 30 MHz crystal or external oscillator module.
XOUT	119	O	**Oscillator output.** Connect to crystal, or leave open if using an external oscillator module. The oscillator stops when the device is suspended.
LCLKO	29	O, 6mA, TP	**Local Clock Output.** This pin is a buffered clock output from the internal PLL, with the frequency depending on the state of the *Local Clock Frequency* field of the **DEVINIT** register. This pin oscillates for 500 µsec after the NET2280 is put into suspend mode. It is not driven while the device is suspended. When the internal oscillator is started, LCLKO is prevented from being driven for 2 msec.
RESET#	2	I, S	**Reset.** External reset. Connect to local or power-on reset. To reset when oscillator is stopped (initial power-up or in suspend state), assert for at least two ms. When oscillator is running, assert for at least five PCI clock periods. When PCI Adapter mode is selected, this pin will typically be connected to the RST# pin.
PWRDOWN#	4	I	**Power Down.** Driving a LOW level holds the NET2280 in a low-power mode by disabling the internal oscillator. The POSCENB pin is driven low when this pin is low.
POSCENB	3	O, 3mA, TP	**PCI Oscillator Enable.** This pin can be connected to the enable input of the 33 MHz PCI clock oscillator. When the NET2280 goes into suspend, this pin goes low, providing a means to stop the PCI clock to reduce power consumption. When the NAND tree test mode is selected, the test output is driven on this pin.
VBUS	38	I	**USB VBUS.** This input indicates when the NET2280 is connected to a powered-up USB host connector. An external voltage divider (24.3K to USB connector pin 1, 47K to ground) is required on this pin.
EECS#	30	O, 3mA, TP	**E^{2}PROM Chip Select.** Active low chip select.
EECLK	33	O, 3mA, TP	**E^{2}PROM Clock.** This pin provides the clock to the EEPROM. The frequency of this pin is determined by the **EECLKFREQ** register, and can vary from 1.875 MHz to 15 MHz.
EEWRDATA	34	O, 3mA, TP	**E^{2}PROM Write Data.** This pin is used to write data to the device.
EERDDATA	35	I	**E^{2}PROM Read Data.** This pin is used to read data from the device. A 50K pull-down resistor is required on this pin.
HOST	112	I	**PCI Host Mode.** When low, the device operates in PCI Adapter mode. When high, the device operates in PCI Host mode.
EXTARB	113	I	**External Arbiter Enable.** When low, the internal PCI arbiter services requests from an external PCI device. When high, the NET2280 requests the PCI bus from an external arbiter.
GPIO[3:0]	5, 6, 7, 8	I/O, 12mA	**General Purpose I/O.** Each of these bits can be programmed as either an input or output general-purpose pin. Interrupts can be generated on each of the pins that are programmed as inputs.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

			GPIO3 can be programmed as an active high output that is asserted during USB activity. This feature is selected using the *GPIO3 LED Select* field of the **GPIOCTL** register. GPIO3 must be programmed as an output for this feature to be available.
RXD	114	I	**Receive Data.** Debug port receive data.
TXD	115	O, 3mA, TP	**Transmit Data.** Debug port transmit data. When test modes are enabled, this pin is the test status output pin.
SMC	28	I	**Test input.** Scan path mode control. Connect to 3.3V for normal operation.
TMC1	26	I	**TMC1 Test input.** IDDQ test control input. Connect to ground for normal operation.
TMC2	27	I	**TMC2 Test input.** I/O buffer control. Connect to ground for normal operation.
TRST	36	I	**TRST Test input.** TAP controller reset. Connect to ground for normal operation.
TEST1, TEST2	37, 58	I	**Test inputs.** Connect to ground for normal operation TEST1 TEST2 Mode 0 0 Normal 0 1 Scan 1 0 Select "through clock" to bypass PLL. 1 1 NAND Tree
NC	57, 93	-	**No Connect.**

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

2.4 PCI Bus (52 pins)

Signal Name	Pins	Type	Description
CLK	41	I	**Clock.** All PCI signals, except RST# and interrupts, are sampled on the rising edge of this clock. The frequency can vary from 0 to 33 MHz. This clock needs to be oscillating during the EEPROM initialization sequence.
RST#	40	I/O, OD	**PCI Reset.** Reset is asserted and negated asynchronously to CLK, and is used to bring a PCI device to an initial state. All PCI signals are asynchronously tri-stated during reset. In PCI Host mode, this pin is driven when either the RESET# pin is asserted, or during a soft reset. In PCI Adapter mode, this is an input pin that resets the PCI logic.
AD[31:0]	48, 49, 50, 51, 52, 54, 55, 56, 64, 65, 66, 67, 69, 70, 71, 72, 87, 88, 89, 94, 95, 96, 97, 98, 100, 101, 102, 106, 107, 108, 109, 110	I/O, TS, PCI	**Address/Data Bus.** The PCI address and data are multiplexed onto the same bus. During the address phase, AD[31:0] contain the physical address of the transfer. During the data phase, AD[31:0] contain the data. AD31 is the most significant bit. Write data is stable when IRDY# is asserted, and read data is stable when TRDY# is asserted. Data is transferred when both IRDY# and TRDY# are asserted.
CBE[3:0]#	62, 73, 86, 99	I/O, TS, PCI	**Command/Byte Enable Bus.** The bus command and byte enables are multiplexed onto the same bus. During the address phase, CBE[3:0]# contain the bus command. During the data phase, CBE[3:0]# contain the byte enables. CBE0# corresponds to byte 0 (AD[7:0]), and CBE3# corresponds to byte 3 (AD[31:24]). **CBE[3:0]# Command** 0000 Interrupt Acknowledge 0001 Special Cycle 0010 I/O Read 0011 I/O Write 0100 Reserved 0101 Reserved 0110 Memory Read 0111 Memory Write 1000 Reserved 1001 Reserved 1010 Configuration Read 1011 Configuration Write 1100 Memory Read Multiple 1101 Dual Address Cycle 1110 Memory Read Line 1111 Memory Write and Invalidate
PAR	85	I/O, TS, PCI	**Parity.** Even parity is generated across AD[31:0], and C/BE[3:0]#. This means that the number of '1's on AD[31:0], C/BE[3:0]#, and

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

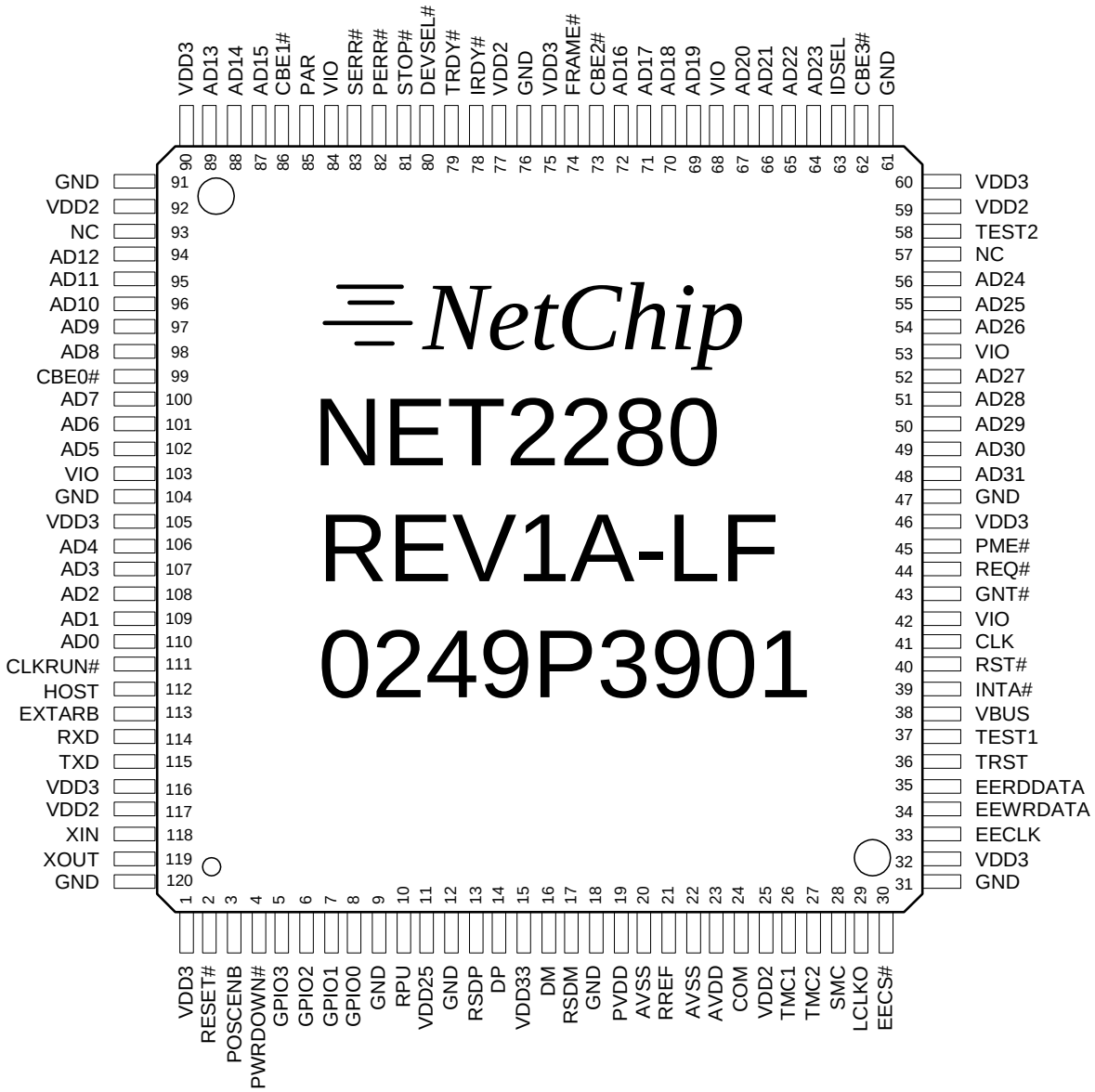
			PAR is an even number. PAR is valid one clock after the address phase. For data phases, PAR is valid one clock after IRDY# is asserted on write cycles, and one clock after TRDY# is asserted on read cycles. PAR has the same timing as AD[31:0], except delayed by one clock cycle. The bus initiator drives PAR for address and write data phases, and the target drives PAR for read data phases.
FRAME#	74	I/O, STS, PCI	**Frame.** This signal is driven by the initiator, and indicates the beginning and duration of an access. When FRAME# is first asserted, the address phase is indicated. When FRAME# is negated, the transaction is in the last data phase.
IRDY#	78	I/O, STS, PCI	**Initiator Ready.** This signal indicates that the initiator (bus master) is ready to transfer data. A data phase is completed when both IRDY# and TRDY# are asserted.
TRDY#	79	I/O, STS, PCI	**Target Ready.** This signal indicates that the target (bus slave) is ready to transfer data. A data phase is completed when both IRDY# and TRDY# are asserted.
STOP#	81	I/O, STS, PCI	**Stop.** This signal indicates that the target (bus slave) is requesting that the master stop the current transaction. Once STOP# is asserted, it must remain asserted until FRAME# is negated, whereupon STOP# must be negated. Also, DEVSEL# and TRDY# cannot be changed until the current data phase completes. STOP# must be negated in the clock following the completion of the last data phase, and must be tri-stated in the next clock. Data is transferred when IRDY# and TRDY# are asserted, independent of STOP#.
IDSEL	63	I	**Initialization Device Select.** This signal is used as a chip select during configuration read and write cycles. Each PCI slot or device typically has its IDSEL connected to a signal address line, allowing the host to select individual sets of configuration registers.
DEVSEL#	80	I/O, STS, PCI	**Device Select.** This signal indicates that the target (bus slave) has decoded its address during the current bus transaction. As an input, DEVSEL# indicates whether any device on the bus has been selected.
REQ#	44	I/O, TS	**Bus Request.** This signal indicates that an agent desires use of the bus. If the internal PCI arbiter is enabled, this pin is an input used to service an external bus request. If the internal PCI arbiter is disabled, this pin is an output used to request control of the bus.
GNT#	43	I/O, TS	**Bus Grant.** This signal indicates that the central arbiter has granted the bus to an agent. If the internal PCI arbiter is enabled, this pin is an output used to grant the bus to an external device. If the internal PCI arbiter is disabled, this pin is an input used to grant the bus to the NET2280.
PERR#	82	I/O, STS, PCI	**Parity Error.** This signal indicates that a data parity error has occurred. It is driven active by the receiving agent two clocks following the data that had bad parity.
SERR#	83	I/O, OD, PCI	**System Error.** This signal indicates that an address parity error, data parity error on the Special Cycle command, or other catastrophic error has occurred. It is driven active for one PCI clock period, and is synchronous to the CLK.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

INTA#	39	I/O, OD, PCI	**Interrupt A.** This signal is asserted to request an interrupt. Once asserted, it must remain asserted until the device driver clears it. INTA# is level sensitive and is asynchronous to the CLK. In PCI Host mode, this is an input pin used to service PCI interrupts. In PCI Adapter mode, this is an output pin used to request PCI interrupt service.
CLKRUN#	111	I/O, OD, STS, PCI	**Clock Run.** In PCI Adapter mode, this pin is used as an input to determine the status of the PCI CLK. It is used as an open-drain output to request starting or speeding up the PCI CLK. For systems that don't implement this signal, connect this pin to ground. In PCI Host mode, this pin is asserted continuously, indicating that the PCI clock is always enabled.
PME#	45	I/O, OD, PCI	**Power Management Event**. In PCI Adapter mode, this output pin is used to request a change in the device or system power state. An isolation circuit is typically required when using PME# in Adapter mode. In PCI Host mode, this input pin is used to monitor requests to change the power state. The polarity of this pin is controlled by the *PME Polarity* bit in the **USBCTL** register. When programmed for active low, the output is configured as an open-drain driver. When programmed for active high, the output is configured as a totem-pole driver. If this pin is not used in an application, it should be left open in PCI Adapter mode, or pulled-up in PCI Host mode.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

2.5 Physical Pin Assignment



335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

3 Reset and Initialization

3.1 Overview

The NET2280 normal initialization sequence consists of the following:

- Assert/De-Assert RESET# pin
- EEPROM is verified and loaded into configuration registers and 8051 Program RAM
- 8051 reset pin is de-asserted
- 8051 initializes other USB and PCI configuration registers

3.2 RESET# Pin

The RESET# pin causes all logic in the NET2280 to be set to its default state. It is typically connected to a power-on reset circuit. The EEPROM is loaded into configuration registers and the 8051 RAM when RESET# is de-asserted. Once the EEPROM has been read, the reset to the 8051 is de-asserted, allowing it to start executing instructions. If no valid EEPROM is detected, the 8051 is held in reset. If PCI host mode is selected, the USB interface is enabled for a default enumeration, and the PCI bus is not enabled. If PCI Adapter mode is selected, the PCI bus is enabled for a default enumeration, and the USB bus is not enabled.

3.3 PCI RST# Pin

If PCI Adapter mode is selected, the PCI RST# pin is an input, and causes all of the PCI interface logic and PCI configuration registers in the NET2280 to be set to their default state. It is typically generated by the PCI Central Resource Function reset logic. RST# and RESET# could be connected together to cause a full-chip reset when the PCI RST# is asserted. If RST# and RESET# are driven separately, EEPROM controller writes to PCI configuration registers can fail if RESET# is de-asserted before RST#.

If PCI Host mode is selected, the PCI RST# pin is an output. It is asserted when the RESET# pin is asserted, or the *PCI Soft Reset* bit is written. The RST# pin has an internal timer that keeps it asserted for between 2 and 3 milliseconds after RESET# is de-asserted, or after the *PCI Soft Reset* bit is written.

3.4 Root Port Reset

If the NET2280 detects a single-ended zero on the root port for greater than 2.5 microseconds, it is interpreted as a root port reset. The root port reset is only recognized if the VBUS input pin is high, and the *USB Detect Enable* bit in the **USBCTL** register is set. The following resources are reset:

- SIE
- USB state machines
- **OURADDR** and **OURCONFIG** Registers
- FIFO pointers

The root port reset does not affect the remainder of the configuration registers. The *Root Port Reset Interrupt* bit is set when a root port reset has been detected. The CPU (8051 or PCI) should take appropriate action when this interrupt occurs.

According to the USB Specification, the width of the USB reset is minimally 10ms and may be longer depending on the upstream host or hub. There is no specified maximum width for the USB reset.

3.5 Soft Resets

There are five soft reset bits in the configuration resisters that allow each section of the chip to be reset:

- Soft Reset PCI: resets the PCI module and PCI configuration registers
- Soft Reset CFG: resets all of the configuration registers
- Soft Reset USB: resets the SIE, USB, and **OURADDR** and **OURCONFIG** registers
- Soft Reset 8051: resets the 8051 CPU
- Soft Reset FIFO: resets the endpoint FIFOs

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

3.6 Reset Summary

The following table shows which device resources are reset when each of the 8 reset sources are asserted.

Device Resources / Reset Sources	8051	USB, SIE modules, **OURADDR, OURCONFIG** registers	PCI module, PCI Config Registers,	PCI RST# pin output (host mode)	Main, PCI Control, USB, Endpoint, Index Registers	All Configuration Registers	FIFOs
RESET# pin	X	X	X	X	X	X	X
PCI RST# pin (Adapter Mode)			X				
USB Root Port Reset		X					X
Soft Reset 8051	X						
Soft Reset USB		X					
Soft Reset CFG					X	X	
Soft Reset FIFO							X
Soft Reset PCI			X	X			

3.7 Initialization Summary

Various initialization sequences of the NET2280 are described below:

- No EEPROM, blank EEPROM, or invalid EEPROM
  - If the EERDDATA pin is always low, then an invalid EEPROM is detected. In this case, the default USB and PCI product ID ('h2280) is selected. A 50K pull-down resistor assures that EERDDATA is low if no EEPROM is installed.
  - If PCI Adapter mode, enable PCI interface (*PCI Enable* bit of **DEVINIT** register) using default register values. Don't automatically enable the USB interface, but wait for the PCI host to configure the USB registers for enumeration.
  - If PCI host mode, enable the USB interface using default register values. Don't automatically enable the PCI interface, but wait for the USB host to configure the PCI registers and perform the PCI enumeration.
  - Hold the 8051 in reset.
- Valid EEPROM with configuration register data only.
  - If in PCI Adapter mode, enable PCI interface (*PCI Enable* bit of **DEVINIT** register) using the register values loaded from the EEPROM. If in PCI host mode, wait for the USB host to enumerate the PCI bus.
  - Enable the USB interface using the register values loaded from the EEPROM. Optionally, if in PCI Adapter mode, wait for the PCI host to configure the USB registers.
  - Hold the 8051 in reset.
- Valid EEPROM with 8051 program memory only.
  - Start up the 8051 and let it configure and enable both the PCI and USB interfaces.
- Valid EEPROM with configuration register data and 8051 program memory.
  - If PCI Adapter mode, load PCI configuration registers from the EEPROM and enable the PCI interface (*PCI Enable* bit of **DEVINIT** register). This allows the NET2280 to respond quickly to the PCI enumeration process.
  - Enable the USB interface using the register values loaded from the EEPROM, or wait for the 8051 to configure the USB registers and handle the enumeration.
  - Start up the 8051 and let it configure the PCI and/or USB interfaces.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

4 EEPROM

4.1 Overview

The NET2280 provides an interface to SPI (Serial Peripheral Interface) compatible serial EEPROMs. This interface consists of a chip select, clock, write data, and read data pins, and operates at up to 15 MHz. Some 8Kx8 EEPROMs compatible with this interface are the Atmel AT25640, Catalyst CAT25C65, or ST Microelectronics M95640. The NET2280 will support up to a 16Mx8 EEPROM, utilizing 1, 2 or 3 byte addressing. The appropriate addressing mode is determined automatically by the NET2280.

4.2 EEPROM Data Format

The data in the EEPROM is stored in the following format:

Location	Value	Description
0	5A	Validation Signature
1	See below	EEPROM Format Byte
2	REG BYTE COUNT (LSB)	Configuration register byte count (LSB)
3	REG BYTE COUNT (MSB)	Configuration register byte count (MSB)
4	REGADDR (LSB)	1st Configuration Register Address (LSB)
5	REGADDR (MSB)	1st Configuration Register Address (MSB)
6	REGDATA (Byte 0)	1st Configuration Register Data (Byte 0)
7	REGDATA (Byte 1)	1st Configuration Register Data (Byte 1)
8	REGDATA (Byte 2)	1st Configuration Register Data (Byte 2)
9	REGDATA (Byte 3)	1st Configuration Register Data (Byte 3)
10	REGADDR (LSB)	2nd Configuration Register Address (LSB)
11	REGADDR (MSB)	2nd Configuration Register Address (MSB)
12	REGDATA (Byte 0)	2nd Configuration Register Data (Byte 0)
13	REGDATA (Byte 1)	2nd Configuration Register Data (Byte 1)
14	REGDATA (Byte 2)	2nd Configuration Register Data (Byte 2)
15	REGDATA (Byte 3)	2nd Configuration Register Data (Byte 3)
……		
REG BYTE COUNT + 4	MEM BYTE COUNT (LSB)	8051 Program memory byte count (LSB)
REG BYTE COUNT + 5	MEM BYTE COUNT (MSB)	8051 Program memory byte count (MSB)
REG BYTE COUNT + 6	PROG MEM (byte 0)	1st byte of 8051 program memory
REG BYTE COUNT + 7	PROG MEM (byte 1)	2nd byte of 8051 program memory
……		
FFFF	PROG MEM (byte n)	Last byte of 8051 program memory

The EEPROM Format Byte is organized as follows:

Bits	Description
7:3	**Reserved**
2	**8051 Start Enable.** When set, the 8051 reset is de-asserted after its program memory is loaded from the EEPROM. This bit is valid only when the *Program Memory Load* bit is set.
1	**Program Memory Load.** When set, the 8051 program memory is loaded from the EEPROM starting at location REG BYTE COUNT + 6. The number of bytes to load is determined by the value in EEPROM locations REG BYTE COUNT + 4 and REG BYTE COUNT + 5.
0	**Configuration Register Load.** When set, EEPROM locations 2 and 3 are read to determine how many 32-bit configuration registers are to be loaded.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

4.3 Initialization

After the RESET# pin is de-asserted, the first byte of the EEPROM is read to determine if the EEPROM has been programmed. If 'h5A is read, it is assumed that the EEPROM has been programmed for the NET2280. A pull-down resistor on the EERDDATA pin will produce a value of 0x00 if there is no EEPROM installed. If the first byte is not 'h5A, then the EEPROM is either not installed, is blank, or has been programmed with invalid data. In this case, the 8051 is held in reset, and either the USB or PCI interface is enabled for a default enumeration, depending on the PCI host mode pin.

If the EEPROM has valid data, then the second byte (EEPROM Format Byte) is read to determine which sections of the EEPROM should be loaded into the NET2280 configuration registers and memory. If bit 0 is set, then bytes 2 and 3 determine how many EEPROM locations contain configuration register addresses and data. Each configuration register entry consists of 2 bytes of register address (bit 10 low selects the PCI configuration registers, and bit 10 high selects the memory mapped configuration registers) and 4 bytes of register write data. For PCI Adapter mode, this feature allows the PCI interface to be available for host configuration transactions very quickly after RESET# is de-asserted. If bit 1 of the EEPROM Format Byte is set, locations REG BYTE COUNT + 4 and REG BYTE COUNT + 5 are read to determine how many bytes to transfer from the EEPROM into the 8051 program RAM. After this transfer has completed and if bit 2 in the EEPROM Format Byte is set, the 8051 reset is de-asserted, allowing it to start executing the firmware.

The EECLK pin frequency is determined by the *EE Clock Frequency* field of the **EECLKFREQ** register. The default speed is 1.875 MHz. For faster loading of large EEPROMs that support a faster clock, the first configuration register load from the EEPROM could be to the **EECLKFREQ** register.

Note 1: If operating in PCI Adapter mode, be sure to have the EEPROM set the *PCI Enable* bit in the **DEVINIT** register. Also, don't clear the *8051 Reset* pin in the **DEVINIT** register when loading this register from the EEPROM. The 8051 firmware will not have been loaded into the program memory yet.

Note 2: If operating in PCI Adapter mode, attention must be given to the relationship between the de-assertion of RST# and RESET#. If RESET# is de-asserted before RST#, then EEPROM writes to any PCI configuration register will fail. To avoid this situation, it is suggested that RESET# and RST# both be connected to the PCI reset when configured in PCI Adapter mode.

4.4 EEPROM Random Read/Write Access

A USB host, the 8051, or a master device on the PCI bus can use the **EECTL** register to access the EEPROM. This register contains 8-bit read and write data fields, read and write start signals, and related status bits. The following "C" routines demonstrate the firmware protocol required to access the EEPROM through the **EECTL** register. An interrupt can be generated whenever the *EEPROM Busy* bit of the **EECTL** register goes from true to false.

4.4.1 EEPROM Opcodes

READ_STATUS_EE_OPCODE = 5
WREN_EE_OPCODE = 6
WRITE_EE_OPCODE = 2
READ_EE_OPCODE = 3

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

4.4.2 EEPROM Low-Level Access Routines

```
int EE_WaitIdle()
{
   int eeCtl, ii;

   for (ii = 0; ii < 100; ii++)
   {
      Net2280Read(EECTL, eeCtl);                    /* read current value in EECTL */
      if ((eeCtl & (1 << EEPROM_BUSY)) == 0)        /* loop until idle */
         return(eeCtl);
   }
   PANIC("EEPROM Busy timeout!\n");
}

void EE_Off()
{
   EE_WaitIdle();                   /* make sure EEPROM is idle */
   Net2280Write(EECTL, 0);          /* turn off everything (especially EEPROM_CS_ENABLE) */
}

int EE_ReadByte()
{
   int eeCtl = EE_WaitIdle();                    /* make sure EEPROM is idle */

   eeCtl |= (1 << EEPROM_CS_ENABLE) |
         (1 << EEPROM_BYTE_READ_START);
   Net2280Write(EECTL, eeCtl);                   /* start reading */
   eeCtl = EE_WaitIdle();                        /* wait until read is done */
   return((eeCtl >> EEPROM_READ_DATA) & 0xff);   /* extract read data from EECTL */
}

void EE_WriteByte(int val)
{
   int eeCtl = EE_WaitIdle();                    /* make sure EEPROM is idle */

   eeCtl &= ~(0xff << EEPROM_WRITE_DATA);        /* clear current WRITE value */
   eeCtl |= (1 << EEPROM_CS_ENABLE) |
         (1 << EEPROM_BYTE_WRITE_START) |
         ((val & 0xff) << EEPROM_WRITE_DATA);
   Net2280Write(EECTL, eeCtl);
}
```


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

4.4.3 EEPROM Read Status Routine

```
...
EE_WriteByte(READ_STATUS_EE_OPCODE);        /* read status opcode */
status = EE_ReadByte();                     /* get EEPROM status */
EE_Off();                                   /* turn off EEPROM */
...
```

4.4.4 EEPROM Write Data Routine

```
...
EE_WriteByte(WREN_EE_OPCODE);               /* must first write-enable */
EE_Off();                                   /* turn off EEPROM */
EE_WriteByte(WRITE_EE_OPCODE);              /* opcode to write bytes */
#ifdef THREE_BYTE_ADDRESS_EEPROM            /* three-byte addressing EEPROM? */
   EE_WriteByte(addr >> 16);                /* send high byte of address */
#endif
EE_WriteByte(addr >> 8);                    /* send next byte of address */
EE_WriteByte(addr);                         /* send low byte of address */
for (ii = 0; ii < n; ii++)
{
   EE_WriteByte(buffer[ii]);                /* send data to be written */
}
EE_Off();                                   /* turn off EEPROM */
...
```

4.4.5 EEPROM Read Data Routine

```
...
EE_WriteByte(READ_EE_OPCODE);               /* opcode to write bytes */
#ifdef THREE_BYTE_ADDRESS_EEPROM            /* three-byte addressing EEPROM? */
   EE_WriteByte(addr >> 16);                /* send high byte of address */
#endif
EE_WriteByte(addr >> 8);                    /* send next byte of address */
EE_WriteByte(addr);                         /* send low byte of address */
for (ii = 0; ii < n; ii++)
{
   buffer[ii] = EE_ReadByte(buffer[ii]);    /* store read data in buffer */
}
EE_Off();                                   /* turn off EEPROM */
```


335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

5 8051 CPU

5.1 Overview

The embedded 8051CPU has 8 Kbytes of RAM that is shared between program and data. It also has 256 bytes of RAM for CPU registers, stack and scratch pad space. A debug port can be used for 8051 firmware development. The CPU executes machine cycles in 4 clock periods.

5.2 8051 Memory Map

5.2.1 Program Space (64 Kbytes)

The program memory consists of 8 Kbytes of RAM that can be written by the EEPROM, the USB interface, or the PCI interface.

Address	Device
0x0000-0x1FFF	Program RAM (8 Kbytes, shared with External Data Space)
0x2000-0xFFFF	Unused

5.2.2 External Data Space (64 Kbytes)

The data RAM is used by the 8051 for variable storage, and shares the 8K RAM with the Program Space. The MOVX instruction is used to access this memory.

Address	Device
0x0000-0x1FFF	Data RAM (8 Kbytes, shared with Program Space)
0x2000-0xFFFF	Unused

5.2.3 Internal Data Space (256 bytes)

The lower 128 bytes of internal RAM are accessed using either direct (MOV A, direct) or indirect (MOV A, @Ri) addressing. The upper 128 bytes of internal RAM are accessed using only indirect addressing. The Special Function Registers are accessed using only direct addressing. This allows them to occupy the same address range.

Address	Device
0x0000-0x007F	Internal RAM (128 bytes)
0x0080-0x00FF	Internal RAM (128 bytes)
0x0080-0x00FF	Special Function Registers (128 bytes)

5.2.3.1 Internal RAM (256 Bytes)

Address	Device
0x00-0x1F	CPU register set
0x20-0x2F	Bit addressable space
0x30-0x7F	Scratch pad and stack
0x80-0xFF	User definable

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

5.2.3.2 Special Function Registers

Address	Register	Description	2280 Unique	Bit Addressable
0x80	P0	Port 0		x
0x81	SP	Stack Pointer		
0x82	DPL	Data Pointer Low Byte		
0x83	DPH	Data Pointer High Byte		
0x84	DPL1	Data Pointer 1 Low Byte		
0x85	DPH1	Data Pointer 1 High Byte		
0x86	DPS	Data Pointer Select		
0x87	PCON	Power Control		
0x88	TCON	Timer/Counter Control		x
0x89	TMOD	Timer/Counter Mode Control		
0x8A	TL0	Timer/Counter 0 Low Byte		
0x8B	TL1	Timer/Counter 1 Low Byte		
0x8C	TH0	Timer/Counter 0 High Byte		
0x8D	TH1	Timer/Counter 1 High Byte		
0x8E	CKCON	Clock Control		
0x90	P1	Port 1		x
0x98	SCON	Debug Port Control		x
0x99	SBUF	Debug Port Data Buffer		
0xA0	P2	Port 2		x
0xA8	IE	Interrupt Enable Control		x
0xA9	SADDR	Debug port slave address		
0xB0	P3	Port 3		x
0xB8	IP	Interrupt Priority Control		x
0xB9	SADEN	Debug Port Automatic Address Recognition Enable		
0xC4	PMR	Power Management		
0xC7	TA	Timed Access		
0xC8	T2CON	Timer/Counter 2 Control		x
0xC9	T2MOD	Timer/Counter 2 Mode Control		
0xCA	RCAP2L	Timer/Counter 2 Capture (LSB)		
0xCB	RCAP2H	Timer/Counter 2 Capture (MSB)		
0xCC	TL2	Timer/Counter 2 Low Byte		
0xCD	TH2	Timer/Counter 2 High Byte		
0xD0	PSW	Program Status Word		x
0xE0	ACC	Accumulator		x
0xE8	IRQ0A	Interrupt 0 (LSB)	x	x
0xF0	B	B Register		x
0xF2	CFGCTL0	Cfg Register Control Byte 0	x	
0xF4	CFGADDR0	Cfg Register Address Byte 0	x	
0xF5	CFGADDR1	Cfg Register Address Byte 1	x	
0xF8	IRQ0B	Interrupt 0 (MSB)	x	x
0xF9	CFGDATA0	Cfg Register Data Byte 0	x	
0xFA	CFGDATA1	Cfg Register Data Byte 1	x	
0xFB	CFGDATA2	Cfg Register Data Byte 2	x	
0xFC	CFGDATA3	Cfg Register Data Byte 3	x	

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

5.2.3.2.1 Configuration Register Access Special Function Registers

A set of 7 SFRs allows the 8051 to access any configuration register in the NET2280. For a configuration register access, the 8051 first sets up the register address and data (for a write). Then the control byte is written with the *Start* bit set. The other fields in the control byte are formatted as follows:

- Read/Write : 0 = write, 1 = read
- Space Select: 0 = PCI configuration registers, 1 = Memory mapped configuration registers
- Byte Enables: select which of the four bytes are written.

The *Start* bit is automatically cleared when the transaction is done. The *Read/Write* bit determines the value returned in the *RegisterData* registers (0 = write data, 1 = read data returned from configuration read transaction).

Configuration or PCI access busy bits can also be monitored in the bit-addressable IRQ0B SFR.

Address	7	6	5	4	3	2	1	0
0xF2	Read/ Write	Start	Space Select		Byte Enables			
0xF3	Reserved							
0xF4	Register Address (LSB)							
0xF5	Register Address (MSB)							
0xF6	Reserved							
0xF7	Reserved							
0xF9	Register Data 0 (LSB)							
0xFA	Register Data 1							
0xFB	Register Data 2							
0xFC	Register Data 3 (MSB)							

5.2.3.2.2 Interrupt Status Special Function Registers

A set of 2 bit-addressable SFRs allows the 8051 to read any of the NET2280 **IRQSTAT0** interrupt status bits. Also, the busy status of a configuration register or PCI master access can be monitored. Normally, these transactions will be finished before the 8051 even tests these bits.

Address	7	6	5	4	3	2	1	0
0xE8 (IRQ0A)	Setup interrupt	EP F interrupt	EP E interrupt	EP D interrupt	EP C interrupt	EP B interrupt	EP A interrupt	EP 0 interrupt
0xF8 (IRQ0B)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	PCI Access Busy	Cfg Access Busy

5.2.4 PCI Master Cycles

The 8051 can initiate master cycles on the PCI bus. The 8051 does not have a wait input and only has an 8-bit data bus, so PCI accesses must be performed indirectly through a set of configuration registers. The **PCIMSTADDR** register determines the base PCI address of these accesses, and data is read or written through the **PCIMSTDATA** register. Bits in the **PCIMSTCTL** register are used to determine the direction of the transaction, start the transaction, and detect when the transaction is complete. The **PCIMSTADDR** register is used to determine the PCI address when performing type 0 or type 1 configuration cycles.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

These three registers are also used by the USB interface to access the PCI bus. The PCIOUT and PCIIN dedicated endpoints allow these registers to be accessed with one USB transaction. There isn't any configuration register resource locking during PCI master cycles, so the USB host and 8051 need to negotiate for control of the PCI bus.

5.3 8051 Interrupts

The 8051 can service interrupts from internal resources, USB sources, or PCI sources. Individual interrupts are enabled to the 8051 using the **CPUIRQENB0** and **CPUIRQENB1** registers. Interrupts from **IRQSTAT0** are routed to the 8051 interrupt input 0, and interrupts from **IRQSTAT1** are routed to the 8051 interrupt input 1.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

6 PCI Interface

6.1 Overview

The PCI interface allows the NET2280 to act as a PCI initiator (bus master), or PCI target (bus slave).

6.2 Configuration Transactions

Every PCI device has a 256-byte configuration space that can only be accessed with a Configuration Transaction. Type 0 configuration transactions are used to select a device on the bus where the transaction is being run. The address of a type 0 configuration transaction is shown below. Bits 1:0 select type 0, bits 7:2 select a Dword in the Configuration Space of the target, and bits 10:8 select one of eight possible functions on a multifunction device.

31 11	10 8	7 2	1	0
Reserved	Function Number	Register Number	0	0

Type 1 configuration transactions are used to select a device on another PCI bus. The address of a type 1 configuration transaction is shown below. Bits 15:11 select one of 32 devices on a given bus and bits 23:16 select one of 256 buses in a system.

31 24	23 16	15 11	10 8	7 2	1	0
Reserved	Bus Number	Device Number	Function Number	Register Number	0	1

When the NET2280 is configured in PCI Host mode, the internal 8051 or USB host will typically be used to configure other devices on the PCI bus. The 8051 needs to perform the following to run a configuration transaction:

- Write type 0 or type 1 configuration address (shown above) to the **PCIMSTADDR** register.
- Program *PCI Master Command Select* field of the **PCIMSTCTL** register to select configuration commands.
- For write transactions, write the data value into the **PCIMSTDATA** register.
- Start the transaction by setting the *PCI Master Start*, *PCI Master Read/Write*, and *PCI Master Byte Enables* fields of the **PCIMSTCTL** register.
- For read transactions, wait until the *PCI Master Start* bit is clear. Then read the data from the **PCIMSTDATA** register.

Note that the USB host can perform PCI master transactions with one OUT transaction to the PCIOUT dedicated endpoint. This OUT transaction causes all three PCI master registers to be written.

When the NET2280 is configured in PCI Adapter mode, another master on the PCI bus will program the PCI configuration registers using configuration transactions. Note that the *PCI Enable* bit of the **DEVINIT** register must be set before the NET2280 will respond to PCI transactions. If the *PCI Enable* bit is not set, the NET2280 will always terminate PCI transactions with a retry.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

6.3 Initiator Transactions

As an initiator, the internal 8051 or USB interface can perform single Dword memory, I/O, or configuration transactions to other PCI target devices. Since the NET2280 doesn't support bursting for these types of transactions, they won't be typically used to transfer performance sensitive data. The *PCI Master Command Select* field in the **PCIMSTCTL** register determines the type of PCI transaction that is performed (memory, I/O, configuration, etc). No FIFOs are involved in these types of cycles; data is transferred directly between the **PCIMSTDATA** register and the PCI bus.

The DMA controllers in the NET2280 can initiate PCI burst memory transactions. For an IN endpoint, if there is space available in the corresponding endpoint FIFO, the DMA controller will read data from a target on the PCI bus and store it in the FIFO. For an OUT endpoint, if there is data available in the corresponding endpoint FIFO, the DMA controller will read data from an endpoint FIFO and write it to a target on the PCI bus.

6.4 Target Transactions

As a PCI target device, the NET2280 can accept configuration or memory transactions. When the NET2280 is configured in PCI Adapter Mode, another PCI master will use configuration transactions to program the NET2280 PCI configuration registers. PCI memory transactions are used to access the other configuration registers, as well as the 8051 Program RAM. PCI Base Address Register 0 provides a 64 Kbyte range for accesses to the configuration registers. PCI Base Address Register 1 provides a 64 Kbyte range for accesses to the 8051 Program RAM. Since the 8051 Program RAM is only 8-bits wide, byte gathering logic allows the PCI bus to write or read 4 bytes during one transaction. PCI Base Address Register 2 provides an alternative means of accessing the endpoint A, B, C, or D FIFOs, and can be programmed for any size in increments of 64 Kbytes.

In some applications, PCI devices with integrated DMA controllers will transfer blocks of data to and from the NET2280 endpoint FIFOs. When the NET2280 detects an access to one of the endpoint FIFOs, it will accept or provide burst data to or from an endpoint FIFO. Some DMA controllers don't have an address hold feature, and expect the target to respond throughout a range of PCI addresses. The NET2280 allows an external PCI master to access endpoint FIFOs A, B, C, or D, using PCI Base Address 2. The *FIFO Base2 Select* field of the **FIFOCTL** register determines how the BASE2 address space is divided up between the endpoints. When this bit is low, each endpoint is mapped into one quarter of the entire BASE2 space. When this bit is high, writes to the first half are directed to endpoint A, and reads from the first half are from endpoint B. This allows a pair of endpoints to simulate a read/write memory. Endpoints C and D are similarly accessed using the top half of the BASE2 space.

If an IN FIFO fills up, the NET2280 PCI interface will assert the STOP# signal in response to memory write transactions until some space becomes available in the FIFO. If an OUT FIFO becomes empty, the NET2280 PCI interface will assert the STOP# signal in response to memory read transactions until some data becomes available in the FIFO. Alternatively, the *Ignore FIFO Availability* bit in the **FIFOCTL** register can be set. When this bit is high, writes to IN FIFOs and reads from OUT FIFOs proceed without regard to FIFO availability. Writes to a full FIFO result in the data being discarded and the *FIFO Overflow* bit in the **EP_STAT** register being set. Reads from an empty FIFO result in invalid data being driven onto the PCI bus, and the *FIFO Underflow* bit in the **EP_STAT** register being set. This mode should only be used if the PCI master knows the status of the FIFO before performing a data transfer.

Note: Care should be taken when operating the NET2280 in a PCI environment utilizing a PCI bridge. The NET2280 does not detect boundary crossings with respect to accesses to the BASE2 space. For example, if one PCI master is writing to the last two words of the endpoint A range, and another PCI master is writing to the first two words of the endpoint B range, a PCI bridge could perform burst combining. This would result in all four bytes being written to the endpoint A range. One possible solution is to make the BASE2 space twice as big as needed, thus preventing accesses near the endpoint boundaries.

Note: When an endpoint FIFO is read from the PCI bus, 4 bytes are transferred during each data transfer, independent of the PCI byte enables.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

6.5 Bus Arbitration

6.5.1 Overview

A PCI system using the NET2280 can either utilize an external bus arbiter, or use the NET2280 internal arbiter. This internal arbiter can only accept one bus request from another PCI device.

The NET2280 has 9 internal controllers that can request control of the PCI bus:

- 8051/USB Master Controller
- DMA Channels A-D
- DMA Scatter/Gather Controllers A-D

6.5.2 External Arbiter Mode

When the external PCI request input is disabled (EXTARB pin is high), the NET2280 generates a PCI request to an external arbiter. The PCI grant to the NET2280 is then routed to one of the nine internal requesters listed above. The 8051/USB Master Controller has priority, and will preempt any DMA operations in progress. After the 8051/USB Master Controller finishes its transaction, the preempted DMA controller is allowed to continue. The priority of the DMA controllers rotates after each DMA tenure on the bus.

6.5.3 Internal Arbiter Mode

If the external PCI request input is enabled (EXTARB pin is low), the arbiter accepts a PCI request from an external device. This external request is placed into the three-level round robin arbiter with the 8051/USB, four internal DMA channels, and four DMA scatter/gather controllers. Level 0 alternates between the 8051/USB and level 1, thus guaranteeing that the 8051/USB is granted up to 50% of the accesses. Level 1 consists of the four DMA controllers and the external requester. A sixth 'level 1' position is given to the four level 2 requestors which are the DMA scatter/gather controllers. For example, if all internal and external agents are requesting the bus, then the order of the agents accessing the bus would be:

8051/USB, DMA Channel A,
8051/USB, DMA Channel B,
8051/USB, DMA Channel C,
8051/USB, DMA Channel D,
8051/USB, External Requester,
8051/USB, DMA Scatter/Gather Channel A,
8051/USB, DMA Channel A,
8051/USB, DMA Channel B,
8051/USB, DMA Channel C,
8051/USB, DMA Channel D,
8051/USB, External Requester,
8051/USB, DMA Scatter/Gather Channel B,
and so forth.

6.5.4 Arbitration Parking

The PCI bus is not allowed to float for more than 8 clock cycles. When there are no requests for the bus, the arbiter must select a device to drive the bus to a known state by driving its GNT# pin active. If the external PCI request input is enabled (EXTARB pin is low), the NET2280 selects a PCI master to be parked on the bus during idle periods. The *PCI Arbiter Park Select* field of the **PCIMSTCTL** register determines which master is parked on the bus. When parked (GNT# driven during idle bus), the device drives AD[31:0], C/BE[3:0]#, and PAR to a known state.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7 USB Functional Description

7.1 USB Interface

The NET2280 is a USB function device, and as a result is always a slave to the USB host. The bit and packet level protocols, as well as the electrical interface of the NET2280, conform to USB Specification Version 2.0. The USB host initiates all USB data transfers to and from the NET2280 USB port. The NET2280 can be configured for up to 11 endpoints, in addition to Endpoint 0. Five of the endpoints are dedicated, and the other six can be an isochronous, bulk or interrupt type. The configuration registers are used to program the characteristics of each endpoint. The NET2280 operates in either Full speed (12 Mbps) or High speed (480 Mbps) modes.

7.2 USB Protocol

The packet protocol of the USB bus consists of tokens, packets, transactions, and transfers.

7.2.1 Tokens

Tokens are a type of Packet Identifier (PID), and follow the sync byte at the beginning of a token packet. The four classic types of tokens are OUT, IN, SOF, and SETUP. In high speed mode, the NET2280 also recognizes the PING token.

7.2.2 Packets

There are four types of packets: start-of-frame (SOF), token, data, and handshake. Each packet begins with a sync field and a Packet Identifier (PID). The other fields vary depending on the type of packet.

An SOF packet consists of the following fields:

- Sync byte (8-bits)
- Packet Identifier (8-bits)
- Frame Number (11-bits)
- CRC (5-bits)

A token packet consists of the following fields:

- Sync byte (8-bits)
- Packet Identifier (8-bits)
- Address (7-bits)
- Endpoint (4-bits)
- CRC (5-bits)

A data packet consists of the following fields: a token packet always precedes Data packets.

- Sync byte (8-bits)
- Packet Identifier (8-bits)
- Data (n bytes)
- CRC (16-bits)

A handshake packet consists of the following fields:

- Sync byte (8-bits)
- Packet Identifier (8-bits)

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.2.3 Transaction

A transaction consists of a token packet, optional data packet(s), and a handshake packet.

7.2.4 Transfer

A transfer consists of one or more transactions. Control transfers consist of a setup transaction, optional data transactions, and a handshake (status) transaction.

7.3 Automatic Retries

7.3.1 Out Transactions

If an error occurs during an OUT transaction, the NET2280 reloads its USB side FIFO write pointer back to the beginning of the failed packet. The host then sends another OUT token and re-transmits the packet. Once the packet has been successfully received by the NET2280, the Packet Received interrupt is set. The NET2280 can handle any number of back-to-back retries, but the host determines how many times a packet is retried.

7.3.2 In Transactions

If an error occurs during an IN transaction, the NET2280 reloads its USB side FIFO read pointer back to the beginning of the failed packet. The host then sends another IN token and the NET2280 re-transmits the packet. Once the host has successfully received the packet, the Packet Transmitted interrupt is set.

7.4 Ping Flow Control

When operating in high-speed mode, the NET2280 supports the PING protocol for OUT bulk and control endpoints. This protocol allows the NET2280 to indicate to the host that it can't accept an OUT packet. The host then sends PING tokens to query the NET2280. Once the NET2280 can accept a maximum size packet, it returns an ACK in response to the PING. Now the host sends an OUT token and data packet. The NET2280 returns an ACK handshake if the packet is accepted, and there is space to receive an additional packet. If it can accept the current packet, but no others, it returns a NYET handshake to the host. The host then starts sending PING tokens again.

7.5 Packet Sizes

The maximum packet size of an endpoint is determined by the corresponding **EP_MAXPKT** register. For IN transactions, the NET2280 will return a maximum size packet to the host if there are at least ‘max packet’ bytes in the FIFO. A packet of size less than the maximum is returned to the host in response to an IN token if the data in the FIFO has been explicitly validated.

The following table shows the allowable maximum packet sizes:

Type of Endpoint	Low Speed Mode*	Full-Speed Mode	High-Speed Mode
Control	8	8, 16, 32, 64	64
Bulk	N/A	8, 16, 32, 64	512
Interrupt	8	64 max	1024 max
Isochronous	N/A	1023 max	1024 max

* Low Speed Mode is not supported by the NET2280

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6 USB Endpoints

The NET2280 supports Control, Isochronous, Bulk, and Interrupt endpoints. All endpoints are unidirectional except for Control endpoints. Bi-directional bulk, isochronous, or interrupt traffic requires two endpoints.

7.6.1 Control Endpoint - Endpoint 0

The control endpoint, Endpoint 0, is a reserved endpoint. The host uses this endpoint to configure and gain information about the device, its configurations, interfaces and other endpoints. Control endpoints are bi-directional, and data delivery is guaranteed.

The host sends 8-byte setup packets to Endpoint 0 to which the device interprets and responds. The NET2280 has a set of registers dedicated to storing the setup packet, and uses the endpoint 0 FIFO for Control data. For Control writes, data flows through the FIFO from the USB bus to the PCI bus. For Control reads, data flows through the FIFO from the PCI bus to the USB bus.

When Endpoint 0 detects a setup packet, the NET2280 sets status bits and interrupts the CPU (8051 or PCI). The CPU reads the setup packet from NET2280 registers, and responds based on the contents. The CPU (8051 or PCI) provides any data returned to the host, including status and descriptors, unless the corresponding auto-enumerate bit is set. Refer to the Chapter 12, Standard Device Requests, for a description of the data that must be returned for each USB request. The host will reject descriptors that have unexpected values in any of the fields.

7.6.1.1 Control Write Transfer

A successful control write transfer to Control Endpoint 0 consists of the following:

Transaction	Stage	Packet Contents	# of bytes	Source
Setup	Setup Token	SETUP PID, address, endpoint, and CRC5	3	Host
	Data	DATA0 PID, 8 data bytes, and CRC16	11	Host
	Status	ACK	1	NET2280
Data (zero, one or more packets)	OUT Token	OUT PID, address, endpoint, and CRC5	3	Host
	Data (1/0)	DATA PID, N data bytes, and CRC16	N+3	Host
	Status	ACK	1	NET2280
Status	IN Token	IN PID, address, endpoint, and CRC5	3	Host
	Data	DATA1 PID, zero length packet, and CRC16	3	NET2280
	Status	ACK	1	Host

During the Setup transaction, the NET2280 stores the data stage packet in its setup registers. The NET2280 returns an ACK handshake to the host after all 8 bytes have been received. A *Setup Packet Interrupt* bit is set to notify the CPU (8051 or PCI) that a setup packet has been received. The 8-byte data packet is then read and interpreted by the CPU (8051 or PCI). A Setup transaction cannot be stalled or NAKed, but if the data is corrupted, then the NET2280 will not return an ACK to the host.

During the Data transaction, zero, one or more data packets are written into the Endpoint 0 FIFO. For each packet:

- Interrupt bits are set and can interrupt the CPU (8051 or PCI)
- The CPU (8051 or PCI) reads the FIFO
- The NET2280 returns an ACK if no error has occurred.

For a successful Status transaction, the NET2280 returns a zero length data packet. A NAK or STALL handshake can be returned if an error occurred.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.1.2 Control Write Transfer Details

For control write transfers, the host first sends 8 bytes of setup information. The setup bytes are stored into an 8-byte register bank that can be accessed by the CPU (8051 or PCI). After the eight bytes have been stored into the setup registers, the *Setup Packet Interrupt* bit is set.

The CPU (8051 or PCI) then reads the 8-byte setup packet and prepares to respond to the optional Data transactions. The number of bytes to be transferred in the Data transactions is specified in the setup packet. When the setup packet is received, the *Control Status Phase Handshake* bit is automatically set in anticipation of the control status phase. While this bit is set, the control status phase will be acknowledged with a NAK, allowing the CPU (8051 or PCI) to prepare its handshake response (ACK or STALL). Once the *Control Status Phase Handshake* bit is cleared and the OUT FIFO is empty, the ACK or STALL handshake will be returned to the host. Waiting for the OUT FIFO to become empty prevents another Control Write from corrupting the current packet data in the FIFO.

During a control write operation, optional Data transactions can follow the Setup transaction. The *Data Out Token Interrupt* bit is set at the beginning of each Data transaction. The bytes corresponding to the Data transaction are stored into the Endpoint 0 FIFO. If the FIFO fills up and another byte is transferred from the host, the NET2280 will return a NAK handshake to the host, signaling that the data could not be accepted.

If a packet is not successfully received (NAK or Timeout status), the *Data Packet Received Interrupt* bit will not be set, and the data will be automatically flushed from the FIFO. The host will re-send the same packet again. This process is transparent to the CPU (8051 or PCI).

If the CPU (8051 or PCI) has stalled this endpoint by setting the *Endpoint Halt* bit, the NET2280 will not store any data into the FIFO, and will respond with a STALL acknowledge to the host. There will not be a Status transaction in this case.

The CPU (8051 or PCI) can either poll the *Data Packet Received Interrupt* bit, or enable it as an interrupt, and then read the packet from the FIFO. If the host tries to write more data than was indicated in the setup packet, then the CPU (8051 or PCI) should set the *Endpoint Halt* bit for Endpoint 0. In this case there will not be a status stage from the host.

After all of the optional Data transaction packets have been received, the host will send an IN token, signifying the Status transaction. The *Control Status Interrupt* bit is set after the IN token of the Status transaction has been received. Until the *Control Status Phase Handshake* bit is cleared by the CPU (8051 or PCI) and the OUT FIFO is empty, the NET2280 will respond to the Status transaction with NAKs, indicating that the device is still processing the setup command. When the *Control Status Phase Handshake* bit has been cleared by the CPU (8051 or PCI) and the firmware has emptied the data from the OUT FIFO, the NET2280 will respond with a zero length data packet (transfer OK) or STALL (device had an error).

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.1.3 Control Read Transfer

A successful control read transfer from Control Endpoint 0 consists of the following:

Transaction	Stage	Packet Contents	# of bytes	Source
Setup	Setup Token	SETUP PID, address, endpoint, and CRC5	3	Host
	Data	DATA0 PID, 8 data bytes, and CRC16	11	Host
	Status	ACK	1	NET2280
Data (zero, one, or more packets)	IN Token	IN PID, address, endpoint, and CRC5	3	Host
	Data (1/0)	DATA PID, N data bytes, and CRC16	N+3	NET2280
	Status	ACK	1	Host
Status	OUT Token	OUT PID, address, endpoint, and CRC5	3	Host
	Data	DATA1 PID, zero length packet, and CRC5	3	Host
	Status	ACK	1	NET2280

The Setup transaction is processed in the same way as for control write transfers.

During the Data transaction, zero, one or more data packets are read from the Endpoint 0 FIFO. For each packet:

- Interrupt bits are set and can interrupt the CPU (8051 or PCI)
- The CPU (8051 or PCI) writes data to the FIFO
- If there is no data in the FIFO, a NAK or zero length packet is returned to the host
- The Host returns an ACK to the NET2280 if no error has occurred.

For a successful Status transaction, the Host sends a zero length data packet, and the NET2280 responds with an ACK. A NAK or STALL can be returned if an error occurred.

7.6.1.4 Control Read Transfer Details

For control read transfers, the host first sends 8 bytes of setup information. The setup bytes are stored into an 8-byte register bank that can be accessed from the CPU (8051 or PCI). After the eight bytes have been stored into the setup registers, the *Setup Packet Interrupt* bit is set.

The CPU (8051 or PCI) then reads the 8-byte setup packet and prepares to respond to the optional Data transactions. The number of bytes to be transferred in the Data transactions is specified in the setup packet. When the setup packet is received, the *Control Status Phase Handshake* bit is automatically set. While this bit is set, the control status phase will be acknowledged with a NAK, allowing the CPU (8051 or PCI) to prepare its handshake response (ACK or STALL). Once the *Control Status Phase Handshake* bit is cleared, the ACK or STALL handshake will be returned to the host.

During a control read operation, optional Data transactions can follow the Setup transaction. After the Setup transaction, the CPU (8051 or PCI) can start writing the first byte of packet data into the Endpoint 0 FIFO in anticipation of the Data transaction. The *Data In Token Interrupt* bit is set at the beginning of each Data transaction. If there is data in the Endpoint 0 FIFO, it is returned to the host. If Endpoint 0 has no data to return, it returns either a zero length packet (signaling that there is no more data available) or a NAK handshake (the data is not available yet).

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Packet Validated	Amount of Data in FIFO	Action
0	< Max Packet Size	NAK to host
X	>= Max Packet Size	Return data to host
1	empty	Zero length packet to host
1	>0	Return data to host

Note that the Max Packet Size validation can only be utilized for endpoints whose max Packet Size is a multiple of 4. For other endpoints, the packet must be explicitly validated using the *Endpoint Byte Count* field of the **EP_CFG** register.

After each packet has been sent to the host, the *Data Packet Transmitted Interrupt* bit is set.

If a packet is not successfully transmitted (*Timeout* status bit set), the *Data Packet Transmitted Interrupt* bit will not be set, and the same packet is sent to the host when another IN token is received. The retry operation is transparent to the CPU (8051 or PCI).

If the host tries to read more data than was requested in the setup packet, the CPU (8051 or PCI) should set the STALL bit for the endpoint.

After all of the optional Data transaction packets have been transmitted, the host will send an OUT token, followed by a zero length data packet, signifying the Status transaction. The *Control Status Interrupt* bit is set after the OUT token of the Status transaction has been received. Until the *Control Status Phase Handshake* bit is cleared by the CPU (8051 or PCI), the NET2280 will respond to the Status transaction with NAKs, indicating that the device is still processing the command specified by the Setup transaction. When the *Control Status Phase Handshake* bit has been cleared by the CPU (8051 or PCI), the NET2280 will respond with an ACK (transfer OK) or STALL (Endpoint 0 is stalled).

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.1.5 Auto-Enumerate

The Auto-Enumerate feature relieves the CPU (8051 or PCI) from servicing Standard Read or Write Requests from the host. Each type of Standard Read or Write Request has a register bit associated with it that determines how the request will be serviced. When the bit is low, the request is passed to the CPU (8051 or PCI) through the Setup registers. When the bit is high, the request is serviced without any support required by the CPU (8051 or PCI). The values returned to the host in Auto-Enumerate mode are determined by the values in other configuration registers. The following registers need to be initialized before using the Auto-Enumerate mode:

Register	Description
STDRSP	Standard Response
PRODVENDID	Product/Vendor ID
RELNUM	Release Number
USBCTL	USB Control
EP_CFG (for all endpoints)	Endpoint Configuration Registers
EP_RSP (for all endpoints)	Endpoint Response Registers
HS_MAXPOWER	High-speed maximum power
FS_MAXPOWER	Full-speed maximum power
HS_INTPOLL_RATE	High-speed interrupt polling rate
FS_INTPOLL_RATE	Full-speed interrupt polling rate
HS_NAK_RATE	High-speed OUT endpoint NAK rate
EP_(A-F)_HS_MAXPKT	Endpoint High-Speed Maximum Packet Size
EP_(A-F)_FS_MAXPKT	Endpoint Full-Speed Maximum Packet Size
STATIN_HS_INTPOLL_RATE	High-speed interrupt polling rate for STATIN
STATIN_FS_INTPOLL_RATE	Full-speed interrupt polling rate for STATIN

7.6.2 Isochronous Endpoints

Isochronous endpoints are used for the transfer of time critical data. Isochronous transfers do not support any handshaking or error checking protocol, and are guaranteed a certain amount of bandwidth during each frame. The Serial Interface Engine in the NET2280 ignores CRC and bit stuffing errors during isochronous transfers, but sets the handshaking bits in the **EP_STAT** register the same as for non-isochronous packets so that the CPU (8051 or PCI) can detect errors. Isochronous endpoints are unidirectional, with the direction defined by the endpoint configuration registers.

For an Isochronous OUT endpoint, the CPU (8051 or PCI) or DMA can read data from the FIFO after an entire packet has been received. If the FIFO is the same size as the maximum packet size, then the ISO bandwidth must be set such that the FIFO can be emptied before the next ISO packet arrives.

For an Isochronous IN endpoint, the CPU (8051 or PCI) or DMA can write data to the FIFO at the same time that data is being transmitted to the USB.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.2.1 Isochronous Out Transactions

Isochronous Out endpoints are used to transfer data from a USB host to the NET2280 PCI bus. An Isochronous OUT transaction consists of the following:

Stage	Packet Contents	# of bytes	Source
OUT Token	OUT PID, address, endpoint, and CRC5	3	Host
Data	DATA0 PID, N data bytes, and CRC16	N+3	Host

The USB host initiates an Isochronous OUT transaction by sending an OUT token to an Isochronous OUT endpoint. The *Data OUT Token Interrupt* bit is set when the OUT token is recognized. The bytes corresponding to the Data stage are stored into the endpoint's FIFO. Isochronous transactions are not retried, so if the FIFO is full when a packet is transferred from the host (or the *NAK OUT Packets* bit is set), the packet is discarded and the *FIFO Overflow* status bit is set. No handshake packets are returned to the host, but the *USB OUT ACK Sent*, and *Timeout* status bits are still set to indicate the status of the transaction. If a CRC error is detected, the packet is accepted, and the *Timeout* status bit is set. After every data packet is received, the CPU (8051 or PCI) should sample these status bits to determine if the NET2280 successfully received the packet.

By definition, isochronous endpoints do not utilize handshaking with the host. Since there is no way to return a stall handshake from an isochronous endpoint to the host, data that is sent to a stalled isochronous endpoint will be received normally. The Maximum Packet Size must be less than or equal to the FIFO size.

If small packets are sent to this endpoint, the PCI side must read a packet from the FIFO before the next one is sent from the host. For example, if two 1-byte packets are sent, each byte occupies one line in the FIFO. The **EP_AVAIL** register reports that 2 bytes are available, but there is no indication that two 32-bit reads are required to access them. Only one byte is valid for each 32-bit read, and the other bytes are ignored. The maximum packet size of an Isochronous endpoint must be a multiple of 4 if the DMA controller will be transferring more than one packet.

The CPU (8051 or PCI) or DMA wait for the *Data Packet Received Interrupt* bit to be set. Once the interrupt is set, the data can be read from the FIFO.

7.6.2.2 High Bandwidth Isochronous OUT Transactions

The host sends high-bandwidth OUT PID sequences for each microframe depending on the *Additional Transaction Opportunities* field in the Endpoint Descriptor as follows:

Additional Opportunities	PID Sequence
0	DATA0 (normal ISO)
1	MDATA, DATA1 (one extra transaction)
2	MDATA, MDATA, DATA2 (two extra transactions)

The NET2280 accepts data (unless the FIFO is full), and records the PID in the *OUT PID* field of **EP_STAT**. This allows firmware to track PIDs as they arrive and determine if the data sequence is complete.

OUT PID field	PID Received
00	DATA0
01	DATA1
10	DATA2
11	MDATA

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.2.3 Isochronous In Transactions

Isochronous In endpoints are used to transfer data from the NET2280 PCI bus to a USB host. An isochronous IN transaction consists of the following:

Stage	Packet Contents	# of bytes	Source
IN Token	IN PID, address, endpoint, and CRC5	3	Host
Data	DATA0 PID, N data bytes, and CRC16	N+3	NET2280

The USB host initiates an Isochronous IN transaction by sending an IN token to an Isochronous IN endpoint. The *Data IN Token Interrupt* bit is set when the IN token is recognized. If there is data in the endpoint's FIFO, it is returned to the host. If the endpoint has no data to return, a zero length packet is returned to the host. The NET2280 responds to the IN token according to the following table.

Packet Validated	Amount of Data in FIFO	Action
0	< Max Packet Size	Zero length packet to host; USB IN NAK Sent status bit set
X	>= Max Packet Size	Return data to host.
1	empty	Zero length packet to host
1	>0	Return data to host

Note that the Max Packet Size validation can only be utilized for endpoints whose max Packet Size is a multiple of 4. For other endpoints, the packet must be explicitly validated using the *Endpoint Byte Count* field of the **EP_CFG** register. For endpoints with an odd maximum packet size, all four bytes must be written to the FIFO at the end of the packet. Extra bytes in the last FIFO line are not transmitted to the host.

After the packet has been sent to the host, the *Data Packet Transmitted Interrupt* bit is set. If an IN token arrives and there is no valid packet in the FIFO, the NET2280 returns a zero-length packet, and the *FIFO Underflow* status bit is set. No handshake packets are returned to the host, but the *USB IN ACK Sent*, and *Timeout* status bits are still set to indicate the status of the transaction. After every data packet is transmitted, the CPU (8051 or PCI) should sample these status bits to determine if the packet was successfully transmitted to the host.

By definition, isochronous endpoints do not utilize handshaking with the host. Since there is no way to return a stall handshake from an isochronous endpoint to the host, data that is requested from a stalled isochronous endpoint will be transmitted normally.

7.6.2.4 High Bandwidth Isochronous IN Transactions

A USB device is required to send ISO PID sequences for each microframe according to the *Additional Transaction Opportunities* field in the Endpoint Descriptor and the **EP_n_HS_MAXPKT** register as follows:

Additional Opportunities	PID Sequence
0	DATA0 (normal ISO)
1	DATA1, DATA0 (one extra transaction)
2	DATA2, DATA1, DATA0 (two extra transactions)

When the first IN token of a microframe arrives, the NET2280 copies the *Additional Opportunities* field from the **EP_n_HS_MAXPKT** register to determine the initial PID. On each succeeding IN token of the microframe, the PID advances to the next token.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.3 Bulk Endpoints

Bulk endpoints are used for guaranteed error-free delivery of large amounts of data between a host and device. Bulk endpoints are unidirectional, with the direction defined by the endpoint configuration registers.

7.6.3.1 Bulk Out Transactions

Bulk Out endpoints are used to transfer data from a USB host to the NET2280 PCI bus. A bulk OUT transaction to a Bulk Out endpoint consists of the following:

Stage	Packet Contents	# of bytes	Source
OUT Token	OUT PID, address, endpoint, and CRC5	3	Host
Data (1/0)	DATA PID, N data bytes, and CRC16	N+3	Host
Status	ACK, NAK, or STALL	1	NET2280

The USB host initiates a Bulk OUT transaction by sending an OUT token to a Bulk OUT endpoint. The *Data OUT Token Interrupt* bit is set when the OUT token is recognized. The bytes corresponding to the Data stage are stored into the endpoint's FIFO. If the FIFO is full when another byte is transferred from the host, the byte will be discarded and the *USB OUT NAK Sent* status bit will be set. At the completion of the packet, a NAK handshake will be returned to the host, indicating that the packet could not be accepted.

All USB data passes through the endpoint's FIFO to the PCI bus. The CPU waits until the *Data Packet Received Interrupt* occurs before reading the data from the FIFO.

If a packet is not successfully received (*USB OUT NAK Sent* or *Timeout* status bits set), the *Data Packet Received Interrupt* bit will not be set, and the data will be automatically flushed from the FIFO. The host will re-send the same packet again. This process is transparent to the CPU (8051 or PCI).

If the CPU (8051 or PCI) has stalled this endpoint by setting the *Endpoint Halt* bit, the NET2280 will not store any data into the FIFO, and will respond with a STALL handshake to the host.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.3.2 Bulk In Endpoints

Bulk IN Endpoints are used to transfer data from the NET2280 PCI bus to a USB host. A bulk read transaction from a Bulk IN Endpoint consists of the following:

Stage	Packet Contents	# of bytes	Source
IN Token	IN PID, address, endpoint, and CRC5	3	Host
Data (1/0)	DATA PID, N data bytes, and CRC16, or NAK or STALL	N+3	NET2280
Status	ACK	1	Host

The USB host initiates a Bulk IN transaction by sending an IN token to a Bulk IN endpoint. The *Data IN Token Interrupt* bit is set when the IN token is recognized. If there is validated data in the endpoint's FIFO, it is returned to the host. If the endpoint has no data to return, it returns either a zero length packet (signaling that there is no more data available) or a NAK handshake (the data is not available yet).

Packet Validated	Amount of Data in FIFO	Action
0	< Max Packet Size	NAK to host
X	>= Max Packet Size	Return data to host
1	empty	Zero length packet to host
1	>0	Return data to host

Note that the Max Packet Size validation can only be utilized for endpoints whose max Packet Size is a multiple of 4. For other endpoints, the packet must be explicitly validated using the *Endpoint Byte Count* field of the **EP_CFG** register.

After the packet has been sent to the host, the *Data Packet Transmitted Interrupt* bit is set.

If a packet is not successfully transmitted (*Timeout* status bit set), the *Data Packet Transmitted Interrupt* bit will not be set, and the same packet is sent to the host when another IN token is received. The retry operation is transparent to the CPU (8051 or PCI).

If the CPU (8051 or PCI) has stalled this endpoint by setting the *Endpoint Halt* bit, the NET2280 will respond to the IN token with a STALL handshake to the host.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.4 Interrupt Endpoints

Interrupt endpoints are used for sending or receiving small amounts of data to the host with a bounded service period.

7.6.4.1 Interrupt Out Transactions

Interrupt Out endpoints are used to transfer data from a USB host to the NET2280 PCI bus. An interrupt OUT transaction to an Interrupt OUT endpoint consists of the following:

Stage	Packet Contents	# of bytes	Source
OUT Token	OUT PID, address, endpoint, and CRC5	3	Host
Data (1/0)	DATA PID, N data bytes, and CRC16	N+3	Host
Status	ACK, NAK, or STALL	1	NET2280

The behavior of an Interrupt OUT endpoint is the same as a Bulk OUT endpoint, except for the toggle bit. If the *Interrupt Mode* bit in the **EP_RSP** register is cleared, the toggle bit of the Interrupt OUT endpoint is initialized to 0 (DATA0 PID), and behaves the same as a Bulk OUT endpoint. If the *Interrupt Mode* bit is set, the toggle bit of the Interrupt OUT endpoint changes after each data packet is received from the host, without regard to the Status stage. Also, the PING protocol is not supported for Interrupt OUT endpoints.

7.6.4.2 Interrupt In Endpoints

An Interrupt IN endpoint is polled at a rate which is specified in the endpoint descriptor. An interrupt transaction from an Interrupt IN endpoint consists of the following:

Stage	Packet Contents	# of bytes	Source
IN Token	IN PID, address, endpoint, and CRC5	3	Host
Data (1/0)	DATA PID, N data bytes, and CRC16	N+3	NET2280
Status	ACK	1	Host

The behavior of an Interrupt IN endpoint is the same as a Bulk IN endpoint, except for the toggle bit. If the *Interrupt Mode* bit is cleared, the toggle bit of the Interrupt IN endpoint is initialized to 0 (DATA0 PID), and behaves the same as a Bulk IN endpoint. An interrupt endpoint may be used to communicate rate feedback information for certain types of isochronous functions. To support this mode, the *Interrupt Mode* bit is set, and the toggle bit of the Interrupt IN endpoint changes after each data packet is sent to the host, without regard to the Status stage.

If the max packet size is not a multiple of 4, the packet must be explicitly validated using the *Endpoint Byte Count* field of the **EP_CFG** register. Once the packet has been validated and send to the host, the NET2280 will **not** return a zero-length packet in response to the next IN token.

7.6.4.3 High Bandwidth INTERRUPT Endpoints

From the USB device point of view, high-bandwidth INTERRUPT endpoints are the same as BULK endpoints, except that the MAXPKT can be any value from 1 to 1024. Normal INTERRUPT endpoints in full-speed mode can set MAXPKT from 1 to 64.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.5 Dedicated Endpoints

The NET2280 has five dedicated endpoints:

- CFGOUT – used to perform configuration register writes, and to select the address for configuration register reads
- CFGIN – used to perform configuration register reads
- PCIOUT – used to perform PCI master writes, and to initiate PCI master reads
- PCIIN – used to retrieve PCI master read data
- STATIN – used to report interrupt status to the USB host

The endpoint address of each of these endpoints is determined by the **DEP_CFG** registers. *No hardware locking is provided between 8051 and USB accesses to the PCI bus or configuration registers, so firmware must provide this function if these two devices could be making simultaneous accesses.*

7.6.5.1 CFGOUT Endpoint

This BULK OUT endpoint is used to perform configuration register writes, and to select the address for configuration register reads. The **DEP_CFG** configuration register determines its endpoint number. The maximum packet size is 16 bytes in full-speed mode and 512 bytes in high-speed mode. Packets sent to this endpoint should always have up to 10 bytes formatted as follows:

Byte Index	7	6	5	4	3	2	1	0
0	Reserved	Reserved	Space Select		Byte Enables			
1	Reserved							
2	Register/Memory Address (LSB)							
3	Register/Memory Address (MSB)							
4	Reserved							
5	Reserved							
6	Register Write Data 0 (LSB)							
7	Register Write Data 1							
8	Register Write Data 2							
9	Register Write Data 3 (MSB)							

An OUT packet that is ACKed and is at least 10 bytes long causes a configuration register write. An OUT packet that is ACKed and that has 6 bytes validates the address and control for a configuration register read. All OUT bytes after the tenth byte are ignored, and subsequent OUT packets to this endpoint are NAKed until the configuration write has completed. Packets with less than 6 bytes result in a STALL handshake and the endpoint being halted.

An OUT that results in a timeout invalidates the address for reading and writing. Trying to read (IN) from an invalid address results in the endpoint NAKing until a valid address is provided.

The 16-bit Register/Memory Address field selects one of the NET2280 configuration registers, or a location in the 8051 Program RAM. Since the registers are Dword aligned, the least significant two bits of the address are not used for register accesses. The Space Select field determines whether the PCI Configuration registers (00h-3Fh), the normal configuration registers (000h-3FFh), or the 8051 RAM are accessed. For writes to the 8051 RAM, only data byte 0 is valid (single byte writes only). The upper 3 bytes are ignored. The byte enables determine which of the four bytes in the selected configuration register are accessed (byte enable 0 corresponds to bits 7:0, and so on).

Space Select	Resource
0	PCI Configuration Registers
1	Memory Mapped Configuration Registers
2	8051 Program RAM
3	Reserved

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

In USB high-speed mode, the delay from the beginning of the OUT token until the configuration register write transaction finishes and the OUT packet is ACKed is about 900 ns. In USB full-speed mode, the delay is about 14.5 µs. Taking into account the host latencies on a typical PC running Windows, the average time to perform a configuration register write is about 250 µs in high-speed mode and 8.5 ms in full-speed mode.

7.6.5.2 CFGIN Endpoint

This BULK IN endpoint is used to perform configuration register reads. The **DEP_CFG** configuration register determines its endpoint number. The maximum packet size is 8 bytes in full-speed mode and 512 bytes in high-speed mode. Packets read from this endpoint always have 4 bytes formatted as follows:

Byte Index	7	6	5	4	3	2	1	0
0	Register Read Data 0 (LSB)							
1	Register Read Data 1							
2	Register Read Data 2							
3	Register Read Data 3 (MSB)							

For reads from the 8051 RAM, only data byte 0 is valid (single byte reads only). The upper 3 bytes are undefined.

An IN token to this endpoint causes a configuration register read to start. If the read completes before the USB data packet must be provided, the endpoint will ACK (this is likely on the CFG endpoint in a system with a 33 MHz PCI bus and little contention for the registers). Otherwise, the endpoint will NAK until the read data becomes valid. If a new OUT (ACKed or otherwise) occurs after the read data becomes valid but before the host performs the IN to collect the data, the read data is invalidated. An IN that returns valid data invalidates the current read data so that the next IN causes a new read.

In USB high-speed mode, the delay from the beginning of the OUT token (for the OUT packet that sets the configuration read address) until the Data IN packet (containing the configuration read data) is ACKed is about 2.5 µs. In USB full-speed mode, the delay is about 28 µs. Taking into account the host latencies on a typical PC running Windows, the average time to perform a configuration register read in high-speed mode is about 500 µs (250 µs to set the address and 250 µs to read the data). In full-speed mode it takes about 17 ms (8.5 ms to set the address and 8.5 ms to read the data).

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.5.3 PCIOUT Endpoint

This BULK OUT endpoint is used to initiate PCI master writes and reads. The **DEP_CFG** configuration register determines its endpoint number. The maximum packet size is 16 bytes in full-speed mode and 512 bytes in high-speed mode. Packets sent to this endpoint should always have up to 10 bytes formatted as follows:

Byte Index	7	6	5	4	3	2	1	0
0	**PCIMSTCTL** register bits 7:0 (*PCI Master Start* and *PCI Master Read/Write* bits not used)							
1	**PCIMSTCTL** register bits 15:8 (determines PCI bus operation during master transaction)							
2	**PCIMSTADDR** register bits 7:0							
3	**PCIMSTADDR** register bits 15:8							
4	**PCIMSTADDR** register bits 23:16							
5	**PCIMSTADDR** register bits 31:24							
6	**PCIMSTDATA** register bits 7:0							
7	**PCIMSTDATA** register bits 15:8							
8	**PCIMSTDATA** register bits 23:16							
9	**PCIMSTDATA** register bits 31:24							

An OUT packet that is ACKed and is at least 10 bytes long initiates a PCI master write transaction. An OUT packet that is ACKed and has 6 bytes initiates a PCI master read transaction. All OUT bytes after the tenth byte are ignored, and subsequent OUT packets to this endpoint are NAKed until the PCI write transaction has completed. Packets with less than 6 bytes result in a STALL handshake and the endpoint being halted.

An OUT that results in a timeout invalidates the address for reading and writing. Trying to read (IN) from an invalid address results in the endpoint NAKing until a valid address is provided.

In USB high-speed mode, the delay from the beginning of the OUT token until the PCI write transaction finishes (assuming no PCI target wait states) is about 1.5 µs. In USB full-speed mode, the delay is about 16.5 µs. Taking into account the host latencies on a typical PC running Windows, the average time to perform a PCI memory write is about 250 µs in high-speed mode and 8.5 ms in full-speed mode.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.6.5.4 PCIIN Endpoint

This BULK IN endpoint is used to perform PCI read transactions. The **EP_CFG** configuration register determines its endpoint number. The maximum packet size is 8 bytes in full-speed mode and 512 bytes in high-speed mode. Packets read from this endpoint always have 4 bytes formatted as follows:

Byte Index	7	6	5	4	3	2	1	0
0	PCI Read Data 0 (LSB)							
1	PCI Read Data 1							
2	PCI Read Data 2							
3	PCI Read Data 3 (MSB)							

An IN token to this endpoint causes a PCI read data to be returned. The PCI read transaction was previously initiated by a write to the PCIOUT endpoint. If the PCI read completes before the USB data packet must be provided, the endpoint will ACK. Otherwise, the endpoint will NAK until the read data becomes valid. If a new OUT (ACKed or otherwise) occurs after the read data becomes valid but before the host performs the IN to collect the data, the read data is invalidated. An IN that returns valid data invalidates the current read data so that the next IN causes a new read.

In USB high-speed mode, the delay from the beginning of the OUT token (for the OUT packet that sets the PCI read address) until the Data IN packet (containing the PCI read data) is ACKed is about 3.6 µs. The delay in full-speed mode is about 28 µs. This assumes no PCI target wait states. Taking into account the host latencies on a typical PC running Windows, the average time to perform a PCI read is about 500 µs (250 µs to set the address and 250 µs to read the data). In full-speed mode it takes about 17 ms (8.5 ms to set the address and 8.5 ms to read the data) to perform the PCI read.

7.6.5.5 STATIN Endpoint

This IN endpoint can be configured as either a BULK or INTERRUPT endpoint. It is used to report a change in the **IRQSTAT1** status register if the corresponding enable bits are set in the **USBIRQENB1** register. Each interrupt status bit has a corresponding interrupt enable bit in the **USBIRQENB1** register. Only status bits with the corresponding enable bits set are reported by this endpoint. The **DEP_CFG** configuration register determines its endpoint number. When configured as a BULK endpoint, the maximum packet size is 8 bytes in full-speed mode and 512 bytes in high-speed mode. When configured as an INTERRUPT endpoint, the maximum packet size is always 4. The interrupt polling rate of this endpoint is determined by the **STATIN_HS_POLL_RATE** and **STATIN_FS_POLL_RATE** registers. Packets read from this endpoint will always have 4 bytes formatted as follows:

Byte Index	7	6	5	4	3	2	1	0
0	IRQSTAT1 register bits 7:0							
1	IRQSTAT1 register bits 15:8							
2	IRQSTAT1 register bits 23:16							
3	IRQSTAT1 register bits 31:24							

If an IN token is received, and the interrupt status hasn't changed since the last IN token, then a NAK handshake is returned to the host.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.7 FIFOs

The NET2280 contains one 4-Kbyte bank of memory that is allocated to endpoint A, B, C, and D FIFOs. There are three FIFO configurations available for this group of endpoints:

- Endpoints A, B, C, and D each have a 1 Kbyte FIFO. This allows each endpoint to operate in Bulk mode with double buffered FIFOs. Each of the endpoints can also be operated in Interrupt or Isochronous mode with single buffered FIFOs, assuming that the maximum packet size is set to 1024 bytes. For Interrupt or Isochronous endpoints with a maximum packet size of 512, double buffering becomes available.
- Endpoints A and B each have a 2 Kbyte FIFO. This allows each endpoint to operate in Isochronous or Interrupt mode with double buffered FIFOs, or in Bulk mode with quadruple buffered FIFOs. Endpoints C and D are **not** available.
- Endpoint A has a 2 Kbyte FIFO, and endpoints B and C each have a 1 Kbyte FIFO. This allows endpoint A to operate in Isochronous or Interrupt mode with a double-buffered endpoint, and endpoints B and C to operate in Bulk mode with double buffered FIFOs or in Isochronous or Interrupt mode with single buffered FIFOs. Endpoint D is **not** available.

Endpoints 0, E, and F each have their own 64 byte FIFO. Data is stored in the FIFOs in 32-bit words, so each entry contains between 1 and 4 bytes.

If a PCI write to a full FIFO or a PCI read from an empty FIFO is detected, a target retry will be generated (STOP# and no TRDY#). For debugging purposes, the *Ignore FIFO Availability* bit of the **FIFOCTL** register can be set. When this bit is set, the FIFO status is ignored, and all PCI accesses are completed with TRDY#. If reading an empty FIFO, unknown data is presented to the PCI bus. If writing to a full FIFO, the data is discarded.

7.7.1 IN Endpoint FIFOs

IN packet data is written by the CPU (8051 or PCI) or DMA into one of the IN endpoint FIFOs. Once the FIFO data has been validated, it is returned to the USB host in response to an IN token. The NET2280 will not send more than **EP_n_MAXPKT** bytes per packet. The CPU (8051 or PCI), DMA controller, of external PCI master can continue loading data for the next packet until the FIFO is full, and the NET2280 will automatically divide the data flow into **EP_n_MAXPKT** packets. This allows USB transactions to overlap with loading of data.

If the FIFO data hasn't been validated, the NET2280 responds to an IN token with a NAK handshake. There are several methods for validating the data in the IN FIFO:

- For large amounts of data, the PCI bus controller can write data to the FIFO as long as there is space available. When there are at least **EP_n_MAXPKT** bytes in the FIFO, the NET2280 will respond to an IN token with a packet of data. If the entire data transfer is a multiple of **EP_n_MAXPKT** bytes, then nothing else needs to be done to validate the FIFO data. If a zero length packet needs to be sent to the host, the CPU (8051 or PCI) can set the *Endpoint Byte Count* field of the **EP_CFG** register to 0 and then write to the **EP_DATA** register.
- When using the internal DMA controller, the DMA byte counter is used. This counter is initialized to the total transfer byte count before any data is written to the FIFO. The counter is decremented as data is written to the FIFO. When the counter reaches zero, the remaining data in the FIFO is validated. Excess bytes in the last word are automatically ignored. If the last packet of a transfer has **EP_n_MAXPKT** bytes, then the NET2280 will respond to the next IN token with a zero length packet.
- For small amounts of data (character oriented applications), the data is first written to the FIFO. The CPU (8051 or PCI) can explicitly validate the packet using the *Endpoint Byte Count* field of the **EP_CFG** register.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

- When using an external bus master (DMA or CPU) to write data into the FIFO, a short packet of length N is validated by writing (N / 4) dwords into the FIFO, then writing (N modulo 4 = 3, 2, 1, or 0) into the *Endpoint Byte Count* field of **EP_CFG**. Then the final dword is written. If more BE bits are set than are indicated by the *Endpoint Byte Count* field, the upper bytes are discarded.
  For example, to validate a 9 byte packet, write two dwords (8 bytes) to **EP_DATA**, then write 1 to the *Endpoint Byte Count* field, then write a byte, word, or dword with the final byte in the least significant location to **EP_DATA**.

Up to 15 short (less than **EP_n_MAXPKT** bytes) packets can be stored in an IN FIFO. A 4-bit counter is incremented when a short packet is validated, and is decremented when a short packet is successfully sent to the host. The counter is cleared by a pin reset, USB reset, FIFO flush, or if SETUP token received (on endpoint 0). For Interrupt or Isochronous endpoints whose maximum packet size is not a multiple of 4, any extra bytes received from the PCI bus during the last Dword transfer of a packet are discarded. For any IN endpoint, if all PCI byte enables of a Dword are not asserted, an end of packet is assumed. The short packet counter is not decremented until the actual End Of Packet (EOP) is detected in the FIFO.

7.7.2 OUT Endpoint FIFOs

When receiving data, the NET2280 will NAK the host (indicating that it cannot accept the data) if either the FIFO runs out of room, or if both the *NAK OUT Packets Mode* bit and the *NAK OUT Packets* bits are set. If the packets received are of maximum size, then additional packets can be received independently of the *NAK OUT Packets Mode* bit. This bit will only cause additional OUT packets to be NAKed if the last packet received was a short packet.

If *NAK OUT Packets Mode* is true (blocking mode), USB OUT transfers can overlap with the CPU (8051 or PCI) unloading the data using the following sequence:

- CPU (8051 or PCI) responds to the *Data Packet Received Interrupt* and reads the **EP_AVAIL** register so it knows how many bytes are in the current packet.
- CPU (8051 or PCI) clears *the Data Packet Received Interrupt* and the *NAK OUT Packets* bit, allowing the next packet to be received.
- Now the CPU (8051 or PCI) can unload data from the FIFO while the next USB OUT transaction is occurring.

If *NAK OUT Packets Mode* is false (non-blocking mode), the NET2280 will accept packets as long as there is room for the complete packet in the FIFO. Note that there are no indications of packet boundaries when there are multiple packets in the FIFO.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

7.8 USB Test Modes

The *Force Full Speed* and *Force High Speed* bits of the **XCVRDIAG** register can be used to force the NET2280 into full and high speed modes, respectively. These forcing bits **must not be used in normal operation**; they are for testing purposed only. In normal operation, the NET2280 automatically performs USB 2.0 Chirp Protocol negotiation with the host to determine the correct operating speed.

USB 2.0 Test Mode support is provided via the *Test Mode Select* field of the **XCVRDIAG** register. These bits select the appropriate USB Test Mode settings (see section 9.4.9 in the USB Specification Revision 2.0 for more details). Normally, the host sends a SET_FEATURE request with the Test Selector in the upper byte of wIndex. The Test Selector can be copied directly into the NET2280 *Test Mode Select* field of the **XCVRDIAG** register to select the correct test mode.

Note that USB Test Mode settings only have an effect if the NET2280 is in high-speed mode. Also, if the NET2280 is in high-speed mode, and the *Test Mode Select* field is set to non-zero, the NET2280 is prevented from switching out of high-speed mode. Normal USB Suspend and Reset, as well as the *Force High Speed* and *Force Full Speed* bits, are ignored for test purposes.

Note also that the NET2280 can be forced into high-speed mode (using the *Force High Speed* bit) even if the NET2280 is not connected to a host controller. After setting high-speed mode, USB Test Modes can be selected.

Most USB Test Modes require no further support from the NET2280 firmware. However, the Test_Packet (0x04) Test Mode Selector requires a specific packet to be returned by the device. The NET2280 will respond correctly by:

1. Set *Test Mode Select* to 0x04
2. Flush endpoint 0
3. Load the following 53 (0x35) byte packet into endpoint 0:

00 00 00 00 00 00 00 00 - 00 AA AA AA AA AA AA AA AA EE EE EE EE EE EE EE –
EE FE FF FF FF FF FF FF FF FF FF FF FF 7F BF DF - EF F7 FB FD FC 7E BF DF EF
F7 FB FD 7E

The packet is validated using the *Endpoint Byte Count* field of the **EP_CFG** register.

Test modes can be auto-responded, in which case the Test TX Packet Test Mode (4) automatically loads the specified test packet, or they can be manually loaded in which case any test packet (up to 64 bytes) may be loaded into the endpoint 0 FIFO.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

8 DMA Controller

8.1 Overview

The NET2280 has four DMA channels that can be used to transfer data between USB endpoint FIFOs and the PCI bus. The four channels are assigned to endpoints A, B, C, and D. Each channel can either perform a single DMA transfer, or can perform a scatter/gather transfer by processing a linked list of descriptors. The starting address of the DMA can be on any byte boundary. **For aligned or unaligned DMA operations, a single DMA transfer should not be used for more than one USB transfer unless all USB packets are a multiple of 4 bytes.**

8.2 Single Transfer Mode

In this mode, a single block of data is transferred between a USB FIFO and the PCI bus.

8.2.1 OUT Endpoints

When a packet is received into an OUT endpoint that has an assigned DMA channel, a request is made to the PCI bus. Once the PCI bus has been granted, the DMA controller reads data from the endpoint FIFO and writes it to a PCI target using PCI burst memory write transactions. If the *Memory Write and Invalidate* enable bits in the **PCIMSTCTL** and **PCICMD** registers are set, and there are at least *Cache Line Size* bytes in the FIFO, and the address is on a cache line boundary, then the NET2280 will issue a Memory Write and Invalidate command. Otherwise, a normal PCI memory write command is issued. These transactions continue until the DMA byte count reaches zero. If the FIFO becomes empty, the DMA controller will pause until more data is available. The following registers must be programmed for a single DMA read transfer:

Register	Description	Notes
DMACTL	DMA Control	Select Single Transfer Mode and other miscellaneous controls
DMACOUNT	DMA Byte Count	Set to number of bytes to transfer and the direction
DMAADDR	DMA Address	Set to PCI target address

Writing to the *DMA Start* bit in the **DMASTAT** register starts the DMA transfer. The DMA transfer can also be automatically started when an OUT packet is received if the *DMA OUT Auto Start Enable* bit in the **DMACTL** register is set**.** When the DMA transfer is complete, various interrupts can be generated.

8.2.2 IN Endpoints

When an IN endpoint is ready to transmit a packet to the host and there is space available in the endpoint FIFO, a request is made to the PCI bus. Once the PCI bus has been granted, the DMA controller reads data from the PCI target using PCI burst read memory transactions and writes it to the endpoint FIFO. If the *Read Line Enable* bit in the **PCIMSTCTL** register is set, and the address is not on a cache line boundary or the *Read Multiple* bit isn't set, and the FIFO has enough space to accept one Cache Line of data, then the NET2280 will issue a Read Line command. If the *Read Multiple Enable* bit in the **PCIMSTCTL** register is set, and the address is on a Cache Line boundary, and the FIFO has enough space to accept one Cache Line of data, then the NET2280 will issue a Read Multiple command. Otherwise, a Read Line (if full cache line can be transferred and *Read Line Enable* bit is set) or normal PCI memory read command is issued. These transactions continue until the DMA byte count reaches zero. If the FIFO becomes full, the DMA controller will pause until more space is available. The following registers must be programmed for a single DMA write transfer:

Register	Description	Notes
DMACTL	DMA Control	Select Single Transfer Mode and other miscellaneous controls
DMACOUNT	DMA Byte Count	Set to number of bytes to transfer and the direction
DMAADDR	DMA Address	Set to PCI target address

Writing to the *DMA Start* bit in the **DMASTAT** register starts the DMA transfer. When the DMA transfer is complete, various interrupts can be generated.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

8.3 Scatter/Gather Mode

In this mode, creating a linked list of descriptors can set up a series of DMA transfers. The list of descriptors is stored in system memory on the PCI bus, with the address of the first descriptor programmed into the **DMADESC** register. Each descriptor consists of the following four Dwords:

Offset	31	30	29	28	27 24	23 4	3 0
0	Valid Bit	Direction	Done Interrupt Enable	End of Chain	Reserved	DMA Byte Count	
4	PCI Starting Address						
8	Next Descriptor Address						0000
C	Reserved						

Writing to the *DMA Start* bit in the **DMASTAT** register starts the DMA controller. The DMA controller then reads the four Dwords at the PCI address determined by the **DMADESC** register. A normal PCI memory read command is used for these transfers. These four Dwords define the first DMA transfer, including the direction, byte count, PCI address, and address of next descriptor. After the DMA transfer completes, additional descriptors are processed if the *End of Chain* bit is not set.

8.3.1 Valid Bit

The *Valid Bit* controls the processing of DMA descriptors by the scatter/gather controller. When the firmware enters a descriptor into the linked list, it sets the *Valid Bit* after the other fields in the descriptor have been written. As the scatter/gather controller processes the descriptor list, it first checks the *Valid Bit* to determine if a descriptor is valid. If the *Valid Bit* is set, then descriptor is considered valid and the corresponding DMA operation is started. If the *Valid Bit* is not set, the descriptor is periodically polled by the scatter/gather controller until the bit is set. The polling rate is determined by the *Descriptor Polling Rate* field of the **DMACTL** register. If desired, the scatter/gather controller can be configured to pause when a cleared *Valid Bit* is detected. In this case, the firmware would need to re-start the scatter/gather controller after setting up and validating additional descriptors.

After a DMA transfer completes, the scatter/gather controller can be configured by the *Clear Count Enable* bit to clear the *Valid Bit*, effectively passing ownership of the descriptor back to the firmware.

If a descriptor is encountered with a *DMA Byte Count* equal to zero, the *Valid Bit* is set, and the *End of Chain* bit is not set, then the scatter/gather controller processes the next descriptor in the chain.

Some applications may not require the use of the *Valid Bit*. In those cases, the *DMA Valid Bit Enable* bit in the **DMACTL** register is set low, allowing descriptors to be processed without regard to the *Valid Bit.*

8.3.2 Clear Count Enable

The **DMACTL** register determines whether the first descriptor Dword is updated after the DMA transfer is completed. If this update is enabled, the *Valid Bit* and *DMA Byte Count* fields are cleared after the DMA transfer is done. If a DMA transfer is aborted prematurely, the number of remaining bytes is available in the *DMA Byte Count* field.

8.3.3 Direction

When this bit is low, the DMA controller transfers data from the USB to PCI bus (OUT packets). When this bit is high, the DMA controller transfers data from the PCI bus to the USB (IN packets).

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

8.3.4 Done Interrupt Enable

When a DMA transfer associated with a descriptor completes (*DMA Byte Count* reaches 0), an interrupt can be generated if the *Done Interrupt Enable* bit is set.

8.3.5 End of Chain

The *End of Chain* bit in the first Dword indicates if there are more descriptors in the chain. If there are more descriptors in the chain, then the third Dword contains the address of the next descriptor.

8.4 OUT Transfer DMA Completion

The *Short Packet OUT Done Interrupt* is set when the *Short Packet Transferred Interrupt* is set and the FIFO becomes empty as a result of DMA writes (from the FIFO to the PCI bus). The consequences for firmware are two:

- *Short Packet OUT Done Interrupt* is not set if firmware clears *Short Packet Transferred Interrupt* before the FIFO becomes empty
- *Short Packet OUT Done Interrupt* is not set if the FIFO is empty and the Net2280 receives a Zero Length Packet (with no data payload).

There are two methods firmware can use to reliably determine when an OUT Transfer DMA is complete. Both methods require *NAK OUT Packets Mode* (bit 2/10 of **EP_RSP**) to be true. Setting *NAK OUT Packets Mode* causes the Net2280 to set *NAK OUT Packets* (bit 7/15 of **EP_RSP** and bit 4 of **EP_STAT**) whenever a short OUT packet is accepted, preventing new packet data from entering the FIFO.

The simplest method for detecting OUT Transfer DMA completion is to interrupt on the *Short Packet Transferred Interrupt*. In the interrupt service routine, firmware polls **EP_AVAIL** (or the *FIFO Empty* status flag) until the FIFO indicates there is no data left. Note that *Short Packet Transferred Interrupt* is set when the new packet enters the FIFO. Polling until the FIFO is empty ensures that the newly arrived data is completely written to the PCI bus. Note also that in most PCI systems, the polling loop will be very short because the DMA is bursting the FIFO data to the PCI bus while the firmware is polling. If the polling takes more than a few loops to complete, it indicates that the DMA is not working (for example: the target is not accepting the DMA cycles).

To avoid polling, firmware can use the second method: interrupt on either *Short Packet Transferred Interrupt* or *Short Packet OUT Done Interrupt*. In the interrupt service routine, check if **EP_AVAIL** (or the *FIFO Empty* status flag) indicates that the FIFO is empty. If it is empty, the DMA is finished and normal OUT transfer completion handling can continue. If the FIFO is not empty, firmware disables the *Short Packet Transferred Interrupt* and returns from the interrupt service routine.

Note that firmware must not *clear* the *Short Packet Transferred Interrupt* at this time - instead, it must disable it by clearing bit 5 of **EP_IRQENB**. When the DMA completes and the FIFO is empty, the *Short Packet OUT Done Interrupt* occurs. The same interrupt service routine now finds the FIFO status to be empty, so normal OUT transfer completion handling can continue.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

8.5 DMA Abort

When the *DMA Abort* bit in the **DMASTAT** register is written, a DMA PCI burst in progress is terminated within a few data transfers and no new transactions are started. If a Memory Write and Invalidate is in progress, then the burst is terminated at the next cache line boundary. The *DMA Enable* bit in the **DMACTL** bit is cleared. If a scatter/gather DMA is in progress, no more descriptor accesses are initiated. After the DMA abort, the associated FIFO should be flushed and the DMA registers re-programmed before attempting another DMA transfer.

A DMA transfer is also aborted when a Master or Target abort is detected, or if the *PCI Retry Abort Enable* bit in the **PCIMSTCTL** register is set and 256 target retries are encountered.

If a DMA abort occurs during a PCI retry, the abort is delayed until the PCI target has either responded with a TRDY#, or the retry timer has timed out.

8.6 DMA Pause

When the *DMA Enable* bit in the **DMACTL** register is cleared, a DMA PCI burst in progress is terminated within a few data transfers. If a Memory Write and Invalidate is in progress, then the burst is terminated at the next cache line boundary. The context of the DMA transfer is maintained while the channel is paused. When the *DMA Enable* bit is set high again, the DMA transfer resumes from where it left off. If the scatter/gather controller is active, it will continue to process descriptors even if the *DMA Enable* bit is cleared, but no DMA data transfers will be initiated.

8.7 PCI Unaligned Write Transfers

DMA write transfers are permitted to start at unaligned PCI address boundaries. For example, if the **DMAADDR** register is initialized to a value of 1 and there are 3 valid lines (12 bytes) in the OUT FIFO, the data is transferred as follows:

FIFO Read

	OUT FIFO Byte Lanes			
FIFO Line	Lane3	Lane2	Lane1	Lane0
0	Byte3	Byte2	Byte1	Byte0
1	Byte7	Byte6	Byte5	Byte4
2	Byte11	Byte10	Byte9	Byte8

PCI Write

	PCI Byte Lanes			
PCI Address	AD[31:24]	AD[23:16]	AD[15:8]	AD[7:0]
‘h0	Byte2	Byte1	Byte0	Not written
‘h4	Byte6	Byte5	Byte4	Byte3
‘h8	Byte10	Byte9	Byte8	Byte7
‘hC	Not written	Not written	Not written	Byte11

After the DMA is completed, the **DMAADDR** register will have a value of ‘hD.

8.7.1 Restrictions

Multiple DMA transfers operating on a continuous stream of USB packets can only be used if the DMA byte count of each transfer is a multiple of 4. For example, if two 512 byte packets are received into an OUT FIFO, and multiple DMA transfers are used to write the packets to the PCI bus, each DMA transfer must have a byte count that is a multiple of 4. The exception to this is at the end of a USB transfer when the last packet is a short packet (less than Max Packet size). The DMA controller has a transfer limit of 16 Mbytes.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

8.8 PCI Unaligned Read Transfers

DMA read transfers are permitted to start at unaligned PCI address boundaries. For example, if the **DMAADDR** register is initialized to a value of 3 and the **DMACOUNT** register is initialized to a value of 16, the data is transferred as follows:

PCI Read

	PCI Byte Lanes			
PCI Address	AD[31:24]	AD[23:16]	AD[15:8]	AD[7:0]
‘h0	Byte0	Ignored	Ignored	Ignored
‘h4	Byte4	Byte3	Byte2	Byte1
‘h8	Byte8	Byte7	Byte6	Byte5
‘hC	Byte12	Byte11	Byte10	Byte9
‘h10	Ignored	Byte15	Byte14	Byte13

FIFO Write

	IN FIFO Byte Lanes			
FIFO Line	Lane3	Lane2	Lane1	Lane0
0	Byte3	Byte2	Byte1	Byte0
1	Byte7	Byte6	Byte5	Byte4
2	Byte11	Byte10	Byte9	Byte8
2	Byte15	Byte14	Byte13	Byte12

Bytes 3:0 are written to the first line in the FIFO, bytes 7:4 to the second line, and so on. After the DMA is completed, the **DMAADDR** register will have a value of ‘h13.

8.8.1 Restrictions

Multiple DMA transfers operating on a continuous stream of USB packets can only be used if the DMA byte count is a multiple of 4. For example, if two 512 byte packets are written into an IN FIFO, and multiple DMA transfers are used to read the packets from the PCI bus, each DMA transfer must have a byte count that is a multiple of 4. The exception to this is at the end of a USB transfer when the last packet is a short packet (less than Max Packet size). The DMA controller has a transfer limit of 16 Mbytes.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

9 Interrupt and Status Register Operation

9.1 Overview

There are many sources of interrupts from the USB, PCI, DMA, and EEPROM sections of the chip. Each of these interrupt sources can be routed to the internal 8051 CPU, the PCI INTA# interrupt pin, or the STATIN interrupt endpoint. Each interrupt source has three enable bits: one for the 8051, one for the PCI INTA# pin, and one for the STATIN interrupt endpoint. When the NET2280 is configured for PCI Host Mode, then the INTA# pin is an input and all interrupts are serviced by the 8051 CPU or by the USB host via the STATIN interrupt endpoint. The 8051 has two interrupt inputs. One interrupt is asserted when one of the **IRQSTAT0** interrupts is active, and the other when one of the **IRQSTAT1** interrupts is active.

9.2 Interrupt Status Registers (IRQSTAT0 and IRQSTAT1)

Bits 6:0 of the **IRQSTAT0** register indicate whether one of the endpoints 0, A-F has an interrupt pending. These bits cannot be written, and can cause an interrupt if the corresponding interrupt enable bits are set in the **PCIIRQENB0** or **CPUIRQENB0** registers. Bit 7 indicates whether a Setup packet has been received, and is cleared by writing a 1.

Bits 0 of **IRQSTAT1** is automatically set when a start-of-frame (SOF) token is received, and is cleared by writing a 1. Note that the interrupt bits can be set without the corresponding interrupt enable bit being set. This allows the CPU (8051 or PCI) to operate in a polled, as well as an interrupt driven environment.

Bits 8:1 of the **IRQSTAT1** register are set when a particular event occurs in the NET2280, and are cleared by writing a 1 to the corresponding bit. Bits 27:16 of **IRQSTAT1** are related to the PCI bus interface. These bits can cause an interrupt if the corresponding interrupt enable bits are set in the **PCIIRQENB1, CPUIRQENB1**, or **USBIRQENB1** registers. Bit 3 of **IRQSTAT1** is set when there is a suspend request from the host, but typically is not enabled to generate an interrupt. Writing a 1 clears this bit and causes the NET2280 to enter the suspend state.

Bits 12:9 of the **IRQSTAT1** register indicate whether one of the DMA channels has an interrupt pending. These bits cannot be written, and can cause an interrupt if the corresponding interrupt enable bits are set in the **PCIIRQENB1**, **CPUIRQENB1,** or **USBIRQENB1** registers.

Bit 13 of the **IRQSTAT1** register indicates whether one of the GPIO bits has an interrupt pending. These bits cannot be written, and can cause an interrupt if the corresponding interrupt enable bits are set in the **PCIIRQENB1**, **CPUIRQENB1,** or **USBIRQENB1** registers.

9.3 Endpoint Response Registers (EP_RSP)

Each configurable endpoint has an Endpoint Response Register. The bits in this register determine how the NET2280 will respond to various situations during a USB transaction. Writing a 1 to any of the bits 7:0 in the **EP_RSP** register will clear the corresponding bits. Writing a 1 to any of the bits 15:8 in the **EP_RSP** register will set the corresponding bits. Reading either byte 0 or byte 1 of the register returns the current state of the bits.

9.4 Endpoint Status Register (EP_STAT)

Each endpoint has an Endpoint Status Register**.** Each of the bits of this register is set when a particular endpoint event occurs, and is cleared by writing a 1 to the corresponding bit. Bits 7:0 can cause an interrupt if the corresponding interrupt enable bits are set in the **EP_IRQENB** register. Reading the **EP_STAT** register returns the current state of the bits.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10 Power Management

10.1 Overview

The NET2280 supports the PCI Bus Power Management Interface Specification, Rev 1.1 and the USB power management requirements detailed in the Revision 2.0 USB specification.

10.2 USB Power Configurations

The USB specification defines both bus-powered and self-powered devices. A *bus-powered* device is a peripheral that derives all of its power from the upstream USB connector, while a *self-powered* device has an external power supply.

The most significant consideration when deciding whether to build a bus-powered or a self-powered device is power consumption. The USB specification dictates the following requirements for maximum current draw:

- A device not configured by the host can draw only 100 mA from the USB power pins.
- A device may not draw more than 500 mA from the USB connector's power pins.
- In suspend mode, the device may not draw more than 500 μA (for low-power device) or 2.5mA (for high-power device) from the USB connector's power pins

If these power considerations can be met without the use of an external power supply, the device can be bus-powered; otherwise a self-powered design should be implemented.

10.2.1 Self-Powered Device

Generally, a device with higher power requirements will be self-powered. In a self-powered device, the NET2280 VDD pins are powered by the local power supply. This allows the PCI bus to continue accessing the NET2280, even when the device is not connected to the USB bus. The USB connector's power pin is connected only to the VBUS pin.

While the device is connected to the USB, the NET2280 will automatically request suspend mode when appropriate, as described in section 10.3. The NET2280 should not be powered-down when its PCI bus is still connected to a powered-up device. There are ESD protection circuits in the NET2280 that will short VDD pins to ground. If the VDD pins are not powered, they will sink too much current from the board.

10.3 USB Suspend Mode

When there is a three-millisecond period of inactivity on the USB, the USB specification requires a bus-powered device to enter into a low-power suspended state. The device may not draw more than 500 μA (low-power device) or 2.5 mA (high-power device) while in this state. This requirement only applies to bus-powered devices. To facilitate this, the NET2280 provides a *Suspend Request Change Interrupt* bit and a *Suspend Request Interrupt* bit. The USB suspend/resume sequence is also supported by the PCI power management features of the NET2280.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10.4 PCI Power Management

The NET2280 provides the configuration registers and support hardware required by the PCI Power Management Specification. The *Capability Valid* bit in the **PCISTAT** register must be high for power management to be enabled. The **PCICAPPTR** register points to the base address of the power management registers (40h in the NET2280).

10.4.1 Power States

The following power states are supported by the NET2280, and are selected by the *Power State* field in the **PWRMNGCSR** register:

- D0: fully operational. This state requires the most current.
- D1: light sleep. Only PCI configuration transactions are accepted. No master cycles are allowed, and the INTA# interrupt is disabled. The PME# pin can be asserted by the NET2280. The PCI clock must continue to run in this state. The PCI host should only put the NET2280 into this state if a USB suspend request has been detected.
- D2: heavy sleep. Same as D1, except that the PCI host can stop the PCI clock.
- D3: function context not maintained. In this state, the USB interface is suspended and the device is logically disconnected from the USB host (*USB Detect Enable* bit in the **USBCTL** register is clear). Only PCI configuration transactions are accepted. From this state, the next power state can only be D0. When transitioning from D3 to D0, the entire chip is reset.

An interrupt, indicated by the *Power State Change Interrupt* bit in the **IRQSTAT1** register, can be generated when the power state is changed.

10.5 USB Suspend/Resume for PCI Host Mode

10.5.1 Suspend Sequence

If the 8051 is running, perform the following suspend procedure when operating in PCI host mode.

- During configuration register initialization, the *Suspend Request Change Interrupt* bit in the **CPUIRQENB1** register is enabled to generate an 8051 interrupt.
- When the USB is idle for three milliseconds, the NET2280 sets the *Suspend Request Change Interrupt* bit, generating an interrupt to the 8051. This interrupt can also occur if the NET2280 is not connected to a host, and the USB data lines are pulled to the idle state (DP high, DM low).
- The 8051 accepts this interrupt by clearing the *Suspend Request Change Interrupt* bit, and performs the tasks required to ensure that not more than 2.5 mA of current is drawn from the USB power bus. This may include putting other PCI devices into a suspend state and stopping the PCI clock to other devices.
- The 8051 writes a 1 to the *Suspend Request Interrupt* bit to initiate the suspend. After 500 μsec, the 30 MHz USB oscillator stops and the POSCENB pin goes low. This pin can be used to disable the external 33 MHz PCI clock oscillator.
- A device remote wakeup event will not be recognized during the 500 μsec suspend delay period.

If the *Suspend Immediately* bit in the **USBCTL** register is set, the NET2280 suspends automatically:

- When the USB is idle for three milliseconds, the NET2280 automatically initiates the suspend sequence. After 500 μsec, the 30 MHz USB oscillator stops, and the POSCENB pin goes low. This pin can be used to disable the external 33 MHz PCI clock oscillator.
- A device remote wakeup event will not be recognized during the 500 μsec suspend delay period.

If a device is self-powered, it may ignore the USB suspend request and never write a 1 to the *Suspend Request Interrupt* bit. Note that input pins on the NET2280 that do not have an internal pull-up or pull-down resistors should not be allowed to float during suspend mode.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10.5.2 Host Initiated Wake-Up

The host may wake up the NET2280 by driving any non-idle state on the USB. The NET2280 will detect the host's wake-up request, re-start its internal oscillator, and drive the POSCENB pin high. The host initiated wake-up is only recognized if the VBUS input pin is high, and the *USB Detect Enable* and *USB Root Port Wakeup Enable* bits in the **USBCTL** register are set. The 8051 gets a resume interrupt and powers up the other PCI devices and associated clocks. The NET2280 GPIO pins can be used for controlling power and clocks to other devices.

10.5.3 Device-Remote Wake-Up

Another PCI device can initiate a USB device remote wakeup by asserting the PME# input to the NET2280. If the *PME Wake-up Enable* bit in the **USBCTL** register is set, the NET2280 re-starts its USB oscillator and drives the POSCENB pin high. After the NET2280 has completed its wake-up, the 8051 must write to the *Generate Resume* bit of the **USBSTAT** register. This will cause a 10-ms wake-up signal to be sent to the USB host. If the 8051 is held in the reset state, the resume is initiated automatically when PME# is asserted and the clocks are running.

10.5.4 Resume Interrupt

When the NET2280 begins either a Device-Remote Wake-Up or Host-Initiated Wake-Up, it may generate a resume interrupt, if enabled. The *Resume Interrupt* bit of the **IRQSTAT** register is set when a resume is detected, and can be enabled to generate an interrupt with the *Resume Interrupt Enable* bit.

10.6 USB Suspend/Resume for PCI Adapter Mode

10.6.1 Suspend Sequence with 8051 held in reset

- When a USB host suspend condition is detected, the NET2280 asserts the PME# pin to the PCI host.
- The PCI host puts the NET2280 into power state D1 by writing to the **PWRMNGCSR** register. After 500 µsec, the 30 MHz USB oscillator stops and the POSCENB pin goes low. This pin can be used to disable the external 33 MHz PCI clock oscillator.
- A device remote wakeup event will not be recognized during the 500 µsec suspend delay period.

If a device is self-powered, the PCI host may ignore the PME# and never put the NET2280 into the D1 power state.

10.6.2 Host Initiated Wake-Up with 8051 held in reset

The USB host may wake up the NET2280 by driving any non-idle state on the USB. The NET2280 will detect the host's wake-up request, re-start its internal oscillator, and drive the POSCENB pin high. The host initiated wake-up is only recognized if the VBUS input pin is high, and the *USB Detect Enable* and *USB Root Port Wakeup Enable* bits in the **USBCTL** register are set. The NET2280 asserts the PME# signal to the PCI host. The PCI host then changes the NET2280 power state to D0. It is now ready to process USB packets again.

If the PCI host needs to wake up the NET2280 in the absence of any USB host activity, it can just change the power state to D0. Since this can only happen if the PCI clock is running, the POSCENB would not be used to disable the PCI oscillator, and the oscillator would need to be left running during suspend. The PCI host can also wake up the NET2280 by writing a 1 to the *Generate Device Remote Wakeup* bit in the **USBSTAT** register.

10.6.3 Suspend Sequence with 8051 operating

- When a USB host suspend condition is detected, the 8051 receives the *Suspend Request Change Interrupt* and writes to the **PWRMNGCSR** register to assert the PME# pin to the PCI host.
- The PCI host puts the NET2280 into power state D1 by writing to the **PWRMNGCSR** register. After 500 µsec, the 30 MHz USB oscillator stops and the POSCENB pin goes low. This pin can be used to disable the external 33 MHz PCI clock oscillator.
- A device remote wakeup event will not be recognized during the 500 µsec suspend delay period.

If a device is self-powered, the PCI host may ignore the PME# and never put the NET2280 into the D1 power state.

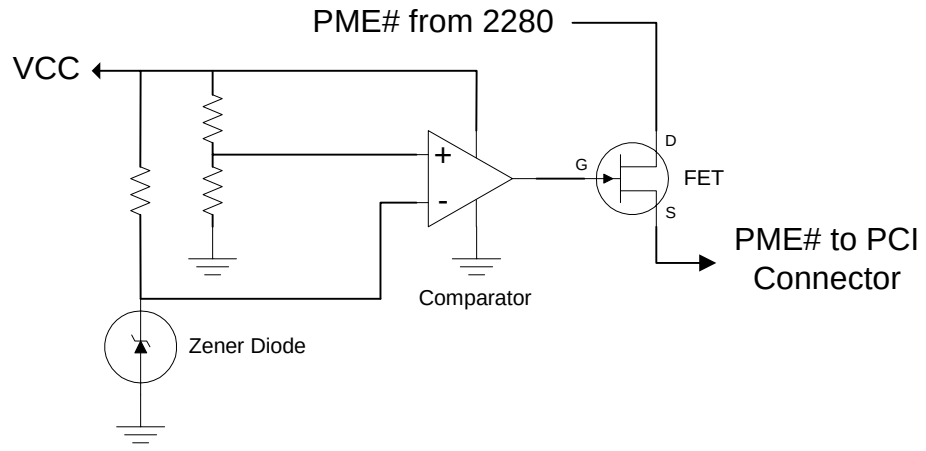
335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10.6.4 Host Initiated Wake-Up with 8051 operating

The USB host may wake up the NET2280 by driving any non-idle state on the USB. The NET2280 will detect the host's wake-up request, re-start its internal oscillator, and drive the POSCENB pin high. The host initiated wake-up is only recognized if the VBUS input pin is high, and the *USB Detect Enable* and *USB Root Port Wakeup Enable* bits in the **USBCTL** register are set. The 8051 receives the *Resume Interrupt* and writes to the **PWRMNGCSR** register to assert the PME# pin to the PCI host. The PCI host then changes the NET2280 power state to D0. It is now ready to process USB packets again.

10.6.5 PME Isolation

When operating in PCI Adapter mode with PME# connected to the host, the Net2280 PME# output pin needs to be isolated from the host when no power is applied. The following circuit is a general example of an isolation circuit. When no power is applied, the gate to the FET is low, thus isolating the Net2280 PME# pin from the host. If the Net2280 is operating in USB self-powered mode, then the power management state doesn't need to be changed from D0, and the PME# signal does not need to be connected to the PCI host. Refer to the PCI Bus Power Management Interface Specification Revision 1.1, Chapter 7, for more details.



335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10.7 NET2280 Low-Power Modes

10.7.1 USB Suspend (Unplugged from USB)

The NET2280 may draw a small amount of power when disconnected from the USB. Disconnecting from the USB can be accomplished in two different ways:

- Un-plug the USB cable.
- Clear the *USB Detect Enable* bit in the **USBCTL** register.

In power-sensitive applications, the CPU (8051 or PCI) can force the NET2280 to enter low-power suspend mode when disconnected from the USB by writing a 1 to the *Suspend Request Interrupt* bit. The NET2280 will automatically wake-up when the peripheral is re-connected (cable plugged in and USB Detect Enable bit set) to the USB. *Do not force suspend mode unless the peripheral is disconnected from the USB. When the NET2280 is connected to the USB, it is a violation of the USB specification to enter the suspend state unless the upstream port has been idle for at least 3 milliseconds.*

This is the preferred method of suspending the NET2280, since a USB re-connection will automatically cause the NET2280 to wake-up and set the *Resume Interrupt* bit.

10.7.2 Power-On Standby

An external device can prevent the NET2280 from starting its oscillator on power-up by driving a LOW into the PWRDOWN# pin. In this state the NET2280 requires only a small quiescent standby current.

When the external device wishes to start the oscillator, it releases the PWRDOWN# pin and asserts RESET# for a minimum of 2 milliseconds. Note that while the oscillator is stopped, the NET2280 cannot respond to USB requests, so the oscillator must be allowed to start when the external device detects a USB connection event. The CPU (8051 or PCI) is responsible for detecting the connection, and ending the standby condition.

This standby technique is appropriate when the device's power budget does not allow the NET2280 to be active long enough to shut it down by setting the *Suspend Request Interrupt* bit.

10.8 CLKRUN# Pin

The CLKRUN# pin is used by the NET2280 to detect when a PCI central resource is attempting to slow down or stop the PCI clock. If the NET2280 is not suspended and detects that the PCI central resource has driven CLKRUN# high, it drives CLKRUN# low for two clocks. This indicates to the PCI central resource that the NET2280 wants the PCI clock to be maintained at 33 MHz.

If the NET2280 is suspended, it lets the PCI central resource stop the clock. When a resume event is detected in the NET2280, the CLKRUN# pin is driven low. In response, the PCI central resource restarts the PCI clock. The NET2280 continues to assert CLKRUN# until 2 consecutive PCI clock cycles are recognized.

In PCI Host mode, the NET2280 asserts CLKRUN# continuously, indicating that it never intends to stop the clock.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11 Configuration Registers

11.1 Register Description

The PCI Configuration Registers are accessed by the host using the configuration address space, while the Memory Mapped Configuration Registers are accessed using the 64 Kbyte memory space defined by PCI Base Address 0. The indexed registers are accessed using the **Indexed Register Address** and **Data** registers. The USB interface and 8051 CPU can also access all of the configuration registers. Each register is 32-bits wide, and can be accessed a byte, word, or Dword at a time.

Note: If the *PCI Enable* bit of the **DEVINIT** register is not set, the NET2280 will terminate PCI transactions with a retry.

These registers utilize little endian byte ordering which is consistent with the PCI Local Bus Specification. The least significant byte in a Dword is accessed at address 0. The least significant bit in a Dword is 0, and the most significant bit is 31.

After the NET2280 is powered-up or reset, the registers are set to their default values. Writes to unused registers are ignored, and reads from unused registers return a value of 0. For compatibility with future revisions, **reserved** bits within a register should always be written with a zero.

11.2 Access Designators

The following designators are used to indicate the type of access provided by each register bit.

Read/Write Register Access Designators

- r = can be read
- w = can be written
- u = can be updated by hardware

Set and Clr Register Access Designators

- r = can be read
- s = can be set
- c = can be cleared
- u = can be updated by hardware

Any register bit that is writeable by the 8051 or USB is also writeable by the EEPROM controller.

11.3 Register Summary

Register Group	PCI Space	Address Range
PCI Configuration Registers	Configuration	000-0FFh
Main Control Registers	Memory mapped, PCIBASE0	000-07Fh
USB Control Registers	Memory mapped, PCIBASE0	080-0FFh
PCI Control Registers	Memory mapped, PCIBASE0	100-17Fh
DMA Control Registers	Memory mapped, PCIBASE0	180-1FFh
Dedicated Endpoint Registers	Memory mapped, PCIBASE0	200-27Fh
Reserved		280-2FFh
Configurable Endpoint Registers	Memory mapped, PCIBASE0	300-3FFh

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.3.1 PCI Configuration Registers

PCI Configuration Register Address	31 24	23 16	15 8	7 0
00h	Device ID		Vendor ID	
04h	Status		Command	
08h	Class Code			Revision ID
0Ch	BIST	Header Type	Latency Timer	Cache Line Size
10h	PCI Base Address 0 for Memory Mapped Configuration Registers			
14h	PCI Base Address 1 for Memory Mapped 8051 Program RAM			
18h	PCI Base Address 2 for Memory Mapped FIFO accesses			
1Ch	Unused Base Address			
20h	Unused Base Address			
24h	Unused Base Address			
28h	Cardbus CIS Pointer (Not Supported)			
2Ch	Subsystem ID		Subsystem Vendor ID	
30h	PCI Base Address for Expansion ROM (Not Supported)			
34h	Reserved			Capabilities Ptr
38h	Reserved			
3Ch	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line
40h	Power Management Capabilities		Next Item Ptr	Capability ID
44h	Data	Bridge Extensions	Power Management CSR	

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.3.2 Main Control Registers

Address	Register Name	Register Description	Page
00h	DEVINIT	Device Initialization	84
04h	EECTL	EEPROM Control	85
08h	EECLKFREQ	EEPROM Clock Frequency	85
0Ch	Reserved		
10h	PCIIRQENB0	PCI Interrupt Request Enable 0	86
14h	PCIIRQENB1	PCI Interrupt Request Enable 1	86
18h	CPUIRQENB0	CPU Interrupt Request Enable 0	87
1Ch	CPUIRQENB1	CPU Interrupt Request Enable 1	88
20h	Reserved		
24h	USBIRQENB1	USB (STATIN) Interrupt Request Enable 1	89
28h	IRQSTAT0	Interrupt Request Status 0	90
2Ch	IRQSTAT1	Interrupt Request Status 1	90
30h	IDXADDR	Indexed Register Address	92
34h	IDXDATA	Indexed Register Data	92
38h	FIFOCTL	FIFO Control	92
3Ch	Reserved		
40h	MEMADDR	FIFO Memory Diagnostic Address	93
44h	MEMDATA0	FIFO Memory Diagnostic Data 0	93
48h	MEMDATA1	FIFO Memory Diagnostic Data 1	93
4Ch	Reserved		
50h	GPIOCTL	General-Purpose I/O Control	94
54h	GPIOSTAT	General-Purpose I/O Status	94
58-7Fh	Reserved		

11.3.3 USB Control Registers

Address	Register Name	Register Description	Page
80h	STDRSP	Standard Response	95
84h	PRODVENDID	Product/Vendor ID	96
88h	RELNUM	Release Number	96
8Ch	USBCTL	USB Control	97
90h	USBSTAT	USB Status	98
94h	XCVRDIAG	USB Transceiver Diagnostic Port	98
98h	SETUP0123	Setup bytes 0,1,2,3	99
9Ch	SETUP4567	Setup bytes 4,5,6,7	99
A0h	Reserved		
A4h	OURADDR	Our USB address	100
A8h	OURCONFIG	Our USB configuration	100
ACh-FFh	Reserved		

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.3.4 PCI Control Registers

Address	Register Name	Register Description	Page
100h	PCIMSTCTL	PCI Master Control	101
104h	PCIMSTADDR	PCI Master Address	101
108h	PCIMSTDATA	PCI Master Data	102
10Ch	PCIMSTSTAT	PCI Master Status	102
110-17Fh	Reserved		

11.3.5 DMA Control Registers

Register Name	Register Description	Page
DMACTL	DMA Control	103
DMASTAT	DMA Status	104
DMACOUNT	DMA Byte Count	104
DMAADDR	DMA Address	104
DMADESC	DMA Descriptor	104

Register	Channel A	Channel B	Channel C	Channel D
DMACTL	180h	1A0h	1C0h	1E0h
DMASTAT	184h	1A4h	1C4h	1E4h
DMACOUNT	190h	1B0h	1D0h	1F0h
DMAADDR	194h	1B4h	1D4h	1F4h
DMADESC	198h	1B8h	1D8h	1F8h

11.3.6 Dedicated Endpoint Registers

Register Name	Register Description	Page
DEP_CFG	Endpoint Configuration	105
DEP_RSP	Endpoint Response	105

Register	CFGOUT	CFGIN	PCIOUT	PCIIN	STATIN
DEP_CFG	200h	210h	220h	230h	240h
DEP_RSP	204h	214h	224h	234h	244h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.3.7 Configurable Endpoint / FIFO Registers

Register Name	Register Description	Page
EP_CFG	Endpoint Configuration	106
EP_RSP	Endpoint Response	107
EP_IRQENB	Endpoint Interrupt Enable	108
EP_STAT	Endpoint Status	109
EP_AVAIL	Endpoint Available Count	110
EP_DATA	Endpoint DATA	110

Register	EP 0	EP A	EP B	EP C	EP D	EP E	EP F
EP_CFG	300h	320h	340h	360h	380h	3A0h	3C0h
EP_RSP	304h	324h	344h	364h	384h	3A4h	3C4h
EP_IRQENB	308h	328h	348h	368h	388h	3A8h	3C8h
EP_STAT	30Ch	32Ch	34Ch	36Ch	38Ch	3ACh	3CCh
EP_AVAIL	310h	330h	350h	370h	390h	3B0h	3D0h
EP_DATA	314h	334h	354h	374h	394h	3B4h	3D4h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.3.8 Indexed Registers

Index	Register Name	Register Description	Page
00h	DIAG	Diagnostic Controls	111
01h	PKTLEN	Packet Length	111
02h	FRAME	Frame Count	111
03h	CHIPREV	Chip Revision	111
04-05h		Reserved	
06h	HS_MAXPOWER	High-Speed Maximum Power	112
07h	FS_MAXPOWER	Full-Speed Maximum Power	112
08h	HS_INTPOLL_RATE	High-Speed Interrupt Polling Rate	112
09h	FS_INTPOLL_RATE	Full-Speed Interrupt Polling Rate	112
0Ah	HS_NAK_RATE	High-Speed NAK Rate	112
0Bh	SCRATCH	Scratch-pad register	112
0Ch-1Fh		Reserved	
20h	EP_A_HS_MAXPKT	Maximum High-Speed packet size	113
21h	EP_A_FS_MAXPKT	Maximum Full-Speed packet size	113
22h-2Fh		Reserved	
30h	EP_B_HS_MAXPKT	Maximum High-Speed packet size	113
31h	EP_B_FS_MAXPKT	Maximum Full-Speed packet size	113
32h-3Fh		Reserved	
40h	EP_C_HS_MAXPKT	Maximum High-Speed packet size	114
41h	EP_C_FS_MAXPKT	Maximum Full-Speed packet size	114
42h-4Fh		Reserved	
50h	EP_D_HS_MAXPKT	Maximum High-Speed packet size	114
51h	EP_D_FS_MAXPKT	Maximum Full-Speed packet size	114
52h-5Fh		Reserved	
60h	EP_E_HS_MAXPKT	Maximum High-Speed packet size	115
61h	EP_E_FS_MAXPKT	Maximum Full-Speed packet size	115
62h-6Fh		Reserved	
70h	EP_F_HS_MAXPKT	Maximum High-Speed packet size	115
71h	EP_F_FS_MAXPKT	Maximum Full-Speed packet size	115
72h-7Fh		Reserved	
80h-83h		Reserved	
84h	STATIN_HS_INTPOLL_RATE	STATIN Endpoint High-Speed Interrupt Polling Rate	116
85h	STATIN_FS_INTPOLL_RATE	STATIN Endpoint Full-Speed Interrupt Polling Rate	116
86h-FFh		Reserved	

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4 PCI Configuration Registers

11.4.1 (Address 00h; PCIVENDID) PCI Vendor ID

Bits	Description	PCI	USB/8051	Default
15:0	**PCI Vendor ID**. This field identifies the manufacturer of the device.	r	rw	17CCh

11.4.2 (Address 02h; PCIDEVID) PCI Device ID

Bits	Description	PCI	USB/8051	Default
15:0	**PCI Device ID**. This field identifies the particular device, as specified by the Vendor.	r	rw	2280h

11.4.3 (Address 04h; PCICMD) PCI Command

Bits	Description	PCI	USB/8051	Default
15:11	*Reserved*	r	r	0
10	**Interrupt Disable.** When set, this bit disables INTA# from being asserted.	rw	rw	0
9	**Fast Back to Back Enable.** This bit is set by initialization, and indicates to the device's bus master logic that it can perform fast back-to-back transactions to different devices. This feature is not supported, so this bit is forced to 0.	r	r	0
8	**SERR# Enable.** This bit enables the system error pin SERR# to be asserted.	rw	rw	0
7	**Address Stepping Enable.** This bit controls whether the device does address stepping. This feature is not implemented, so this bit is forced to 0.	r	r	0
6	**Parity Checking Enable.** This bit enables PCI parity checking.	rw	rw	0
5	**VGA Palette Snoop.** This feature is not supported.	r	r	0
4	**Memory Write and Invalidate Enable.** When set, this bit enables the bus master logic to use the Memory Write-and-Invalidate command. When clear, the Memory Write command is used instead.	rw	rw	0
3	**Special Cycles.** This bit determines whether a device responds to Special Cycles. This feature is not implemented, so this bit is forced to 0.	r	r	0
2	**Bus Master Enable.** This bit enables the device to act as a PCI bus master, and should be set to a 1.	rw	rw	0
1	**Memory Space Enable.** This bit enables the device to respond to Memory space accesses, and should be set to a 1.	rw	rw	0
0	**IO Space Enable.** This bit enables the device to respond to I/O space accesses. I/O cycles are not used in this implementation, so this is forced to a 0.	r	r	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4.4 (Address 06h; PCISTAT) PCI Status

Bits	Description	PCI	USB/8051	Default
15	**Parity Error Detected.** This bit is set whenever the device detects a parity error on incoming addresses or data from the PCI bus. It is cleared by writing a 1. This bit is set independently of the *PCI Checking Enable* bit.	rcu	rcu	0
14	**SERR Asserted.** This bit is set whenever the device asserts the SERR# pin. Writing a 1 clears this bit.	rcu	rcu	0
13	**Master Abort Received.** This bit is set whenever the device is acting as a bus master, and has its transaction terminated with a Master-Abort. A Master-Abort occurs when no target responds with a DEVSEL. Writing a 1 clears this bit.	rcu	rcu	0
12	**Target Abort Received.** This bit is set whenever the device is acting as a bus master, and has its transaction terminated with a Target-Abort. A Target-Abort occurs when the target detects a fatal error and is unable to complete the transaction. It de-asserts DEVSEL# and asserts STOP# to signal the Target Abort. Writing a 1 clears this bit.	rcu	rcu	0
11	**Target Abort Asserted.** This bit is set whenever the device is acting as a bus target, and terminates a transaction with a Target-Abort. A Target-Abort occurs when the target detects a fatal error and is unable to complete the transaction. This never occurs on the NET2280, so a zero is always returned.	r	r	0
10:9	**Devsel Timing.** This field determines how quickly this device responds to a transaction with DEVSEL. A value of 1 indicates a medium response.	r	r	1
8	**Data Parity Error Detected.** This bit indicates that a data parity error has occurred when this device was the bus master. The *PCI Checking Enable* bit in the **PCICMD** register must be set for this bit to be set. Writing a 1 clears this bit.	rcu	rcu	0
7	**Fast Back-To-Back.** This bit indicates if this device can accept fast back-to-back transactions.	r	rw	0
6	**User Definable.** This device does not support user definable features, so this bit is forced to 0.	r	r	0
5	**66MHz Capable.** This device does not support 66 MHz, so this bit is forced to 0.	r	r	0
4	**Capability Valid.** This bit indicates if the **New Capabilities Pointer** at address 34h is valid.	r	rw	0
3	**Interrupt Status.** This bit reflects the state of the NET2280 INTA# interrupt status. The INTA# signal is asserted when this bit is high and the *Interrupt Disable* bit in the **PCI Command** register is low.	r	r	0
2:0	*Reserved*	r	r	0

11.4.5 (Address 08h; PCIDEVREV) PCI Device Revision ID

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Device Revision ID.** This field identifies the silicon revision of the device.	r	r	---

11.4.6 (Address 09h; PCICLASS) PCI Class Code

Bits	Description	PCI	USB/8051	Default
23:16	**Base Class.** 0Ch = Serial Bus Controller	r	rw	0Ch
15:8	**Sub Class.** 03h = USB Device	r	rw	03h
7:0	**Interface.** FEh = USB Device (not host controller).	r	rw	FEh

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4.7 (Address 0Ch; PCICACHESIZE) PCI Cache Line Size

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Cache Line Size.** This field specifies, in units of 32-bit words, the cache line size used during Read Line, Read Multiple, and Write and Invalidate commands. Only values of 8, 16, and 32 are supported. Writes of values other than these are ignored.	rw	rw	0

11.4.8 (Address 0Dh; PCILATENCY) PCI Bus Latency Timer

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Bus Latency Timer.** This field specifies, in units of PCI clocks, the value of the Latency Timer during bus master bursts. When the Latency Timer expires, the device must terminate its tenure on the bus.	rw	rw	0

11.4.9 (Address 0Eh; PCIHEADER) PCI Header Type

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Header Type.** This field specifies the format of the second part of the predefined configuration header starting at address 10h.	r	r	0

11.4.10 (Address 0Fh; PCIBIST) PCI Built-in Self Test

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Built-in Self Test.** The build-in-self-test function is not supported, and always returns a value of 0.	r	r	0

11.4.11 (Address 10h; PCIBASE0) PCI Base Address 0

Bits	Description	PCI	USB/8051	Default
31:16	**Base Address.** Specifies the upper 16 bits of the 32-bit starting base address of the 64 Kbyte addressing space for the NET2280 Configuration Registers.	rw	rw	0
15:4	**Base Address.** This part of the base address is ignored for a 64 Kbyte space. *Note: **Hardcoded to 0.***	r	r	0
3	**Prefetch Enable.** When set, indicates that pre-fetching has no side effects on reads. *Note: **Hardcoded to 0.***	r	r	0
2:1	**Address Type.** This field indicates the type of addressing for this space. 00 = locate anywhere in 32-bit address space (default) 01 = locate below 1 Meg 10 = locate anywhere in 64-bit address space 11 = reserved *Note: **Hardcoded to 0.***	r	r	0
0	**Space Type.** When low, this space is accessed as memory. When high, this space is accessed as I/O. *Note: **Hardcoded to 0.***	r	r	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4.12 (Address 14h; PCIBASE1) PCI Base Address 1

Bits	Description	PCI	USB/8051	Default
31:16	**Base Address.** Specifies the upper 16 bits of the 32-bit starting base address of the 64 Kbyte addressing space for the 8051 Program RAM.	rw	rw	0
15:4	**Base Address.** This part of the base address is ignored for a 64 Kbyte space. *Note:* ***Hardcoded to 0.***	r	r	0
3	**Prefetch Enable.** When set, indicates that pre-fetching has no side effects on reads.	r	rw	1
2:1	**Address Type.** This field indicates the type of addressing for this space. 00 = locate anywhere in 32-bit address space (default) 01 = locate below 1 Meg 10 = locate anywhere in 64-bit address space 11 = reserved *Note:* ***Hardcoded to 0.***	r	r	0
0	**Space Type.** When low, this space is accessed as memory. When high, this space is accessed as I/O. *Note:* ***Hardcoded to 0.***	r	r	0

11.4.13 (Address 18h; PCIBASE2) PCI Base Address 2

Bits	Description	PCI	USB/8051	Default
31:16	**Base Address.** Specifies the upper 16 bits of the 32-bit starting base address of the space for endpoint FIFO accesses. The range of this space is determined by the *PCI Base2 Range* field of the **FIFOCTL** register.	rw	rw	0
15:4	**Base Address.** This part of the base address is ignored, thus allowing increments of 64 Kbytes. *Note:* ***Hardcoded to 0.***	r	r	0
3	**Prefetch Enable.** When set, indicates that pre-fetching has no side effects on reads.	r	rw	0
2:1	**Address Type.** This field indicates the type of addressing for this space. 00 = locate anywhere in 32-bit address space (default) 01 = locate below 1 Meg 10 = locate anywhere in 64-bit address space 11 = reserved *Note:* ***Hardcoded to 0.***	r	r	0
0	**Space Type.** When low, this space is accessed as memory. When high, this space is accessed as I/O. *Note:* ***Hardcoded to 0.***	r	r	0

11.4.14 (Address 1Ch, 20h, 24h; PCIBASE3, 4, 5) PCI Base Address 3-5

Bits	Description	PCI	USB/8051	Default
31:0	**PCI Base Address 3, 4, 5.** These base address registers are unused.	r	r	0

11.4.15 (Address 28h; CARDBUS); PCI CardBus CIS Pointer

Bits	Description	PCI	USB/8051	Default
31:0	**PCI Cardbus CIS Pointer**. This register provides the Card Information Structure Pointer for the CardBus card. This feature is not supported in the NET2280.	r	r	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4.16 (Address 2Ch; PCISUBVID); PCI Subsystem Vendor ID

Bits	Description	PCI	USB/8051	Default
15:0	**PCI Subsystem Vendor ID.** This field allows board vendors to include their own distinct PCI vendor ID.	r	rw	0

11.4.17 (Address 2Eh; PCISUBID); PCI Subsystem ID

Bits	Description	PCI	USB/8051	Default
15:0	**PCI Subsystem ID.** This field allows board vendors to include their own distinct product ID.	r	rw	0

11.4.18 (Address 30h; PCIROMBASE) PCI Expansion ROM Base Address

Bits	Description	PCI	USB/8051	Default
31:0	**PCI ROM Base Address.** This base address register is unused.	r	r	0

11.4.19 (Address 34h; PCICAPPTR); PCI Capabilities Pointer

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Capabilities Pointer.** This field provides the configuration address of the first New Capabilities register.	r	rw	‘h40

11.4.20 (Address 3Ch; PCIINTLINE); PCI Interrupt Line

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Interrupt Line.** This field indicates which input of the system interrupt controller the device’s interrupt pin is connected to. Device drivers and operating systems use this field.	rw	rw	0

11.4.21 (Address 3Dh; PCIINTPIN); PCI Interrupt Pin

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Interrupt Pin.** This field indicates which PCI interrupt pin this device is connected to. A value of 1 indicates INTA#.	r	rw	1

11.4.22 (Address 3Eh; PCIMINGNT); PCI Minimum Grant

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Minimum Grant.** This field specifies the desired maximum burst period in units of 250 nsec.	r	rw	0

11.4.23 (Address 3Fh; PCIMINLAT); PCI Minimum Latency

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Minimum Latency**. This field specifies how often the device needs to gain access to the PCI bus in units of 250 nsec.	r	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4.24 (Address 40h; PWRMNGID); Power Management Capability ID

Bits	Description	PCI	USB/8051	Default
7:0	**Power Management Capability ID.** This register specifies the Power Management capability ID.	r	r	1

11.4.25 (Address 41h; PWRMNGNEXT); Power Management Next Pointer

Bits	Description	PCI	USB/8051	Default
7:0	**PCI Power Management Next Pointer.** This register points to the first location of the next item in the capabilities linked list. The NET2280 doesn't support any additional capabilities, so this field is forced to 0.	r	r	0

11.4.26 (Address 42h; PWRMNGCAP); Power Management Capabilities

Bits	Description	PCI	USB/8051	Default
15:11	**PME Support.** This field indicates the power states in which the device may assert the PME# signal. Value Description XXXX1 Can be asserted from D0 XXX1X Can be asserted from D1 XX1XX Can be asserted from D2 X1XXX Can be asserted from $D3_{hot}$ 1XXXX Can be asserted from $D3_{cold}$	r	rw	0Bh
10	**D2 Support.** This bit specifies that the device supports the D2 state.	r	rw	0
9	**D1 Support.** This bit specifies that the device supports the D1 state.	r	rw	1
8:6	***Reserved***	r	r	0
5	**Device Specific Initialization.** This bit indicates that the device requires special initialization following a transition to the D0 uninitialized state before the generic class device driver can use it.	r	rw	0
4	**Reserved.**	r	r	0
3	**PME Clock.** When low, it indicates that no PCI clock is required to generate PME#. When high, it indicates that a PCI clock is required to generate PME#.	r	rw	0
2:0	**PME Version.** This field specifies the revision of the *PCI Power Management Interface Specification* to which this device complies	r	r	2

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.4.27 (Address 44h; PWRMNGCSR); Power Management Control/Status

Bits	Description	PCI	USB/8051	Default
15	**PME Status.** This bit indicates that the PME# pin is being driven if *PME Enable* bit is set high. A USB suspend or resume causes this bit to be set, and writing a 1 from the PCI bus clears the bit.	rcu	rcu	0
14:13	**Data Scale.** This field is not supported, and always returns a value of 0.	r	r	0
12:9	**Data Select.** This field is not supported, and always returns a value of 0	r	r	0
8	**PME Enable.** This bit enables the PME# pin to be asserted.	rw	rw	0
7:2	**Reserved**	r	r	0
1:0	**Power State.** This field is used to determine or change the current power state. Value State 00 **D0** 01 **D1** 10 **D2** 11 **$D3_{hot}$** A transition from state **D3** to state **D0** will cause a soft reset to occur. In states **D1** and **D2**, if the corresponding *D1 Support* and *D2Support* bits are set, PCI memory and I/O accesses are disabled, as well as the PCI interrupt, and only configuration cycles are allowed. In state **$D3_{hot}$**, these functions are also disabled, and the *USB Detect Enable* bit is cleared, effectively disconnecting the NET2280 from the USB host.	rw	rw	0

11.4.28 (Address 46h; PWRMNGBRIDGE); Power Management Bridge Support

Bits	Description	PCI	USB/8051	Default
7:0	**Power Management Bridge Support.** This function is not supported, and always returns a value of 0.	r	r	0

11.4.29 (Address 47h; PWRMNGDATA) Power Management Data

Bits	Description	PCI	USB/8051	Default
7:0	**Power Management Data.** This function is not supported, and always returns a value of 0.	r	r	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5 Main Control Registers

11.5.1 (Address 00h; DEVINIT); Device Initialization

Bits	Description	Access	Default
31:12	**Reserved**	r	0
11:8	**Local Clock Frequency.** This field controls the frequency of the LCLKO pin. When set to 0, the clock is stopped. Non-zero values represent divisors of the 60 MHz clock. The default value is 8, representing a frequency of 7.5 MHz.	rw	8
7	**Force PCI RST.** When clear, the PCI RST# pin is driven when the RESET# pin is asserted, the *PCI Soft Reset* pin is set, or a power management soft reset occurs. When set, the PCI RST# pin is driven continuously. This bit is only valid when the NET2280 is in PCI Host mode.	rw	0
6	**PCI ID.** When clear, the normal PCI Device and Vendor ID are returned to the PCI host when the **PCIVENDID** and **PCIDEVID** configuration registers are accessed. When set, the subsystem device and vendor ID values are returned instead.	rw	0
5	**PCI Enable.** When clear, all PCI accesses to the NET2280 will result in a target retry response. When set, the NET2280 will respond normally to PCI accesses. If no valid EEPROM is detected and PCI Adapter mode is selected, this bit is automatically set.	rwu	0
4	**FIFO Soft Reset.** When set, all of the endpoint FIFOs are flushed. Reading this bit always returns a 0.	su	0
3	**CFG Soft Reset.** When set, all of the configuration registers in the device are reset. Reading this bit always returns a 0.	su	0
2	**PCI Soft Reset.** When set, the PCI configuration registers and PCI control section of the device are reset. If in PCI host mode, the PCI RST# pin is driven for 2 to 3 msec. Reading this bit returns the value of the PCI RST# output.	rsu	0
1	**USB Soft Reset.** When set, the USB control section of the device is reset. Also, the **OURADDR** and **OURCONFIG** registers are cleared. Reading this bit always returns a 0.	su	0
0	**8051 Reset.** When clear, the 8051 is enabled to execute firmware. When set, the 8051 is held in a reset state. If a valid EEPROM with 8051 firmware is detected, this bit is automatically cleared when the EEPROM read has completed. If no valid EEPROM is detected, this bit is not cleared.	rwu	1

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.2 (Address 04h; EECTL); EEPROM Control

Bits	Description	Access	Default
31:25	**Reserved.**	r	0
24:23	**EEPROM Address Width.** This field reports the addressing width of the installed EEPROM. If the addressing width can't be determined, a zero is returned. Value Address Width 00 undetermined 01 1 byte 10 2 byte 11 3 byte	r	---
22	**EEPROM Chip Select Active.** This bit is set if the chip select pin to the EEPROM is active. The chip select can be active across multiple byte operations.	r	---
21	**EEPROM Present.** This bit is set if the EEPROM controller determines that an EEPROM is connected to the NET2280.	r	---
20	**EEPROM Valid.** A non-blank EEPROM with 'h5A in the first byte has been detected.	r	---
19	**EEPROM Busy.** When set, the EEPROM controller is busy performing a byte read or write operation. An interrupt can be generated whenever this bit goes false.	r	0
18	**EEPROM Chip Select Enable.** When set, the EEPROM chip select is enabled.	rw	0
17	**EEPROM Byte Read Start.** When set, a byte is read from the EEPROM, and can be accessed using the *EEPROM Read Data* field. This bit is automatically cleared when the read operation is complete.	rwu	0
16	**EEPROM Byte Write Start.** When set, the value in the *EEPROM Write Data* field is written to the EEPROM. This bit is automatically cleared when the write operation is complete.	rwu	0
15:8	**EEPROM Read Data.** This field determines the byte read from the EEPROM when the *EEPROM Byte Read Start* bit is set.	r	--
7:0	**EEPROM Write Data.** This field determines the byte written to the EEPROM when the *EEPROM Byte Write Start* bit is set. This field can represent an opcode, address, or data being written to the EEPROM.	rw	0

11.5.3 (Address 08h; EECLKFREQ); EEPROM Clock Frequency

Bits	Description	Access	Default
31:2	**Reserved.**	r	---
1:0	**EEPROM Clock Frequency.** This field controls the frequency of the EECLK pin. Value Frequency 00 1.875 MHz 01 3.75 MHz 10 7.5 MHz 11 15 MHz	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.4 (Address 10h; PCIIRQENB0) PCI Interrupt Request Enable 0

Bits	Description	Access	Default
31:8	**Reserved.**	r	0
7	**Setup Packet Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when a setup packet has been received from the host.	rw	0
6	**Endpoint F Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0
5	**Endpoint E Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0
4	**Endpoint D Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0
3	**Endpoint C Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0
2	**Endpoint B Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0
1	**Endpoint A Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0
0	**Endpoint 0 Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active on this endpoint.	rw	0

11.5.5 (Address 14h; PCIIRQENB1) PCI Interrupt Request Enable 1

Bits	Description	Access	Default
31	**PCI Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated.	rw	1
30:28	**Reserved.**	r	0
27	**Power State Change Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the *Power State* field of the **PWRMNGCSR** register is changed.	rw	0
26	**PCI Arbiter Timeout Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the PCI arbiter times out waiting for a transaction to start after GNT# is asserted.	rw	0
25	**PCI Parity Error Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the NET2280 has detected a parity error during a PCI master or slave transaction.	rw	0
24:21	**Reserved.**	r	0
20	**PCI Master Abort Received Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the NET2280 has detected a PCI Master Abort (no DEVSEL# from target) during a master cycle.	rw	0
19	**PCI Target Abort Received Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the NET2280 has detected a Target Abort (no DEVSEL# with STOP#) during a master cycle.	rw	0
18	**Reserved.**	rw	0
17	**PCI Retry Abort Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the NET2280 has received 256 consecutive retries from a target, and the *PCI Retry Abort* bit in the **PCIMSTCTL** register is set.	rw	0
16	**PCI Master Cycle Done Interrupt Enable**. When set, this bit enables a PCI interrupt to be generated when an 8051 or USB initiated PCI master transaction completes.	rw	0
15:14	**Reserved.**	r	0
13	**GPIO Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active from one of the GPIO pins.	rw	0
12	**DMA D Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
11	**DMA C Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active from this DMA channel.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

10	**DMA B Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
9	**DMA A Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
8	**EEPROM Done Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an EEPROM read or write transaction completes.	rw	0
7	**VBUS Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when a change has been detected on the VBUS pin.	rw	0
6	**Control Status Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when an IN or OUT token indicating Control Status has been received.	rw	0
5	**Reserved.**	rw	0
4	**Root Port Reset Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when a root port reset is detected.	rw	0
3	**Suspend Request Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when a USB Suspend Request from the host is detected.	rw	0
2	**Suspend Request Change Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when a change in the Suspend Request Interrupt state is detected.	rw	0
1	**Resume Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when the device has resumed from the suspended state.	rw	0
0	**SOF Interrupt Enable.** When set, this bit enables a PCI interrupt to be generated when a start-of-frame packet is received by the NET2280.	rw	0

11.5.6 (Address 18h; CPUIRQENB0) CPU Interrupt Request Enable 0

Bits	Description	Access	Default
31:8	**Reserved.**	r	0
7	**Setup Packet Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when a setup packet has been received from the host.	rw	0
6	**Endpoint F Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0
5	**Endpoint E Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0
4	**Endpoint D Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0
3	**Endpoint C Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0
2	**Endpoint B Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0
1	**Endpoint A Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0
0	**Endpoint 0 Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active on this endpoint.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.7 (Address 1Ch; CPUIRQENB1) CPU Interrupt Request Enable 1

Bits	Description	Access	Default
31	**CPU Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated.	rw	0
30:28	**Reserved.**	r	0
27	**Power State Change Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the *Power State* field of the **PWRMNGCSR** register is changed.	rw	0
26	**PCI Arbiter Timeout Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the PCI arbiter times out waiting for a transaction to start after GNT# is asserted.	rw	0
25	**PCI Parity Error Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the NET2280 has detected a parity error during a PCI master or slave transaction.	rw	0
24	**PCI INTA# Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the PCI INTA# input is asserted and PCI host mode is selected.	rw	0
23	**PCI PME# Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the PCI PME# input is asserted and PCI host mode is selected.	rw	0
22	**PCI SERR# Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the PCI SERR# input is asserted and PCI host mode is selected.	rw	0
21	**PCI PERR# Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the PCI PERR# input is asserted and PCI host mode is selected.	rw	0
20	**PCI Master Abort Received Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the NET2280 has detected a PCI Master Abort (no DEVSEL# from target) during a master cycle.	rw	0
19	**PCI Target Abort Received Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the NET2280 has detected a Target Abort (no DEVSEL# with STOP#) during a master cycle.	rw	0
18	**Reserved.**	rw	0
17	**PCI Retry Abort Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the NET2280 has received 256 consecutive retries from a target, and the *PCI Retry Abort* bit in the **PCIMSTCTL** register is set.	rw	0
16	**PCI Master Cycle Done Interrupt Enable**. When set, this bit enables an 8051 CPU interrupt to be generated when an 8051 or USB initiated PCI master transaction completes.	rw	0
15:14	**Reserved.**	r	0
13	**GPIO Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active from one of the GPIO pins.	rw	0
12	**DMA D Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
11	**DMA C Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
10	**DMA B Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
9	**DMA A Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
8	**EEPROM Done Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an EEPROM read or write transaction completes.	rw	0
7	**VBUS Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when a change has been detected on the VBUS pin.	rw	0
6	**Control Status Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when an IN or OUT token indicating Control Status has been received.	rw	0
5	**Reserved.**	rw	0
4	**Root Port Reset Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when a root port reset is detected.	rw	0
3	**Suspend Request Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when a USB Suspend Request from the host is detected.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

2	**Suspend Request Change Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when a change in the Suspend Request Interrupt state is detected.	rw	0
1	**Resume Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when the device has resumed from the suspended state.	rw	0
0	**SOF Interrupt Enable.** When set, this bit enables an 8051 CPU interrupt to be generated when a start-of-frame packet is received by the NET2280.	rw	0

11.5.8 (Address 24h; USBIRQENB1) USB Interrupt Request Enable 1

Bits	Description	Access	Default
31	**USB Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated.	rw	0
30:28	**Reserved.**	r	0
27	**Power State Change Interrupt Enable.** When set, this bit enables STATIN endpoint interrupt to be generated when the *Power State* field of the **PWRMNGCSR** register is changed.	rw	0
26	**PCI Arbiter Timeout Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the PCI arbiter times out waiting for a transaction to start after GNT# is asserted.	rw	0
25	**PCI Parity Error Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the NET2280 has detected a parity error during a PCI master or slave transaction.	rw	0
24	**PCI INTA# Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the PCI INTA# input is asserted and PCI host mode is selected.	rw	0
23	**PCI PME# Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the PCI PME# input is asserted and PCI host mode is selected.	rw	0
22	**PCI SERR# Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the PCI SERR# input is asserted and PCI host mode is selected.	rw	0
21	**PCI PERR# Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the PCI PERR# input is asserted and PCI host mode is selected.	rw	0
20	**PCI Master Abort Received Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the NET2280 has detected a PCI Master Abort (no DEVSEL# from target) during a master cycle.	rw	0
19	**PCI Target Abort Received Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the NET2280 has detected a Target Abort (no DEVSEL# with STOP#) during a master cycle.	rw	0
18	**Reserved**	rw	0
17	**PCI Retry Abort Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the NET2280 has received 256 consecutive retries from a target, and the *PCI Retry Abort* bit in the **PCIMSTCTL** register is set.	rw	0
16	**PCI Master Cycle Done Interrupt Enable**. When set, this bit enables a STATIN endpoint interrupt to be generated when an 8051 or USB initiated PCI master transaction completes.	rw	0
15:14	**Reserved.**	r	0
13	**GPIO Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an interrupt is active from one of the GPIO pins.	rw	0
12	**DMA D Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
11	**DMA C Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
10	**DMA B Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
9	**DMA A Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an interrupt is active from this DMA channel.	rw	0
8	**EEPROM Done Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an EEPROM read or write transaction completes.	rw	0
7	**VBUS Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when a change has been detected on the VBUS pin.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

6	**Control Status Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when an IN or OUT token indicating Control Status has been received.	rw	0
5	**Reserved.**	rw	0
4	**Root Port Reset Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when a root port reset is detected.	rw	0
3	**Suspend Request Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when a USB Suspend Request from the host is detected.	rw	0
2	**Suspend Request Change Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when a change in the Suspend Request Interrupt state is detected.	rw	0
1	**Resume Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when the device has resumed from the suspended state.	rw	0
0	**SOF Interrupt Enable.** When set, this bit enables a STATIN endpoint interrupt to be generated when a start-of-frame packet is received by the NET2280.	rw	0

11.5.9 (Address 28h; IRQSTAT0) Interrupt Request Status 0

Bits	Description	Access	Default
31:13	**Reserved.**	r	0
12	**INTA# Asserted.** This bit is high when the PCI INTA# pin is being driven.	r	0
11:8	**Reserved.**	r	0
7	**Setup Packet Interrupt.** This bit is set when a setup packet has been received from the host. Writing a 1 clears this bit.	rcu	0
6	**Endpoint F Interrupt.** This bit conveys the interrupt status for Endpoint F. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
5	**Endpoint E Interrupt.** This bit conveys the interrupt status for Endpoint E. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
4	**Endpoint D Interrupt.** This bit conveys the interrupt status for Endpoint D. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
3	**Endpoint C Interrupt.** This bit conveys the interrupt status for Endpoint C. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
2	**Endpoint B Interrupt.** This bit conveys the interrupt status for Endpoint B. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
1	**Endpoint A Interrupt.** This bit conveys the interrupt status for Endpoint A. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
0	**Endpoint 0 Interrupt.** This bit conveys the interrupt status for Endpoint 0. When set, the endpoint's interrupt status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0

11.5.10 (Address 2Ch; IRQSTAT1) Interrupt Request Status 1

Bits	Description	Access	Default
31:28	**Reserved.**	r	0
27	**Power State Change Interrupt.** This bit is set when the *Power State* field of the **PWRMNGCSR** register is changed.	rcu	0
26	**PCI Arbiter Timeout Interrupt.** This bit is set when the PCI arbiter times out waiting for a transaction to start after GNT# is asserted. Writing a 1 clears this status bit.	rcu	0
25	**PCI Parity Error Interrupt.** This bit is set when the NET2280 has detected a parity error during a master or slave transaction. Writing a 1 clears this status bit.	rcu	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

24	**PCI INTA# Interrupt.** This bit is set when the PCI INTA# input is asserted and PCI host mode is selected. Writing a 1 clears this status bit.	rcu	0
23	**PCI PME# Interrupt.** This bit is set when the PCI PME# input is asserted and PCI host mode is selected. Writing a 1 clears this status bit.	rcu	0
22	**PCI SERR# Interrupt.** This bit is set when the PCI SERR# input is asserted and PCI host mode is selected. Writing a 1 clears this status bit.	rcu	0
21	**PCI PERR# Interrupt.** This bit is set when the PCI PERR# input is asserted and PCI host mode is selected. Writing a 1 clears this status bit.	rcu	0
20	**PCI Master Abort Received Interrupt.** This bit is set when the NET2280 has detected a PCI Master Abort (no DEVSEL# from target) during a master cycle. Writing a 1 clears this status bit.	rcu	0
19	**PCI Target Abort Received Interrupt.** This bit is set when the NET2280 has detected a Target Abort (no DEVSEL# with STOP#) during a master cycle. Writing a 1 clears this status bit.	rcu	0
18	**Reserved**	r	0
17	**PCI Retry Abort Interrupt.** This bit is set when the NET2280 has received 256 consecutive retries from a target, and the *PCI Retry Abort* bit in the **PCIMSTCTL** register is set. Writing a 1 clears this status bit.	rcu	0
16	**PCI Master Cycle Done Interrupt**. This bit is set when an 8051 or USB initiated PCI master transaction completes. Writing a 1 clears this status bit.	rcu	0
15:14	**Reserved.**	r	0
13	**GPIO Interrupt.** This bit conveys the interrupt status for the four GPIO pins. When set, the GPIO Status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
12	**DMA D Interrupt.** This bit conveys the interrupt status for DMA Channel D. When set, DMA Channel D Status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
11	**DMA C Interrupt.** This bit conveys the interrupt status for DMA Channel C. When set, DMA Channel C Status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
10	**DMA B Interrupt.** This bit conveys the interrupt status for DMA Channel B. When set, DMA Channel B Status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
9	**DMA A Interrupt.** This bit conveys the interrupt status for DMA Channel A. When set, DMA Channel A Status register should be read to determine the cause of the interrupt. This bit is set independently of the interrupt enable bit.	ru	0
8	**EEPROM Done Interrupt.** This bit is set when an EEPROM read or write transaction completes. Writing a 1 clears this status bit.	rcu	0
7	**VBUS Interrupt.** When set, this bit indicates that a change occurred on the VBUS input pin. Read the **USBCTL** register for the current state of this pin. Writing a 1 clears this status bit.	rcu	-
6	**Control Status Interrupt.** This bit is set when an IN or OUT token indicating Control Status has been received. Writing a 1 clears this status bit.	rcu	0
5	**Reserved.**	r	0
4	**Root Port Reset Interrupt.** This bit indicates a change in state of the root port reset detector. Writing a 1 clears this status bit.	rcu	---
3	**Suspend Request Interrupt**. This bit is set when the NET2280 detects a USB Suspend request from the host. The Suspend Request state cannot be set or cleared by writing this bit. Instead, writing a 1 to this bit puts the NET2280 into the low-power suspend mode (see section 10.1).	rsu	0
2	**Suspend Request Change Interrupt.** This bit is set whenever there is a change in the Suspend Request Interrupt state (bit 3). Writing a 1 clears this status bit.	rcu	0
1	**Resume Interrupt.** When set, this bit indicates that the device has resumed from the suspended state. Writing a 1 clears this status bit.	rcu	0
0	**SOF Interrupt.** This bit indicates when a start-of-frame packet has been received by the NET2280. Writing a 1 clears this status bit.	rcu	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.11 (Address 30h; IDXADDR) Indexed Register Address

Bits	Description	Access	Default
31:16	**Reserved.**	r	0
15:0	**Indexed Register Address.** This register selects which indexed register is accessed when the **IDXDATA** port is read or written.	rw	0

11.5.12 (Address 34h; IDXDATA) Indexed Register Data

Bits	Description	Access	Default
31:0	**Indexed Register Data.** This port provides access to the indexed data register selected by **IDXADDR**.	rw	0

11.5.13 (Address 38h; FIFOCTL) FIFO Control

Bits	Description	Access	Default
31:16	**PCI Base2 Range.** This field determines the range of the **PCI Base Address 2** register, in increments of 64 Kbytes. The default corresponds to a range of 64 K. Starting with bit 16, as each successive bit is changed to a 0, the range doubles. A value of 0 corresponds to a range of 4 Gbytes, and thus causes the **PCI Base Address 2** to be disabled. The **PCI Base Address 2** register must be a multiple of the range.	rw	FFFFh
15:4	**Reserved.**	r	0
3	**Ignore FIFO Availability.** When clear, PCI accesses to empty/full FIFOs result in a retry termination. When set, PCI accesses to empty/full FIFOs result in a normal termination, but read data is undefined and write data is ignored. For PCI writes, the *FIFO Overflow* bit is set and for PCI reads, the *FIFO Underflow* bit is set.	rw	1
2	**PCI Base2 Select**. When clear, endpoint FIFOs A, B, C, and D are each assigned one quarter of the PCI Base 2 space (A = 1st, B = 2nd, C = 3rd, D = 4th). When set, PCI writes to the lower half of the PCI base2 space are directed to the endpoint A FIFO, reads from the lower half are directed to the endpoint B FIFO, writes to the upper half are directed to the endpoint C FIFO, and reads from the upper half are directed to the endpoint D FIFO.	rw	0
1:0	**FIFO Configuration Select**. This field selects the FIFO configuration for endpoints A, B, C, and D. Select — Configuration 0 — Endpoints A, B, C, D each have a 1 Kbyte FIFO 1 — Endpoints A and B each have a 2 Kbyte FIFO. Endpoints C and D are unavailable. 2 — Endpoint A has a 2 Kbyte FIFO, and endpoints B and C each have a 1 Kbyte FIFO. Endpoint D is unavailable. 3 — unused	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.14 (Address 40h; MEMADDR) FIFO Memory Address

Bits	Description	Access	Default
31:29	**Reserved.**	r	0
28	**Start.** Writing a 1 to this bit initiates a diagnostic read or write to FIFO memory. When the access has completed, this bit is automatically cleared.	rsu	0
27	**Direction.** This field determines the direction of the diagnostic access. When clear, data is read from the FIFO memory. When set, data is written to FIFO memory	rw	0
26:25	**Reserved.**	r	0
24	**FIFO Diagnostic Select.** When low, the FIFO operates normally. When high, the FIFO is in diagnostic mode. In the diagnostic mode, the FIFO memory is accessed using the address specified in bits 10:0 of this register. Data is written and read through the **MEMDATA** registers.	rw	0
23:11	**Reserved.**	r	0
10:0	**Memory Address.** This field selects the address in FIFO memories where reads or writes through the **MEMDATA** registers occur. When bit 10 of this field is low, the 4K byte FIFO for endpoints A, B, C, and D is accessed. When bit 10 of this field is high, the 256 byte FIFO for endpoints 0, E, and F is accessed.	rw	0

11.5.15 (Address 44h; MEMDATA0) FIFO Memory Data 0

Bits	Description	Access	Default
31:0	**FIFO Memory Data 0.** This register is used when the FIFO memory diagnostic mode is enabled. During reads, the least significant 32 bits (31:0) of the word addressed by the **MEMADDR** register are returned. During writes, this register provides the least significant 32 bits of data being written to memory.	rw	---

11.5.16 (Address 48h; MEMDATA1) FIFO Memory Data 1

Bits	Description	Access	Default
31:5	**Reserved.**	r	0
4:0	**FIFO Memory Data 1.** This register is used when the FIFO memory diagnostic mode is enabled. During reads, the most significant 5 bits (36:32) of the word addressed by the **MEMADDR** register are returned. During writes, this register provides the most significant 5 bits of data being written to memory.	rw	---

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.5.17 (Address 50h; GPIOCTL) General Purpose I/O Control

Bits	Description	Access	Default
31:13	**Reserved.**	r	0
12	**GPIO3 LED Select.** When set, the GPIO3 pin is driven high during USB activity. The *GPIO3 Output Enable* bit must be set for this feature to be available.	rw	0
11	**GPIO3 Interrupt Enable.** When set, changes on the GPIO3 pin (when programmed as an input), are enabled to generate an interrupt.	rw	0
10	**GPIO2 Interrupt Enable.** When set, changes on the GPIO2 pin (when programmed as an input), are enabled to generate an interrupt.	rw	0
9	**GPIO1 Interrupt Enable.** When set, changes on the GPIO1 pin (when programmed as an input), are enabled to generate an interrupt.	rw	0
8	**GPIO0 Interrupt Enable.** When set, changes on the GPIO0 pin (when programmed as an input), are enabled to generate an interrupt.	rw	0
7	**GPIO3 Output Enable.** When clear, the GPIO3 pin is an input. When set, the GPIO3 pin is an output.	rw	0
6	**GPIO2 Output Enable.** When clear, the GPIO2 pin is an input. When set, the GPIO2 pin is an output.	rw	0
5	**GPIO1 Output Enable.** When clear, the GPIO1 pin is an input. When set, the GPIO1 pin is an output.	rw	1
4	**GPIO0 Output Enable.** When clear, the GPIO0 pin is an input. When set, the GPIO0 pin is an output.	rw	1
3	**GPIO3 Data.** When programmed as an output, values written to this bit appear on the GPIO3 pin. Reading this bit returns the value that was previously written. When programmed as an input, reading this bit returns the value present on the GPIO3 pin.	rw	0
2	**GPIO2 Data.** When programmed as an output, values written to this bit appear on the GPIO2 pin. Reading this bit returns the value that was previously written. When programmed as an input, reading this bit returns the value present on the GPIO2 pin.	rw	0
1	**GPIO1 Data.** When programmed as an output, values written to this bit appear on the GPIO1 pin. Reading this bit returns the value that was previously written. When programmed as an input, reading this bit returns the value present on the GPIO1 pin.	rw	0
0	**GPIO0 Data.** When programmed as an output, values written to this bit appear on the GPIO0 pin. Reading this bit returns the value that was previously written. When programmed as an input, reading this bit returns the value present on the GPIO0 pin.	rw	0

11.5.18 (Address 54h; GPIOSTAT) General Purpose I/O Status

Bits	Description	Access	Default
31:4	**Reserved.**	r	0
3	**GPIO3 Interrupt.** This bit is set when the state of the GPIO3 pin changes and the pin is programmed as an input. Writing a 1 clears this bit.	rcu	0
2	**GPIO2 Interrupt.** This bit is set when the state of the GPIO2 pin changes and the pin is programmed as an input. Writing a 1 clears this bit.	rcu	0
1	**GPIO1 Interrupt.** This bit is set when the state of the GPIO1 pin changes and the pin is programmed as an input. Writing a 1 clears this bit.	rcu	0
0	**GPIO0 Interrupt.** This bit is set when the state of the GPIO0 pin changes and the pin is programmed as an input. Writing a 1 clears this bit.	rcu	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.6 USB Control Registers

11.6.1 (Address 80h; STDRSP) Standard Response Control

Bits	Description	Access	Default
31	**Stall Unsupported Requests.** When clear, unsupported Standard Requests are passed to the firmware. When set, unsupported Standard Requests cause a STALL handshake and endpoint 0 is halted.	rw	1
30:17	**Reserved**	r	0
16	**Set Test Mode.** When clear, a Set Test Mode request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
15	**Get Other Speed Configuration.** When clear, a Get Other Speed Configuration Descriptor request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
14	**Get Device Qualifier.** When clear, a Get Device Qualifier request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
13	**Set Address.** When clear, a Set Address request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
12	**Endpoint Set/Clear Halt.** When clear, Endpoint Set/Clear Feature requests for controlling the Halt condition are passed to the 8051 CPU through the Setup Registers. When set, the requests are handled automatically without notifying the 8051 CPU.	rw	1
11	**Device Set/Clear Device Remote Wake-up.** When clear, Device Set/Clear Feature requests for controlling Device Remote Wake-up Enable are passed to the 8051 CPU through the Setup Registers. When set, the requests are handled automatically without notifying the 8051 CPU.	rw	1
10	**Get String Descriptor 2.** When clear, a Get String Descriptor 2 (Product ID) request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
9	**Get String Descriptor 1.** When clear, a Get String Descriptor 1 (Manufacturer ID) request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
8	**Get String Descriptor 0.** When clear, a Get String Descriptor 0 (Language ID) request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
7	**Reserved**	rw	0
6	**Get/Set Interface.** When clear, Get/Set Interface requests are passed to the 8051 CPU through the Setup Registers. When set, the requests are handled automatically without notifying the 8051 CPU.	rw	1
5	**Get/Set Configuration.** When clear, Get/Set Configuration requests are passed to the 8051 CPU through the Setup Registers. When set, the requests are handled automatically without notifying the 8051 CPU.	rw	1
4	**Get Configuration Descriptor.** When clear, a Get Configuration Descriptor request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU. Other configuration registers in the NET2280 determine the values in the descriptor.	rw	1
3	**Get Device Descriptor.** When clear, a Get Device Descriptor request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU. Other configuration registers in the NET2280 determine the values in the descriptor.	rw	1

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Bits	Description	Access	Default
2	**Get Endpoint Status.** When clear, a Get Endpoint Status request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
1	**Get Interface Status.** When clear, a Get Interface Status request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1
0	**Get Device Status.** When clear, a Get Device Status request is passed to the 8051 CPU through the Setup Registers. When set, the request is handled automatically without notifying the 8051 CPU.	rw	1

11.6.2 (Address 84h; PRODVENDID) Product/Vendor ID

Bits	Description	Access	Default
31:16	**Product ID.** This field determines the Product ID during a 'Get Device Descriptor' request. This register is only used when the Get Device Descriptor request is in Auto-Enumerate mode.	rw	2280h
15:0	**Vendor ID.** This field determines the Vendor ID during a 'Get Device Descriptor' request. This register is only used when the Get Device Descriptor request is in Auto-Enumerate mode.	rw	0525h

11.6.3 (Address 88h; RELNUM) Device Release Number

Bits	Description	Access	Default
31:16	**Reserved**	r	0
15:0	**Device Release Number.** This field determines the device release number during a 'Get Device Descriptor' request. This register is only used when the Get Device Descriptor request is in Auto-Enumerate mode.	rw	REL NUM

Note: RELNUM is the silicon revision, encoded as a 4-digit BCD value. The value of RELNUM for the first release of the chip is 0100h. The least-significant two digits are incremented for mask changes, and the most-significant two digits are incremented for major revisions. This value can be changed by the 8051 CPU to implement an application release number.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.6.4 (Address 8Ch; USBCTL) USB Control

Bits	Description	Access	Default
31:24	**Reserved**	r	0
23:16	**Serial Number Index.** This field determines the Serial Number Index returned to the host during an Auto-Enumerate Get Device Descriptor request.	rw	0
15:14	**Reserved**	r	0
13	**Product ID String Enable.** When clear, the product string index value in the Device Descriptor is set to zero, and the string descriptor read is acknowledged with a stall in Auto-Enumerate mode. When set, this bit allows the default Product string descriptor to be returned to the host in Auto-Enumerate mode.	rw	1
12	**Vendor ID String Enable.** When clear, the vendor string index value in the Device Descriptor is set to zero, and the string descriptor read is acknowledged with a stall in Auto-Enumerate mode. When set, this bit allows the default Vendor string descriptor to be returned to the host in Auto-Enumerate mode.	rw	1
11	**USB Root Port Wakeup Enable.** When clear, the wake-up condition is not detected. When set, the root port wake-up condition is detected when activity is detected on the USB.	rw	1
10	**VBUS pin.** This bit indicates the state of the VBUS pin. When set, this bit indicates that the NET2280 is connected to the USB.	r	---
9	**Timed Disconnect.** When this bit is written, the *USB Detect Enable* bit is disabled after a 20 ms delay, effectively disconnecting the 2280 from the USB host. After a 1 second delay, the *USB Detect Enable* bit is set again, re-connecting the 2280 to the USB host.	rsu	0
8	**Reserved**	r	0
7	**Suspend Immediately.** When clear, the *Suspend Request Interrupt* bit in the **IRQSTAT1** register must be written to initiate the suspend sequence. When set, this bit allows the NET2280 to enter the suspend state automatically if the USB is idle for 3 milliseconds. This bit is automatically set if PCI Host mode is selected and no valid EEPROM is detected at reset time.	rwu	0
6	**Self-Powered USB Device.** This bit is reported to the USB host in the USB configuration descriptor's "bmAttributes" field, bit 6, during Auto-Enumerate, and indicates if the device is self-powered.	rw	1
5	**Remote Wakeup Support.** This bit is reported to the USB host in the USB configuration descriptor's "bmAttributes" field, bit 5, during Auto-Enumerate, and indicates whether the NET2280 supports device remote wakeup.	rw	0
4	**PME Polarity.** When clear, the PME pin is active low. In this mode, the output driver is configured as open-drain. When set, the PME pin is active high. In this mode, the output driver is configured as totem-pole.	rw	0
3	**USB Detect Enable.** When clear, the NET2280 does not appear to be connected to the USB host. When set, the NET2280 appears to be connected to the USB host. This bit should not be set until the configuration registers have been programmed. This bit is automatically set if PCI Host mode is selected and no valid EEPROM is detected at reset time.	rwu	0
2	**PME Wakeup Enable.** When clear, asserting PME# will not cause a device remote wakeup. When set, this bit enables the assertion of PME# to initiate a device remote wakeup. This bit is only valid if the NET2280 is in PCI host mode.	rw	0
1	**Device Remote Wake-up Enable**. When set, this bit enables the device remote wake-up feature of the NET2280. This bit is automatically controlled by the Device Remote Wake-up Set/Clr Standard Request, and is reported to the USB host in the Get Device Status, bit 1.	rwu	0
0	**Self Powered Status.** This bit determines the value of the self-powered bit in the Device Status Request when operating in the Auto-enumerate mode.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.6.5 (Address 90h; USBSTAT) USB Status

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7	**High Speed.** When set, the NET2280 is operating in high-speed mode (480 Mbps).	r	0
6	**Full Speed.** When set, the NET2280 is operating in full-speed mode (12 Mbps).	r	0
5	**Generate Resume.** Writing a 1 to this bit causes a Resume sequence to be initiated to the host if device remote wakeup is enabled. This bit should be written by the 8051 after a device remote wakeup has been generated (PME# pin asserted). This bit is self-clearing, and always returns a 0.	su	0
4	**Generate Device Remote Wakeup.** Writing a 1 to this bit causes the USB section of the NET2280 to wake up from a suspended state, and is intended to be used in conjunction with the *Suspend Immediately* bit in the **USBCTL** register. This bit is self-clearing, and always returns a 0.	su	0
3:0	**Reserved**	r	0

11.6.6 (Address 94h; XCVRDIAG) Transceiver Diagnostic Control

Bits	Description	Access	Default
31	**Force High Speed Mode.** When set, the NET2280 is forced into high-speed mode.	rw	0
30	**Force Full Speed Mode.** When set, the NET2280 is forced into full-speed mode.	rw	0
29:27	**Reserved**	r	0
26:24	**USB Test Mode.** Only valid in high-speed mode. **Value** **Mode** 000 Normal 001 Test 'J' 010 Test 'K' 011 Test SE0_NAK 100 Test Packet 101 Test Force Enable 110 Reserved 111 Reserved	rw	0
23:18	**Reserved**	r	0
17:16	**Line State** **Value** **State** 00 SE0 01 'J' 10 'K' 11 SE1	r	---
15:4	**Reserved.**	r	0
3:2	**Transceiver Operation Mode** **Value** **OPMODE** 00 Normal 01 Non-Driving 10 Disable bit stuffing and NRZI encoding 11 Reserved	r	---
1	**Transceiver Select.** When clear, the high-speed transceiver is enabled. When set, the full-speed transceiver is enabled.	r	---
0	**Termination Select.** When clear, the high-speed termination is enabled. When set, the full-speed termination is enabled.	r	---

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.6.7 (Address 98h; SETUP0123) Setup Bytes 0, 1, 2, 3

Bits	Description	Access	Default
31:24	**Setup Byte 3.** This field provides byte 3 of the last setup packet received. For a Standard Device Request, the most significant byte of the wValue field is returned.	ru	0
23:16	**Setup Byte 2.** This field provides byte 2 of the last setup packet received. For a Standard Device Request, the least significant byte of the wValue field is returned.	ru	0
15:8	**Setup Byte 1.** This field provides byte 1 of the last setup packet received. For a Standard Device Request, the following bRequest Code information is returned: Code Description 00h Get Status 01h Clear Feature 02h Reserved 03h Set Feature 04h Reserved 05h Set Address 06h Get Descriptor 07h Set Descriptor 08h Get Configuration 09h Set Configuration 0Ah Get Interface 0Bh Set Interface 0Ch Synch Frame	ru	0
7:0	**Setup Byte 0.** This field provides byte 0 of the last setup packet received. For a Standard Device Request, the following bmRequestType information is returned: Bit Description 7 Direction: 0 = host to device; 1 = device to host 6:5 Type: 0 = Standard, 1 = Class, 2 = Vendor, 3 = Reserved 4:0 Recipient: 0 = Device, 1 = Interface, 2 = Endpoint, 3 = Other, 4-31 = Reserved	ru	0

11.6.8 (Address 9Ch; SETUP4567) Setup Bytes 4, 5, 6, 7

Bits	Description	Access	Default
31:24	**Setup Byte 7.** This field provides byte 7 of the last setup packet received. For a Standard Device Request, the most significant byte of the wLength field is returned.	ru	0
23:16	**Setup Byte 6.** This field provides byte 6 of the last setup packet received. For a Standard Device Request, the least significant byte of the wLength field is returned.	ru	0
15:8	**Setup Byte 5.** This field provides byte 5 of the last setup packet received. For a Standard Device Request, the most significant byte of the wIndex field is returned.	ru	0
7:0	**Setup Byte 4.** This field provides byte 4 of the last setup packet received. For a Standard Device Request, the least significant byte of the wIndex field is returned.	ru	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.6.9 (Address A4h; OURADDR) Our USB Address

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7	**Force Immediate.** If this bit is set when this register is being written, the NET2280 USB address is updated immediately, without waiting for a valid status phase from the USB host.	su	0
6:0	**Our USB Address.** This field contains the current USB address of the device. This field is cleared when a root port reset is detected. When this field is written, the actual device address isn't changed until a valid status phase is received from the USB host.	rwu	0

11.6.10 (Address A8h; OURCONFIG) Our USB Configuration

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**Our USB Configuration.** This field contains the current USB configuration of the device. This field is cleared when a root port reset is detected.	rwu	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.7 PCI Master Control Registers

Note: USB accesses to the PCIOUT and PCIIN dedicated endpoints will modify the values in these registers.

11.7.1 (Address 100h; PCIMSTCTL); PCI Master Control

Bits	Description	Access	Default
31:16	**Reserved**	r	0
15:13	**PCI Arbiter Park Select.** This field determines which PCI master controller is granted the bus when there are no pending requests. **Value** **Park** 000 Last Grantee 001 8051/USB 010 External Requester 011 Reserved 100 DMA Channel A 101 DMA Channel B 110 DMA Channel C 111 DMA Channel D	rw	0
12	**PCI Multi-Level Arbiter.** When clear, all PCI requesters are placed into a single-level round-robin arbiter, each with equal access to the PCI bus. When set, a 3 level arbiter is selected.	rw	0
11	**PCI Retry Abort Enable.** When clear, the NET2280 will attempt to access a target indefinitely. When set, the NET2280 will abort the transaction after 256 retries, and will set an interrupt. For a DMA transaction, the DMA operation will be aborted.	rw	0
10	**DMA Memory Write and Invalidate Enable.** When clear, the NET2280 DMA controllers will issue a Memory Write command for all write transactions. When set, the DMA controllers will issue a Memory Write and Invalidate command if a transaction is aligned to a cache boundary, and it has at least one cache line of data in the endpoint FIFO. The PCI burst will continue if there are more lines of data in the FIFO. This bit is only effective if the *Memory Write and Invalidate* bit in the **PCICMD** register is set.	rw	0
9	**DMA Read Multiple Enable.** When clear, the NET2280 DMA controllers will issue a Memory Read command for transactions that start on a cache boundary. When set, the DMA controllers will issue a Read Multiple command if a transaction is aligned to a cache boundary, and it has space in a FIFO for at least one cache line of data. The PCI burst will continue if there is space in the FIFO for another line.	rw	0
8	**DMA Read Line Enable.** When clear, the NET2280 DMA controllers will issue a Memory Read command for transactions that don't start on a cache boundary. When set, the DMA controllers will issue a Read Line command if a transaction is not aligned to a cache boundary, and it has space in a FIFO for at least one cache line of data. The PCI burst will be stopped at the cache line boundary if there isn't space in a FIFO for at least one cache line of data or if a Read Multiple command can be started.	rw	0
7:6	**PCI Master Command Select.** When the NET2280 performs PCI transactions initiated by the 8051 or the USB interface, this field determines the PCI command codes that are issued. **Value** **Read Cmd** **Write Cmd** 00 0110 (Mem Read) 0111 (Mem Write) 01 0010 (I/O Read) 0011 (I/O Write) 10 1010 (Cfg Read) 1011 (Cfg Write) 11 Reserved	rw	0
5	**PCI Master Start.** Writing a 1 to this bit causes a PCI read or write transaction to start. The bit is cleared when the PCI transaction is done. For write operations, this bit can be used to determine when to start another write. For read operations, this bit can be used to determine when the **PCIMSTDATA** register has valid data. This bit is automatically cleared if a master abort, target abort, or retry abort occurs.	rsu	0
4	**PCI Master Read/Write.** When clear, a PCI write transaction is selected. When set, a PCI read transaction is selected.	rw	0
3:0	**PCI Master Byte Enables.** This field determines which PCI byte enables (CBE[3:0]#) are asserted during the data phase of a master cycle.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.7.2 (Address 104h; PCIMSTADDR); PCI Master Address

Bits	Description	Access	Default
31:0	**PCI Master Address.** This field determines the target PCI address of a PCI master transaction.	rw	0

11.7.3 (Address 108h; PCIMSTDATA); PCI Master Data

Bits	Description	Access	Default
31:0	**PCI Master Data.** Data being transferred to/from the PCI Bus is accessed through this register. For PCI reads, this register is valid when the *PCI Master Start* bit has been cleared.	rw	0

11.7.4 (Address 10Ch; PCIMSTSTAT); PCI Master Status

Bits	Description	Access	Default
31:3	**Reserved**	r	0
2	**PCI Arbiter Clear.** When an external device is granted the bus, and it doesn't assert FRAME# within 16 clock cycles, it is prevented from being granted the bus in the future. Writing a 1 to this bit clears the lockout, and allows the external device to be granted the bus again. This bit is self-clearing, and always returns a 0.	cu	0
1	**PCI External Arbiter.** This bit reflects the state of the EXTARB pin. When low, the NET2280 enables its internal arbiter. It then expects external requests on REQ# and issues a bus grant on GNT#. When high, the NET2280 asserts REQ# and expects GNT# from an external arbiter.	r	--
0	**PCI Host Mode.** This bit reflects the status of the HOST pin. When low, the device operates in PCI Adapter mode. When high, the device operates in PCI Host mode.	r	--

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.8 DMA Controller Registers

Register	Channel A	Channel B	Channel C	Channel D
DMACTL	180h	1A0h	1C0h	1E0h
DMASTAT	184h	1A4h	1C4h	1E4h
DMACOUNT	190h	1B0h	1D0h	1F0h
DMAADDR	194h	1B4h	1D4h	1F4h
DMADESC	198h	1B8h	1D8h	1F8h

11.8.1 (DMACTL) DMA Control

Bits	Description	Access	Default
31:26	**Reserved.**	r	0
25	**DMA Scatter/Gather Done Interrupt Enable.** When set, an interrupt is generated when the last descriptor in the scatter/gather linked list has completed its transfer (*End of Chain* bit set).	rw	0
24:22	**Reserved.**	r	0
21	**DMA Clear Count Enable.** When clear, the byte count field of a DMA descriptor is not changed at the end of a transfer. When set, the byte count field and Valid Bit of a DMA descriptor are set to zero at the end of a transfer. If a DMA transfer is stopped prematurely, the number of bytes left to transfer is written to the byte count field.	rw	0
20:19	**Descriptor Polling Rate.** This field selects the polling rate when a cleared valid bit is detected. Value Description 0 Continuous 1 1 usec 2 100 usec 3 1 msec	rw	1
18	**DMA Valid Bit Polling Enable.** When clear, the DMA chaining controller stops polling for valid descriptors if it encounters a **Valid Bit** that is not set. When set, the DMA chaining controller continues polling the same descriptor if it encounters a *Valid Bit* that is not set. This bit is only valid if the *DMA Valid Bit Enable* and *DMA Scatter/Gather Enable* bits are set.	rw	0
17	**DMA Valid Bit Enable.** When clear, DMA descriptors are processed without regard to the *Valid Bit* (**DMACOUNT**[31]). When set, DMA descriptors are processed only when the *Valid Bit* is set. If a valid descriptor with a zero byte count is encountered, the DMA chaining controller moves on to the next descriptor.	rw	0
16	**DMA Scatter/Gather Enable.** When clear, a single DMA operation is performed when the *DMA Start* bit is set. The **DMACOUNT** and **DMAADDR** registers are used to define the transfer. When set, the DMA controller traverses a linked list of descriptors, each of which defines a DMA operation. The list of descriptors is located in PCI system memory.	rw	0
15:5	**Reserved.**	r	0
4	**DMA OUT Auto Start Enable.** When set, this DMA channel will automatically start when an OUT packet is received.	rw	0
3	**DMA Preempt Enable.** When set, this DMA channel can be preempted if the 8051 or USB need access to the PCI bus.	rw	0
2	**DMA FIFO Validate.** When clear, the last short packet will not be automatically validated at the end of a DMA transfer. When set, the last short packet is automatically validated at the end of a DMA.	rw	0
1	**DMA Enable.** When set, the DMA controller is enabled to transfer data. If cleared, the DMA controller is disabled. If a DMA transfer is in progress when this bit is cleared, the transfer is paused. This bit is automatically cleared if the *DMA Abort* bit is set or if a PCI master or target abort occurs during a DMA transaction.	rwu	0
0	**DMA Address Hold.** When clear, the PCI address bus is incremented after each data transfer. When set, the address does not increment during the DMA transfer.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.8.2 (DMASTAT) DMA Status

Bits	Description	Access	Default
31:26	**Reserved.**	r	0
25	**DMA Scatter/Gather Done Interrupt.** This bit is set when the last descriptor in the scatter/gather linked list has completed its transfer. Writing a 1 clears this status bit.	rcu	0
24	**DMA Transaction Done Interrupt.** This bit is set when the current DMA in progress completes its transfer and the *DMA Done Interrupt Enable* bit in the **DMACOUNT** register is set. Writing a 1 clears this status bit.	rcu	0
23:2	**Reserved.**	r	0
1	**DMA Abort.** Writing a 1 to this bit causes the DMA transfer to be aborted. This bit is self-clearing, and always returns a 0.	su	0
0	**DMA Start.** Writing a 1 to this bit causes the DMA controller to start. This bit is self-clearing, and always returns a 0.	su	0

11.8.3 (DMACOUNT) DMA Byte Count

Bits	Description	Access	Default
31	**Valid Bit.** This bit indicates the validity of the current DMA descriptor.	rwu	0
30	**DMA Direction.** When clear, the DMA channel transfers data from the USB to PCI bus for OUT endpoints. When set, the DMA channel transfers data from the PCI bus to the USB for IN endpoints.	rwu	0
29	**DMA Done Interrupt Enable.** When set, an interrupt is generated when the *DMA Byte Count* for this descriptor reaches zero.	rwu	0
28	**End of Chain.** When clear, there are additional DMA descriptors following the current one being processed. When set, this descriptor is the last one in the scatter/gather chain.	rwu	0
27:24	**Reserved**	r	0
23:0	**DMA Byte Count.** This register determines the total number of bytes to be transferred. The maximum DMA transfer size is 16 Mbytes. This register is decremented as data is transferred.	rwu	0

11.8.4 (DMAADDR) DMA Address

Bits	Description	Access	Default
31:0	**DMA Address.** This register determines the PCI bus starting address of a DMA transfer. DMA starting addresses may be aligned on any byte boundary. The DMA address may or may not be incremented depending upon the *DMA Address Hold* bit.	rwu	0

11.8.5 (DMADESC) DMA Descriptor

Bits	Description	Access	Default
31:4	**Next Descriptor Address.** This field points to the next DMA descriptor to be processed.	rwu	0
3:0	**Reserved.** Since the descriptors are aligned on 16-byte boundaries, these bits are forced to zero.	r	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.9 Dedicated Endpoint Registers

Register	CFGOUT	CFGIN	PCIOUT	PCIIN	STATIN
DEP_CFG	200h	210h	220h	230h	240h
DEP_RSP	204h	214h	224h	234h	244h

11.9.1 (DEP_CFG) Dedicated Endpoint Configuration

Bits	Description	Access	Default
31:9	**Reserved**	r	0
10	**Endpoint Enable.** When set, this bit enables this endpoint.	rw	1
9	**Reserved.**	r	0
8	**Endpoint Type.** When clear, the STATIN endpoint is a BULK endpoint. When set, the STATIN endpoint is an INTERRUPT endpoint. This bit is only valid for the STATIN endpoint.	rw	STATIN = 1, Others = 0
7:4	**Reserved.**	r	0
3:0	**Endpoint Number.** This field selects the number of the endpoint. Valid numbers are 0 to 15.	rw	CFGOUT = ‘hD, CFGIN = ‘hD, PCIOUT = ‘hE, PCIIN = ‘hE, STATIN = ‘hF

11.9.2 (DEP_RSP) Dedicated Endpoint Response

Note: Writing a 1 to bits 1:0 clears the corresponding register bits, and writing a 1 to bits 9:8 sets the corresponding register bits. When this register is read, bits 1:0 are duplicated on bits 9:8.

Bits	Description	Access	Default
31:10	**Reserved.**	r	0
9	**Endpoint Toggle.**	rs	0
8	**Endpoint Halt**	rs	0
7:2	**Reserved**	r	0
1	**Endpoint Toggle.** This bit is used to clear the endpoint data toggle bit. Reading this bit returns the current state of the endpoint data toggle bit. Under normal operation, the toggle bit is controlled automatically, so the CPU (8051 or PCI) does not need to use this bit.	rc	0
0	**Endpoint Halt.** This bit is used to clear the endpoint halt bit. When an Endpoint Clear Feature Standard Request to the halt bit is detected by the CPU (8051 or PCI), it must write a 1 to this bit. Reading this bit returns the current state of the endpoint halt bit.	rc	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.10 Configurable Endpoint / FIFO Registers

There are 7 sets of endpoint / FIFO registers, one for each configurable endpoint and one for endpoint 0.

Register	EP 0	EP A	EP B	EP C	EP D	EP E	EP F
EP_CFG	300h	320h	340h	360h	380h	3A0h	3C0h
EP_RSP	304h	324h	344h	364h	384h	3A4h	3C4h
EP_IRQENB	308h	328h	348h	368h	388h	3A8h	3C8h
EP_STAT	30Ch	32Ch	34Ch	36Ch	38Ch	3ACh	3CCh
EP_AVAIL	310h	330h	350h	370h	390h	3B0h	3D0h
EP_DATA	314h	334h	354h	374h	394h	3B4h	3D4h

11.10.1 (EP_CFG) Endpoint Configuration

NOTE: For Endpoint 0, all fields in this register except *Endpoint Direction* and *Endpoint Byte Count* are assigned to fixed values, and are **RESERVED**.

Bits	Description	Access	Default
31:19	**Reserved**	r	0
18:16	**Endpoint Byte Count.** This field determines how many bytes are written to the endpoint FIFO for the next write transaction. Following the next FIFO write transaction, this field is restored to its default value. If the value in this field is less than four, the next FIFO write transaction will cause a short packet to be validated. This field is set to its default value when the corresponding FIFO is flushed.	rwu	4
15:11	**Reserved**	r	0
10	**Endpoint Enable.** When set, this bit enables this endpoint. This bit has no effect on Endpoint 0, which is always enabled.	rw	0
9:8	**Endpoint Type.** This field selects the type of this endpoint. Endpoint 0 is forced to a Control type. Value Description 0 **Reserved** 1 Isochronous 2 Bulk 3 Interrupt	rw	EP_0 = 0, EP_A-EP_F = 2
7	**Endpoint Direction.** This bit selects the direction of this endpoint. 0 = OUT, 1 = IN. The direction is with respect to the USB host point of view. Note that a maximum of one OUT and IN endpoint is allowed for each endpoint number. For endpoint 0, this bit is automatically changed according to the direction specified in the SETUP packet.	rwu	0
6:4	**Reserved**	r	0
3:0	**Endpoint Number.** This field selects the number of the endpoint. Valid numbers are 0 to 15. This field has no effect on Endpoint 0, which always has an endpoint number of 0.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.10.2 (EP_RSP) Endpoint Response

Note: Writing a 1 to bits 7:0 clears the corresponding register bits, and writing a 1 to bits 15:8 sets the corresponding register bits. When this register is read, bits 7:0 are duplicated on bits 15:8.

Bits	Description	Access	Default
31:16	**Reserved.**	r	0
15	**NAK OUT Packets.**	rs	0
14	**EP Hide Status Phase.**	rs	0
13	**Endpoint Force CRC Error.**	rs	0
12	**Interrupt Mode.**	rs	0
11	**Control Status Phase Handshake.**	rs	0
10	**'NAK OUT Packets' Mode.**	rs	1
9	**Endpoint Toggle.**	rs	0
8	**Endpoint Halt**	rs	0
7	**NAK OUT Packets.** This bit is set when a short data packet is received from the host by this endpoint, and the *NAK OUT Packets Mode* bit is set. If this bit is set and another OUT token is received, a NAK is returned to the host if another OUT packet is sent to this endpoint. This bit can also be cleared by a bit in the **EP_STAT** register.	rc	0
6	**EP Hide Status Phase.** When set, the *DATA Packet Received* and *Data Packet Transmitted* interrupts and ACK, NAK, and Timeout status bits for control status phase packets are not set. Note that the *Data OUT/PING Token Interrupt* and *Data IN Token Interrupt* bits are still set, independent of this bit. This bit is not used for normal operation, and is intended for special applications.	rc	0
5	**Endpoint Force CRC Error.** When set, all IN packets transmitted and OUT packets received by this endpoint are forced to have a CRC error.	rc	0
4	**Interrupt Mode.** This bit is only used for INTERRUPT endpoints. For normal interrupt data, this bit should be set to zero and standard data toggle protocol is followed. When this interrupt endpoint is used for isochronous rate feedback information, this bit should be set high. In this mode the data toggle bit is changed after each packet is sent to the host without regard to handshaking. No packet retries are performed in the rate feedback mode.	rc	0
3	**Control Status Phase Handshake.** This bit is only used for endpoint 0. This bit is automatically set when a setup packet is detected. While the bit is set, a control status phase will be acknowledged with a NAK. Once cleared, the proper response will be returned to the host (ACK for Control Reads and zero-length packets for Control Writes).	rc	0
2	**'NAK OUT Packets' Mode.** This bit is only used for OUT endpoints, and selects the response if the previous receive packet has not been processed by the CPU (8051 or PCI). If this bit is low (non-blocking mode), the NET2280 will accept the OUT packet if there is space available in the endpoint's FIFO. If this bit is high (blocking mode), the NET2280 will respond to an OUT packet with a NAK if the *NAK OUT Packets* bit is set.	rc	1
1	**Endpoint Toggle.** This bit is used to clear the endpoint data toggle bit. Reading this bit returns the current state of the endpoint data toggle bit. Under normal operation, the toggle bit is controlled automatically, so the CPU (8051 or PCI) does not need to use this bit.	rc	0
0	**Endpoint Halt.** This bit is used to clear the endpoint halt bit. When an Endpoint Clear Feature Standard Request to the halt bit is detected by the CPU (8051 or PCI), it must write a 1 to this bit. Reading this bit returns the current state of the endpoint halt bit. For Endpoint 0, the halt bit is automatically cleared when another Setup packet is received.	rc	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.10.3 (EP_IRQENB) Endpoint Interrupt Enable

Bits	Description	Access	Default
31:7	**Reserved.**	r	0
6	**Short Packet OUT Done Interrupt Enable.** When set, this bit enables an interrupt to be set when a short OUT packet has been successfully transferred by the DMA controller to the PCI target. This bit is only valid for endpoints A, B, C, and D.	rw	0
5	**Short Packet Transferred Interrupt Enable.** When set, this bit enables an interrupt to be set when the length of the last packet was less than the Maximum Packet Size (determined by **EP_n_FS_MAXPKT** or **EP_n_HS_MAXPKT** registers).	rw	0
4	**Reserved.**	rw	0
3	**Data Packet Received Interrupt Enable.** When set, this bit enables an interrupt to be set when a data packet has been received from the host.	rw	0
2	**Data Packet Transmitted Interrupt Enable.** When set, this bit enables an interrupt to be set when a data packet has been transmitted to the host.	rw	0
1	**Data OUT/PING Token Interrupt Enable.** When set, this bit enables an interrupt to be set when a Data OUT or PING token has been received from the host.	rw	0
0	**Data IN Token Interrupt Enable.** When set, this bit enables an interrupt to be set when a Data IN token has been received from the host.	rw	0

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.10.4 (EP_STAT) Endpoint Status

Synchronization Considerations

Due to internal synchronization delays, *Data Packet Received Interrupt*, *Short Packet Transferred Interrupt*, and *FIFO Full and Empty* status may change at slightly different times - up to 100ns apart. Firmware should take care to avoid confusion due to interpreting combinations of bits, particularly bits within the same register.

For example, if the FIFO is empty and a new OUT packet is received, the *Data Packet Received Interrupt* bit may be set up to 600ns before the *FIFO Empty* bit goes false and the *Short Packet Transferred Interrupt* bit is set. Firmware reading **EP_STAT** inside this window might interpret this as an OUT packet with 0 bytes (because *FIFO Empty* is not yet cleared) but which is not a short packet (because *Short Packet Transferred Interrupt* status is not yet set).

Firmware polling for any of these status bits to change should read the register a second time after the desired bit has changed in order to get consistent information.

Bits	Description	Access	Default
31:28	**Reserved**	r	0
27:24	**FIFO Valid Count.** This field provides the number of short packets that are currently in the IN FIFO. If the value is non-zero, a packet is returned in response to an IN token. The count is incremented when a short packet is validated, and is decremented when a short packet is successfully sent to the host. This field is automatically cleared by a FIFO Flush operation, or when a SETUP packet is received by endpoint 0.	ru	0
23:22	**High-Bandwidth OUT Transaction PID.** This field provides the PID of the last high bandwidth OUT packet received. It is stable when the *Data Packet Received Interrupt* bit is set, and remains stable until another OUT packet is received. Value PID 00 DATA0 01 DATA1 10 DATA2 11 MDATA	r	0
21	**Timeout.** For an IN endpoint, the last USB packet transmitted was not acknowledged by the USB host, indicating a bus error. The USB host will expect the same packet to be retransmitted in response to the next IN token. For an OUT endpoint, the last USB packet received had a CRC or bit-stuffing error, and was not acknowledged by the NET2280. The USB host will retransmit the packet. Writing a 1 clears this bit.	rcu	0
20	**USB STALL Sent.** The last USB packet could not be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL. Writing a 1 clears this bit.	rcu	0
19	**USB IN NAK Sent.** The last USB IN packet could not be provided, and was acknowledged with a NAK. Writing a 1 clears this bit.	rcu	0
18	**USB IN ACK Rcvd.** The last USB IN data packet transferred was successfully acknowledged with an ACK from the host. Writing a 1 clears this bit.	rcu	0
17	**USB OUT NAK Sent.** The last USB OUT data packet could not be accepted, and was acknowledged with a NAK to the host. Writing a 1 clears this bit.	rcu	0
16	**USB OUT ACK Sent.** The last USB OUT data packet transferred was successfully acknowledged with an ACK to the host. Writing a 1 clears this bit.	rcu	0
15:14	**Reserved.**	r	0
13	**FIFO Overflow.** When set, this bit indicates that an attempt was made to write to the FIFO when the FIFO was full. Writing a 1 clears this bit.	rcu	0
12	**FIFO Underflow.** When set, this bit indicates that an attempt was made to read the FIFO when the FIFO was empty. Writing a 1 clears this bit.	rcu	0
11	**FIFO Full.** When set, this bit indicates that the FIFO is full.	r	0
10	**FIFO Empty.** When set, this bit indicates that the FIFO is empty.	r	1

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

9	**FIFO Flush.** Writing a 1 to this bit causes the FIFO to be flushed and the corresponding **EP_AVAIL** register and *FIFO Valid Count* field to be cleared. This bit is self-clearing, and reading always returns a 0.	cu	0
8	**Reserved.**	r	0

7	**Reserved.**	rcu	0
6	**Short Packet OUT Done Interrupt.** This bit is set when a short (or zero length) packet has been received from the USB host, and the OUT FIFO becomes empty during a DMA PCI write. Subsequent OUT packets (after the short packet) can be prevented from entering the FIFO by selecting the NAK OUT Packets Mode. To clear this bit, *Short Packet Transferred Interrupt* bit must also be cleared (either at the same time or earlier). This bit is only valid for endpoints A, B, C, and D.	rcu	0
5	**Short Packet Transferred Interrupt.** This bit is set if the length of the last packet was less than the Maximum Packet Size. Writing a 1 clears this bit.	rcu	0
4	**NAK OUT Packets.** This bit is set when a short data packet is received from the host by this endpoint, and the *NAK OUT Packets Mode* bit in the **EP_RSP** register is set. Writing a 1 clears this status bit. If this bit is set and another OUT token is received, a NAK is returned to the host if another OUT packet is sent to this endpoint. This bit can also be controlled by the **EP_RSP** register.	rcu	0
3	**Data Packet Received Interrupt.** This bit is set when a data packet is received from the host by this endpoint. Writing a 1 clears this bit.	rcu	0
2	**Data Packet Transmitted Interrupt.** This bit is set when a data packet is transmitted from the endpoint to the host. Writing a 1 clears this bit.	rcu	0
1	**Data OUT/PING Token Interrupt.** This bit is set when a Data OUT or PING token has been received from the host. Writing a 1 clears this bit.	rcu	0
0	**Data IN Token Interrupt.** This bit is set when a Data IN token has been received from the host. Writing a 1 clears this bit.	rcu	0

11.10.5 (EP_AVAIL) Endpoint Available Count

Bits	Description	Access	Default
31:12	**Reserved.**	r	0
11:0	**Endpoint Available Count.** For an OUT endpoint, this register returns the number of valid bytes in the endpoint FIFO. Values range from 0 (empty) to 2048 (full) for endpoints A and B, from 0 to 1024 for endpoints C and D, and from 0 to 64 for endpoints 0, E, and F. For endpoint 0, this field provides the FIFO count only when data is being received from the host. For an IN endpoint, this register returns the number of empty bytes in the FIFO. Values range from 0 (full) to 2048 (empty) for endpoints A and B, from 0 to 1024 for endpoints C and D, and from 0 to 64 for endpoints 0, E, and F. For endpoint 0, this field provides the number of empty bytes in the FIFO when data is being sent to the host.	r	--

11.10.6 (EP_DATA) Endpoint Data

Bits	Description	Access	Default
31:0	**Endpoint Data.** For an OUT endpoint, this register is used by the CPU (8051 or PCI) to read data from the endpoint's FIFO. For an IN endpoint, this register is used by the CPU (8051 or PCI) to write data to the endpoint's FIFO. The endpoint FIFOs can also be accessed through **PCI Base Address Register 2**.	rw	---

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11 Indexed Registers

11.11.1 (Index 00h; DIAG) Diagnostic Control

Bits	Description	Access	Default
31:16	**Retry Counter.** This field is an accumulator of packet retries. It is cleared when this field is written with any value.	rcu	0
15:12	**Reserved**	r	0
11	**Force PCI SERR.** When set, this bit forces the PCI SERR# pin asserted if the *SERR# Enable* bit in the **PCICMD** register is set.	rw	0
10	**Force PCI Interrupt.** When set, this bit forces the PCI INTA# pin asserted.	rw	0
9	**Force USB Interrupt.** When set, this bit forces a STATIN endpoint interrupt to be generated.	rw	0
8	**Force CPU Interrupt.** When set, this bit forces an interrupt to the 8051 CPU.	rw	0
7:6	**Reserved**	r	0
5	**Illegal Byte Enables.** This bit is set when an external PCI master has performed a transaction to/from one of the endpoint FIFOs, and illegal byte enables were detected. The only valid byte enable combinations (shown active high) are: 0001, 0011, 0111, and 1111. Writing a 1 clears this bit.	rcu	0
4	**Fast Times**. When set, internal timers and counters operate at a fast speed for factory chip testing purposes only.	rw	0
3	**Reserved.**	r	0
2	**Force Receive Error.** When set, an error is forced on the next received data packet. As a result, the packet will not be acknowledged. This bit is automatically cleared at the end of the next packet.	rsu	0
1	**Reserved.**	r	0
0	**Force Transmit CRC Error.** When set, a CRC error is forced on the next transmitted data packet. Inverting the most significant bit of the calculated CRC generates the CRC error. This bit is automatically cleared at the end of the next packet.	rsu	0

11.11.2 (Index 01h; PKTLEN) Packet Length

Bits	Description	Access	Default
31:11	**Reserved.**	r	0
10:0	**Packet Length.** This field provides the length of the last packet transferred. This field is **not** updated when setup packets are received, because they have a fixed length of 8. This field is intended for debugging purposes, and could be in the process of being updated by the USB section when being read. It should be read twice to verify the value.	r	0

11.11.3 (Index 02h; FRAME) Frame Counter

Bits	Description	Access	Default
31:11	**Reserved.**	r	0
10:0	**Frame Counter.** This field contains the frame counter from the most recent start-of-frame packet. This field could be in the process of being updated by the USB section when being read, and should be read twice to verify the value.	r	0

11.11.4 (Index 03h; CHIPREV) Chip Revision

Bits	Description	Access	Default
31:16	**Reserved**	r	0
15:0	**Chip Revision.** This register returns the current silicon revision number of the NET2280.	r	'h0110

Note: CHIPREV is the silicon revision, encoded as a 4-digit BCD value. The least-significant digit is incremented for mask changes, and the most-significant digit is incremented for major revisions.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11.5 (Index 06h; HS_MAXPOWER) High-Speed Maximum Power

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**High-Speed Maximum Power.** The amount of current drawn by the peripheral from the USB port in increments of 2 mA. The NET2280 reports this value to the USB host in the configuration descriptor. The default is 222 mA (6Fh * 2 mA). This register is only used when the Get Configuration Descriptor request is in Auto-Enumerate mode, and the device is operating in high-speed mode.	rw	6Fh

11.11.6 (Index 07h; FS_MAXPOWER) Full-Speed Maximum Power

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**Full-Speed Maximum Power.** The amount of current drawn by the peripheral from the USB port in increments of 2 mA. The NET2280 reports this value to the USB host in the configuration descriptor. The default is 222 mA (6Fh * 2 mA). This register is only used when the Get Configuration Descriptor request is in Auto-Enumerate mode, and the device is operating in full-speed mode.	rw	6Fh

11.11.7 (Index 08h; HS_INTPOLL_RATE) High-Speed Interrupt Polling Rate

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**High-Speed Interrupt Polling Rate.** This field specifies the interrupt polling rate in terms of microframes (125 µsec). It is returned as the last byte of all interrupt endpoint descriptors when the Get Configuration Descriptor is set to Auto-Enumerate mode, and the device is operating in high-speed mode.	rw	FFh

11.11.8 (Index 09h; FS_INTPOLL_RATE) Full-Speed Interrupt Polling Rate

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**Full-Speed Interrupt Polling Rate.** This field specifies the interrupt polling rate in milliseconds. It is returned as the last byte of all interrupt endpoint descriptors when the Get Configuration Descriptor is set to Auto-Enumerate mode, and the device is operating in full-speed mode.	rw	FFh

11.11.9 (Index 0Ah; HS_NAK_RATE) High-Speed NAK Rate

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**High-Speed NAK rate.** This field specifies the maximum NAK rate of high-speed bulk/control endpoints in response to OUT packets. A value of 0 indicates the endpoint never NAKs an OUT packet. Other values indicate at most 1 NAK each **HS_NAK_RATE** number of microframes. Values range from 0 to 255.	rw	0

11.11.10 (Index 0Bh; SCRATCH) Scratchpad

Bits	Description	Access	Default
31:0	**Scratchpad**. General purpose scratchpad register.	rw	FEED FACEh

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11.11 (Index 20h; EP_A_HS_MAXPKT) High-Speed Max Packet Size

Bits	Description	Access	Default
31:13	**Reserved**	r	0
12:11	**Additional Transaction Opportunities.** This field determines the number of additional transaction opportunities per microframe for high-speed isochronous and interrupt endpoints. 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved	rw	0
10:0	**High-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in high-speed mode.	rw	512

11.11.12 (Index 21h; EP_A_FS_MAXPKT) Full-Speed Max Packet Size

Bits	Description	Access	Default
31:11	**Reserved**	r	0
10:0	**Full-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in full-speed mode.	rw	64

11.11.13 (Index 30h; EP_B_HS_MAXPKT) High-Speed Max Packet Size

Bits	Description	Access	Default
31:13	**Reserved**	r	0
12:11	**Additional Transaction Opportunities.** This field determines the number of additional transaction opportunities per microframe for high-speed isochronous and interrupt endpoints. 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved	rw	0
10:0	**High-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in high-speed mode.	rw	512

11.11.14 (Index 31h; EP_B_FS_MAXPKT) Full-Speed Max Packet Size

Bits	Description	Access	Default
31:11	**Reserved**	r	0
10:0	**Full-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in full-speed mode.	rw	64

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11.15 (Index 40h; EP_C_HS_MAXPKT) High-Speed Max Packet Size

Bits	Description	Access	Default
31:13	**Reserved**	r	0
12:11	**Additional Transaction Opportunities.** This field determines the number of additional transaction opportunities per microframe for high-speed isochronous and interrupt endpoints. 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved	rw	0
10:0	**High-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in high-speed mode.	rw	512

11.11.16 (Index 41h; EP_C_FS_MAXPKT) Full-Speed Max Packet Size

Bits	Description	Access	Default
31:11	**Reserved**	r	0
10:0	**Full-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in full-speed mode.	rw	64

11.11.17 (Index 50h; EP_D_HS_MAXPKT) High-Speed Max Packet Size

Bits	Description	Access	Default
31:13	**Reserved**	r	0
12:11	**Additional Transaction Opportunities.** This field determines the number of additional transaction opportunities per microframe for high-speed isochronous and interrupt endpoints. 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved	rw	0
10:0	**High-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in high-speed mode.	rw	512

11.11.18 (Index 51h; EP_D_FS_MAXPKT) Full-Speed Max Packet Size

Bits	Description	Access	Default
31:11	**Reserved**	r	0
10:0	**Full-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in full-speed mode.	rw	64

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11.19 (Index 60h; EP_E_HS_MAXPKT) High-Speed Max Packet Size

Bits	Description	Access	Default
31:13	**Reserved**	r	0
12:11	**Additional Transaction Opportunities.** This field determines the number of additional transaction opportunities per microframe for high-speed isochronous and interrupt endpoints. 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved	rw	0
10:0	**High-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in high-speed mode.	rw	512

11.11.20 (Index 61h; EP_E_FS_MAXPKT) Full-Speed Max Packet Size

Bits	Description	Access	Default
31:11	**Reserved**	r	0
10:0	**Full-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in full-speed mode.	rw	64

11.11.21 (Index 70h; EP_F_HS_MAXPKT) High-Speed Max Packet Size

Bits	Description	Access	Default
31:13	**Reserved**	r	0
12:11	**Additional Transaction Opportunities.** This field determines the number of additional transaction opportunities per microframe for high-speed isochronous and interrupt endpoints. 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved	rw	0
10:0	**High-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in high-speed mode.	rw	512

11.11.22 (Index 71h; EP_F_FS_MAXPKT) Full-Speed Max Packet Size

Bits	Description	Access	Default
31:11	**Reserved**	r	0
10:0	**Full-Speed Max Packet Size.** This field determines the Endpoint Maximum Packet Size when in full-speed mode.	rw	64

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

11.11.23 (Index 84h; STATIN_HS_INTPOLL_RATE) High-Speed Interrupt Polling Rate

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**STATIN High-Speed Interrupt Polling Rate.** This field specifies the interrupt polling rate in terms of microframes (125 µsec). It is returned as the last byte of the STATIN interrupt endpoint descriptor when the Get Configuration Descriptor is set to Auto-Enumerate mode, and the device is operating in high-speed mode.	rw	1

11.11.24 (Index 85h; STATIN_FS_INTPOLL_RATE) Full-Speed Interrupt Polling Rate

Bits	Description	Access	Default
31:8	**Reserved**	r	0
7:0	**STATIN Full-Speed Interrupt Polling Rate.** This field specifies the interrupt polling rate in milliseconds. It is returned as the last byte of the STATIN interrupt endpoint descriptor when the Get Configuration Descriptor is set to Auto-Enumerate mode, and the device is operating in full-speed mode.	rw	1

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12 USB Standard Device Requests

Standard device requests must be supported by Endpoint 0. See also chapter 9, USB specification. In Local-Enumerate mode, the CPU (8051 or PCI) decodes the setup packets for Endpoint 0 and generates a response based on the following tables. In Auto-Enumerate mode, the request is handled automatically by the NET2280 without intervention from the CPU (8051 or PCI).

Table 12-1: Standard Request Codes

bRequest	Value
Get_Status	0
Clear_Feature	1
Reserved	2
Set_Feature	3
Reserved	4
Set_Address	5
Get_Descriptor	6
Set_Descriptor	7
Get_Configuration	8
Set_Configuration	9
Get_Interface	Ah
Set_Interface	Bh
Synch_Frame	Ch

Table 12-2: Descriptor Types

Descriptor Types	Value
Device	1
Configuration	2
String	3
Interface	4
Endpoint	5
Device Qualifier	6
Other_Speed_Configuration	7
Interface Power	8

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.1 Control Read Transfers

12.1.1 Get Device Status

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	2	bits 15:2 = Reserved bit 1 = Device Remote Wakeup enabled bit 0 = Power supply is good in Self-Powered mode.	Bit 1 determined by **USBCTL** *Device Remote Wake-up Enable* bit Bit 0 determined by **USBCTL** *Self Powered Status* bit	Determined by CPU

12.1.2 Get Interface Status

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	2	bits 15:0 = Reserved	0000h	0000h

12.1.3 Get Endpoint Status

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	2	bits 15:1 = Reserved bit 0 = Endpoint is halted	Determined by stall state of endpoint	Determined by CPU

12.1.4 Get Device Descriptor (18 Bytes)

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Length	12h	12h
1	1	Type (device)	01h	01h
2	2	USB Specification Release Number	0210h	0210h
4	1	Class Code	FFh	FFh
5	1	Sub Class Code	00h	00h
6	1	Protocol	00h	00h
7	1	Maximum Endpoint 0 Packet Size	40h	40h
8	2	Vendor ID	Determined by **PRODVENDID** Register	0525h
10	2	Product ID	Determined by **PRODVENDID** Register	2280h
12	2	Device Release Number	Determined by **RELNUM** Register	0110h
14	1	Index of string descriptor describing manufacturer	01h (if **USBCTL** *Vendor. String Enable = 1)* 00h (if **USBCTL** *Vendor String Enable = 0)*	01h
15	1	Index of string descriptor describing product	02h (if **USBCTL** *Product String Enable = 1)* 00h (if **USBCTL** *Product String Enable = 0)*	02h
16	1	Index of string descriptor describing serial number	Determined by **USBCTL** *Serial Number Index* field	00h (not enabled)
17	1	Number of configurations	01h	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.1.5 Get Device Qualifier (10 Bytes)

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Length	Not Supported	0Ah
1	1	Type (device qualifier)		06h
2	2	USB Specification Release Number		0210h
4	1	Class Code		FFh
5	1	Sub Class Code		00h
6	1	Protocol		00h
7	1	Maximum Endpoint 0 Packet Size for other speed		40h
8	1	Number of other-speed configurations		Determined by CPU
9	1	Reserved		00h

12.1.6 Get Other_Speed_Configuration Descriptor

The structure of the other_speed_cofiguration is identical to a configuration descriptor, except that the bDescriptorType is 7 instead of 2.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.1.7 Get Configuration Descriptor

The NET2280 can support a variety of configurations, interfaces, and endpoints, each of which is defined by the descriptor data returned to the USB host. In the Local-Enumerate mode (**STDRSP[4]** low), the CPU (8051 or PCI) has the responsibility of providing this data to the NET2280 when the host requests it. In the Auto-Enumerate mode (**STDRSP[4]** high), the descriptor is returned automatically to the host without CPU intervention. In the Auto-Enumerate mode, only one configuration and interface are supported.

12.1.7.1 Local-Enumerate

This example has one configuration, and two interfaces, each with two configurable endpoints. The first interface defines one Bulk OUT endpoint at address 4 with a full-speed maximum packet size of 64 and one Interrupt IN endpoint at address 85h (endpoint number = 5) with a maximum packet size of 8. The second interface defines one Bulk OUT endpoint at address 6 with a full-speed maximum packet size of 64 and one Bulk IN endpoint at address 86h (endpoint number = 6) with a full-speed maximum packet size of 64. Both of these interfaces also have the five dedicated endpoints of the NET2280.

Note that all interface and endpoint descriptors are returned in response to a Get Configuration Descriptor request, and for this example, 125 bytes are returned.

Offset	Number of Bytes	Description	Suggested Value
Configuration Descriptor			
0	1	Length	09h
1	1	Type (configuration)	02h
2	2	Total length returned for this configuration	007Dh
4	1	Number of Interfaces	02h
5	1	Number of this configuration	01h
6	1	Index of string descriptor describing this configuration	00h
7	1	Attributes bit 7 = 1 bit 6 = Self-Powered bit 5 = Remote-Wakeup bits 4:0 = Reserved	Determined by CPU
8	1	Maximum USB power required (in 2 mA units)	Determined by CPU
Interface 0 Descriptor			
0	1	Size of this descriptor in bytes	09h
1	1	Type (interface)	04h
2	1	Number of this interface	00h
3	1	Alternate Interface	00h
4	1	Number of endpoints used by this interface (excluding Endpoint 0)	07h
5	1	Class Code	FFh
6	1	Sub Class Code	00h
7	1	Device Protocol	00h
8	1	Index of string descriptor describing this interface	00h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
CFGOUT Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	01h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0010h (FS) or 0200h(HS)
6	1	Maximum NAK rate of the endpoint.	Determined by CPU
CFGIN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	81h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0008h (FS) or 0200h(HS)
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
PCIOUT Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	02h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0010h (FS) or 0200h(HS)
6	1	Maximum NAK rate of the endpoint.	Determined by CPU
PCIIN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	82h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0008h (FS) or 0200h(HS)
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
STATIN Interrupt IN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	83h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	03h
4	2	bits 10:0 = Maximum packet size of this endpoint. bits 12:11 = Number of additional transaction opportunities per microframe (high-speed only): 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15:13 = reserved	0004h
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
Bulk OUT Endpoint A Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	04h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0040h (FS) or 0200h(HS)
6	1	Maximum NAK rate of the endpoint.	Determined by CPU
Interrupt IN Endpoint B Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	85h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	03h
4	2	bits 10:0 = Maximum packet size of this endpoint. bits 12:11 = Number of additional transaction opportunities per microframe (high-speed only): 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15:13 = reserved	0008h (FS or HS)
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
Interface 1 Descriptor			
0	1	Size of this descriptor in bytes	09h
1	1	Type (interface)	04h
2	1	Number of this interface	01h
3	1	Alternate Interface	00h
4	1	Number of endpoints used by this interface (excluding Endpoint 0)	07h
5	1	Class Code	FFh
6	1	Sub Class Code	00h
7	1	Device Protocol	00h
8	1	Index of string descriptor describing this interface	00h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
CFGOUT Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	01h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0008h (FS) or 0200h(HS)
6	1	Maximum NAK rate of the endpoint.	Determined by CPU
CFGIN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	81h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0008h (FS) or 0200h(HS)
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
PCIOUT Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	02h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint	0010h (FS) or 0200h(HS)
6	1	Maximum NAK rate of the endpoint.	Determined by CPU
PCIIN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	82h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0008h (FS) or 0200h(HS)
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
STATIN Interrupt IN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	83h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	03h
4	2	bits 10:0 = Maximum packet size of this endpoint. bits 12:11 = Number of additional transaction opportunities per microframe (high-speed only): 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15:13 = reserved	0004h
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
Bulk OUT Endpoint C Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	06h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0040h (FS) or 0200h(HS)
6	1	Maximum NAK rate of the endpoint.	Determined by CPU
Bulk IN Endpoint D Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	86h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0040h (FS) or 0200h(HS)
6	1	Interval for polling endpoint	Determined by CPU

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.1.7.2 Auto-Enumerate

This example has one configuration, one interface, five dedicated endpoints, and four configurable endpoints. The configurable endpoints consists of one Bulk OUT endpoint at address 4 with maximum high-speed packet size of 512, an Interrupt IN endpoint at address 85h (endpoint number = 5) with a maximum packet size of 256, an Isochronous OUT endpoint at address 6 with maximum packet size of 1024, and a Bulk IN endpoint at address 87h (endpoint number = 7) with a maximum high-speed packet size of 512.

Note that all interface and endpoint descriptors are returned in response to a Get Configuration Descriptor request, and for this example, 81 bytes are returned. This example assumes that the NET2280 is operating as a high-speed device.

Offset	Number of Bytes	Description	Auto-Enumerate Value
Configuration Descriptor			
0	1	Length	09h
1	1	Type (configuration)	02h
2	2	Total length returned for this configuration	0051h
4	1	Number of Interfaces	01h
5	1	Number of this configuration	01h
6	1	Index of string descriptor describing this configuration	00h
7	1	Attributes bit 7 = 1 bit 6 = Self-Powered (determined by *Self-Powered USB Device* bit of **USBCTL** register). bit 5 = Remote-Wakeup capability (Determined by *Remote Wakeup Support* bit of **USBCTL** register) bits 4:0 = Reserved	Determined by **USBCTL** Register
8	1	Maximum USB power required (in 2 mA units).	Determined by **HS_MAXPOWER** Register
Interface 0 Descriptor			
0	1	Size of this descriptor in bytes	09h
1	1	Type (interface)	04h
2	1	Number of this interface	00h
3	1	Alternate Interface	00h
4	1	Number of endpoints used by this interface (excluding Endpoint 0)	09h
5	1	Class Code	FFh
6	1	Sub Class Code	00h
7	1	Device Protocol	00h
8	1	Index of string descriptor describing this interface	00h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
CFGOUT Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	01h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0200h
6	1	Maximum NAK rate of the endpoint.	Determined by **HS_NAK_RATE** register
CFGIN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	81h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0200h
6	1	Interval for polling endpoint	00h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
PCIOUT Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	02h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0200h
6	1	Maximum NAK rate of the endpoint.	Determined by **HS_NAK_RATE** register
PCIIN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	82h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	bits 10:0 = Maximum packet size of this endpoint.	0200h
6	1	Interval for polling endpoint	00h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Suggested Value
STATIN Interrupt IN Endpoint Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	83h
3	1	Endpoint Attributes bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	03h
4	2	bits 10:0 = Maximum packet size of this endpoint. bits 12:11 = Number of additional transaction opportunities per microframe (high-speed only): 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15:13 = reserved	0004h
6	1	Interval for polling endpoint	Determined by **STATIN_HS_INTPOLL_RATE** register

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Auto-Enumerate Value
Bulk OUT Endpoint A Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address (Determined by **EP_CFG** register). bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	04h
3	1	Endpoint Attributes (Determined by **EP_CFG** register). bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	Maximum packet size of this endpoint. (Determined by the **EP_A_HS_MAXPKT** register).	0200h
6	1	Maximum NAK rate of the endpoint.	Determined by **HS_NAK_RATE** register
Interrupt IN Endpoint B Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address (Determined by **EP_CFG** register). bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	85h
3	1	Endpoint Attributes (Determined by **EP_CFG** register). bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	03h
4	2	bits 10:0 = Maximum packet size of this endpoint. (Determined by the **EP_B_HS_MAXPKT** register). bits 12:11 = Number of additional transaction opportunities per microframe: 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15:13 = reserved	0100h
6	1	Interval for polling endpoint	Determined by **HS_INTPOLL_RATE** register

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Get Configuration Descriptor (continued)

Offset	Number of Bytes	Description	Auto-enumerate Value
Isochronous OUT Endpoint C Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address (Determined by **EP_CFG** register). bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	06h
3	1	Endpoint Attributes (Determined by **EP_CFG** register). bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	01h
4	2	bits 10:0 = Maximum packet size of this endpoint (Determined by the **EP_C_HS_MAXPKT** register). bits 12:11 = Number of additional transaction opportunities per microframe: 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15:13 = reserved	0400h
6	1	Interval for polling endpoint.	Determined by **HS_INTPOLL_RATE** register
Bulk IN Endpoint D Descriptor			
0	1	Size of this descriptor	07h
1	1	Descriptor Type (endpoint)	05h
2	1	Endpoint Address (Determined by **EP_CFG** register). bit 7 = direction (1 = IN, 0 = OUT) bits 6:4 = reserved bits 3:0 = endpoint number	87h
3	1	Endpoint Attributes (Determined by **EP_CFG** register). bits 7:2 = reserved bits 1:0 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	02h
4	2	Maximum packet size of this endpoint. (Determined by the **EP_D_HS_MAXPKT** register).	0200h
6	1	Interval for polling endpoint.	00h

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.1.8 Get String Descriptor 0

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Size of this descriptor in bytes	04h	04h
1	1	Descriptor type (string)	03h	03h
2	2	Language ID (English = 09, U.S. = 04)	0409h	0409h

12.1.9 Get String Descriptor 1

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Size of this descriptor in bytes	26h	26h
1	1	Descriptor type (string)	03h	03h
2	36	Manufacturer Descriptor. The text string is encoded in UNICODE.	"NetChip Technology"	"NetChip Technology"

12.1.10 Get String Descriptor 2

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Size of this descriptor in bytes	42h	42h
1	1	Descriptor type (string)	03h	03h
2	64	Product Descriptor. The text string is encoded in UNICODE.	"NET2280 USB Interface Controller"	"NET2280 USB Interface Controller"

12.1.11 Get String Descriptor 3

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Size of this descriptor in bytes	Not Enabled	0Ah
1	1	Descriptor type (string)		03h
2	8	Serial Number Descriptor. The text string is encoded in UNICODE.		"1001"

12.1.12 Get Configuration

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Returns current device configuration	00h or 01h	00h or currently selected configuration.

12.1.13 Get Interface

Offset	Number of Bytes	Description	Auto-Enumerate Value	Local-Enumerate Suggested Value
0	1	Returns current alternate setting for the specified interface	00h.	00h or currently selected interface.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.2 Control Write Transfers

12.2.1 Set Address

Note: In Local-enumerate mode, the CPU (8051 or PCI) must write the new device address into the **OURADDR** configuration register. In Auto-enumerate mode, the **OURADDR** configuration register is updated automatically when a Set Address request is completed successfully.

Offset	Number of Bytes	Description	Suggested Value
--	0	Sets USB address of device wValue = device address, wIndex = 0, wLength = 0	--

12.2.2 Set Configuration

Note: In Local-enumerate mode, the CPU (8051 or PCI) must write the new device configuration into the **OURCONFIG** configuration register. In Auto-enumerate mode, the **OURCONFIG** configuration register is updated automatically.

Offset	Number of Bytes	Description	Suggested Value
--	0	Sets the device configuration wValue = Configuration value, wIndex = 0, wLength = 0	--

12.2.3 Set Interface

Note: In Local-enumerate mode, the CPU (8051 or PCI) must keep track of the Interface value. In Auto-enumerate mode, an index and value of 0 are always expected. Any other index or value results in a STALL handshake.

Offset	Number of Bytes	Description	Suggested Value
--	0	Selects alternate setting for specified interface wValue = Alternate setting, wIndex = specified interface, wLength = 0	--

12.2.4 Device Clear Feature

Note: In Local-enumerate mode, the CPU (8051 or PCI) must write the *Device Remote Wake-up Enable* bit of the **USBCTL** register. In Auto-enumerate mode, the *Device Remote Wake-up Enable* bit is updated by this Standard Request.

Offset	Number of Bytes	Description	Suggested Value
--	0	Clear the selected device feature wValue = feature selector, wIndex = 0, wLength = 0 FS = 1 → Device Remote Wakeup (disable)	--

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

12.2.5 Device Set Feature

Offset	Number of Bytes	Description	Suggested Value
--	0	Set the selected device feature wValue = feature selector, wLength = 0 FS = 1 → Device Remote Wakeup (enable), wIndex = 0 FS = 2 → Test Mode, wIndex = specifies test mode	--

12.2.6 Endpoint Clear Feature

Note: In Local-enumerate mode, the CPU (8051 or PCI) must clear the endpoint halt bit by writing to the *Endpoint Halt* bit in the lower byte of the **EP_RSP** register. In Auto-enumerate mode, the appropriate halt bit is cleared automatically.

Offset	Number of Bytes	Description	Suggested Value
--	0	Clear the selected endpoint feature wValue = feature selector, wIndex = endpoint number, wLength = 0 FS = 0 → Endpoint halt (clears halt bit)	--

12.2.7 Endpoint Set Feature

Note: In Local-enumerate mode, the CPU (8051 or PCI) must set the endpoint halt bit by writing to the *Endpoint Halt* bit in the upper byte of the **EP_RSP** register. In Auto-enumerate mode, the appropriate halt bit is set automatically.

Offset	Number of Bytes	Description	Suggested Value
--	0	Set the selected endpoint feature wValue = feature selector, wIndex = endpoint number, wLength = 0 FS = 0 → Endpoint halt (sets halt bit)	--

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13 Electrical Specifications

13.1 Absolute Maximum Ratings

Conditions that exceed the Absolute Maximum limits may destroy the device.

Symbol	Parameter	Conditions	Min	Max	Unit
VDD2, V_{DD25}, P_{VDD}, A_{VDD}	2.5V Supply Voltages	With Respect to Ground	-0.5	3.6	V
VDD3, V_{DD33}	3.3V Supply Voltages	With Respect to Ground	-0.5	4.6	V
V_I	DC input voltage	3.3 V buffer	-0.5	4.6	V
		5 V Tolerant buffer (PCI)	-0.5	5.5	V
I_{OUT}	DC Output Current, per pin	3mA Buffer	-10	10	mA
		12mA Buffer	-40	40	mA
		24mA Buffer (PCI)	-70	70	mA
T_{STG}	Storage Temperature	No bias	-65	150	° C
T_{AMB}	Ambient temperature	Under bias	-40	85	° C
V_{ESD}	ESD Rating	R = 1.5K, C = 100pF		2	KV

13.2 Recommended Operating Conditions

Conditions that exceed the Operating limits may cause the device to function incorrectly.

Symbol	Parameter	Conditions	Min	Max	Unit
VDD2, V_{DD25}, P_{VDD}, A_{VDD}	2.5V Supply Voltages		2.3	2.7	V
VDD3, V_{DD33}	3.3V Supply Voltages		3.0	3.6	V
V_N	Negative trigger voltage	3.3 V buffer	0.8	1.7	V
		5 V tolerant buffer (PCI)	0.8	1.7	V
V_P	Positive trigger voltage	3.3 V buffer	1.3	2.4	V
		5 V tolerant buffer (PCI)	1.3	2.4	V
V_{IL}	Low Level Input Voltage	3.3 V buffer	0	0.7	V
		5 V tolerant buffer (PCI)	0	0.8	V
V_{IH}	High Level Input Voltage	3.3 V buffer	1.7	VDD3	V
		5 V tolerant buffer (PCI)	2.0	VIO+0.5	V
I_{OL}	Low Level Output Current	3 mA buffer		3	mA
		6 mA buffer		6	mA
		12 mA buffer		12	mA
		24 mA buffer (PCI)		24	mA
I_{OH}	High Level Output Current	3 mA buffer		-3	mA
		6 mA buffer		-6	mA
		12 mA buffer		-12	mA
		24 mA buffer (PCI)		-24	mA
T_A	Operating Temperature		0	70	° C
t_R	Input rise times	Normal input	0	200	ns
t_F	Input fall time	Normal input	0	200	ns
t_R	Input rise times	Schmitt input	0	10	ms
t_F	Input fall time	Schmitt input	0	10	ms

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.3 DC Specifications

13.3.1 Core DC Specifications

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

13.3.1.1 Disconnected from USB

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		0.45	0.58	mA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		66.8	77	mA

13.3.1.2 Connected to USB (High-Speed/Un-configured or Configured)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		2	3	mA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		76	89	mA

13.3.1.3 Active (High-Speed)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		6.6	9	mA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		77.6	92	mA

13.3.1.4 Connected to USB (Full-Speed/Un-configured or Configured)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		2	3	mA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		60.6	78	mA

13.3.1.5 Active (Full-Speed)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		6	9	mA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		61	76	mA

13.3.1.6 Suspended (Connected to USB)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		197	227	µA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		0	0	µA

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.3.1.7 Suspended (Disconnected from USB)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDD}	V_{DD} Supply Current	V_{DD} = 3.3V		0	0	µA
I_{VDD2}	V_{DD2} Supply Current	V_{DD2} = 2.5V		0	0	µA

13.3.2 USB Full-Speed DC Specifications

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	Input high level (driven)	Note 4	2.0			V
V_{IHZ}	Input high level (floating)	Note 4	2.7		3.6	V
V_{IL}	Input low level	Note 4			0.8	V
V_{DI}	Differential Input Sensitivity	| (D+) - (D-) |	0.2			V
V_{CM}	Differential Common Mode Range	Includes VDI range	0.8		2.5	V
V_{OL}	Output low level	Notes 4,5	0.0		0.3	V
V_{OH}	Output high level (driven)	Notes 4,6	2.8		3.6	V
V_{SE1}	Single ended one		0.8			V
V_{CRS}	Output signal crossover voltage	Note 10	1.3		2.0	V
C_{IO}	I/O Capacitance	Pin to GND			20	pF

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.3.3 USB High-Speed DC Specifications

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{HSSQ}	High-speed squelch detection threshold (differential signal amplitude)		100		150	mV
V_{HSDSC}	High-speed disconnect detection threshold (differential signal amplitude)		525		625	mV
	High-speed differential input signaling levels	Specified by eye patterns				
V_{HSCM}	Hip-speed data signaling common mode voltage range		-50		500	mV
V_{HSOI}	High-speed idle level		-10		10	mV
V_{HSOH}	High-speed data signaling high		360		440	mV
V_{HSOL}	High-speed data signaling low		-10		10	mV
V_{CHIPRJ}	Chirp J level (differential voltage)		700		1100	mV
V_{CHIPRK}	Chirp K level (differential voltage)		-900		-500	mV
C_{IO}	I/O Capacitance	Pin to GND			20	pF

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.3.4 PCI Bus DC Specification

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IHd}	PCI 3.3V Input High Voltage		0.5*VDD3		VDD3	V
V_{ILd}	PCI 3.3V Input Low Voltage		0		0.7	V
V_{IH}	PCI 5.0V Input High Voltage		2.0		5.5	V
V_{IL}	PCI 5.0V Input Low Voltage		0		0.8	V
I_{IL}	Input Leakage	$0V < V_{IN} < Vdd$	-10		10	µA
I_{OZ}	Hi-Z State Data Line Leakage	$0V < V_{IN} < Vdd$			10	µA
V_{OH3}	PCI 3.3V Output High Voltage	I_{OUT} = -500µA	0.9*VDD3			V
V_{OL3}	PCI 3.3V Output Low Voltage	I_{OUT} = 1500µA			0.1*VDD3	V
V_{OH}	PCI 5.0V Output High Voltage	I_{OUT} = -12mA	2.4			V
V_{OL}	PCI 5.0V Output Low Voltage	I_{OUT} = 12mA			0.4	V
C_{IN}	Input Capacitance	Pin to GND			10	pF
C_{CLK}	CLK Pin Capacitance	Pin to GND	5		12	pF
C_{IDSEL}	IDSEL Pin Capacitance	Pin to GND			8	pF

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.4 AC Specifications

13.4.1 USB Full-Speed Port AC Specifications

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

Symbol	Parameter	Conditions	Waveform	Min	Typ	Max	Unit
T_{FR}	Rise & Fall Times	C_L = 50 pF,	Figure 13-1	4		20	ns
T_{FF}		Note 16		4		20	
T_{FRFM}	Rise/Fall time matching	(T_{FR}/ T_{FF}), Note 10	Figure 13-1	90		110	%
Z_{DRV}	Driver Output Resistance	Steady State Drive		10		15	Ω
$T_{FDRATHS}$	Full-speed Data Rate			11.994	12	12.006	Mbs
T_{DJ1}	Source Differential Driver Jitter to Next Transition	Notes 7,8,10,12	Figure 13-2	-2	0	2	ns
T_{DJ2}	Source Differential Driver Jitter for Paired Transitions	Notes 7,8,10,12	Figure 13-2	-1	0	1	ns
T_{FDEOP}	Source Jitter for Differential Transition to SE0 Transition	Note 8, 11	Figure 13-3	-2	0	5	ns
T_{JR1}	Receiver Data Jitter Tolerance to Next Transition	Note 8	Figure 13-4	-18.5	0	18.5	ns
T_{JR2}	Receiver Data Jitter Tolerance for Paired Transitions	Note 8	Figure 13-4	-9	0	9	ns
T_{EOPT}	Source SE0 interval of EOP		Figure 13-3	160	167	175	ns
T_{FEOPR}	Receiver SE0 interval of EOP	Note 13	Figure 13-3	82			ns
T_{FST}	Width of SE0 interval during differential transition			14			ns

13.4.2 USB High-Speed Port AC Specifications

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

Symbol	Parameter	Conditions	Waveform	Min	Typ	Max	Unit
T_{HSR}	Rise & Fall Times	Note 16		500			ps
T_{HSF}				500			
Z_{DRV}	Driver Output Resistance	Steady State Drive		10		15	Ω
T_{HSDRV}	High-speed Data Rate			479.760	480	480.240	Mbs
	Data source jitter	Specified by eye pattern templates					
	Receiver jitter tolerance	Specified by eye pattern templates					

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.4.3 USB Full-Speed Port AC Waveforms

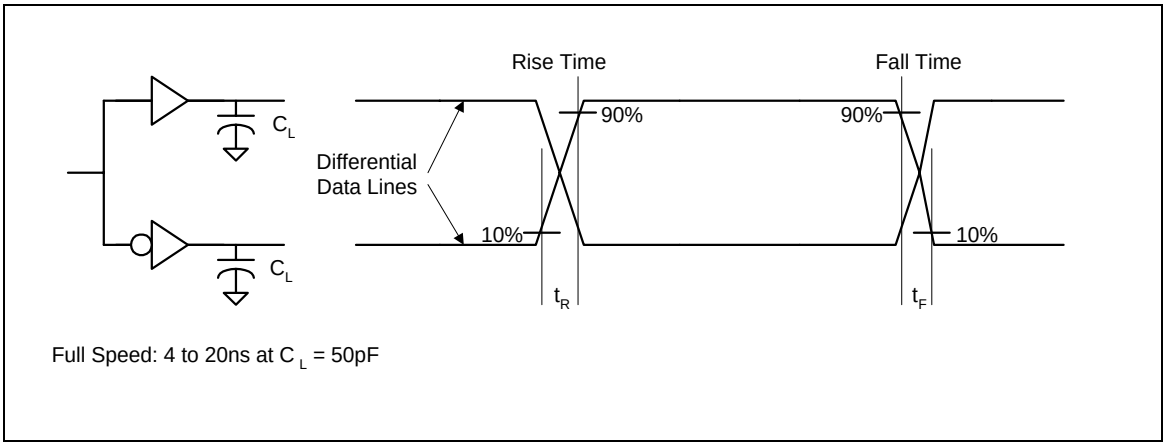


Figure 13-1. Data Signal Rise and Fall Time

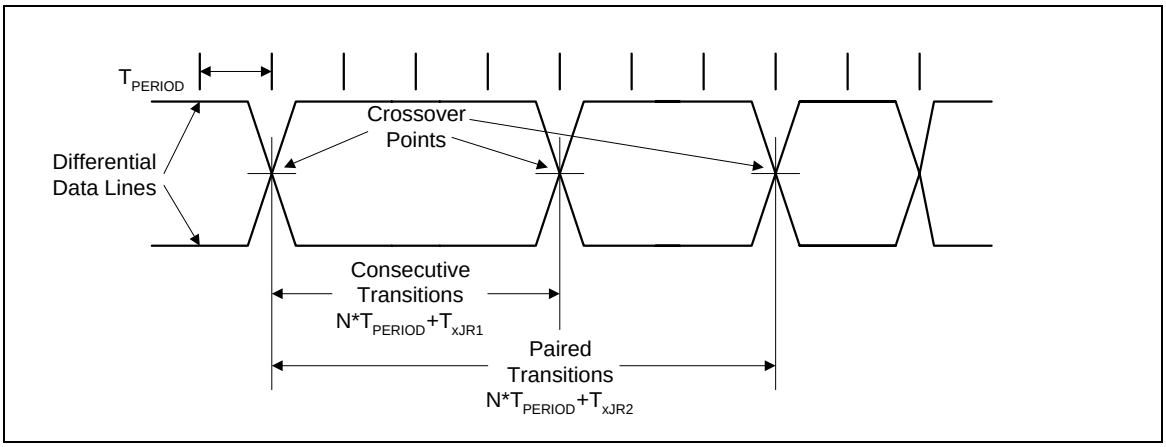
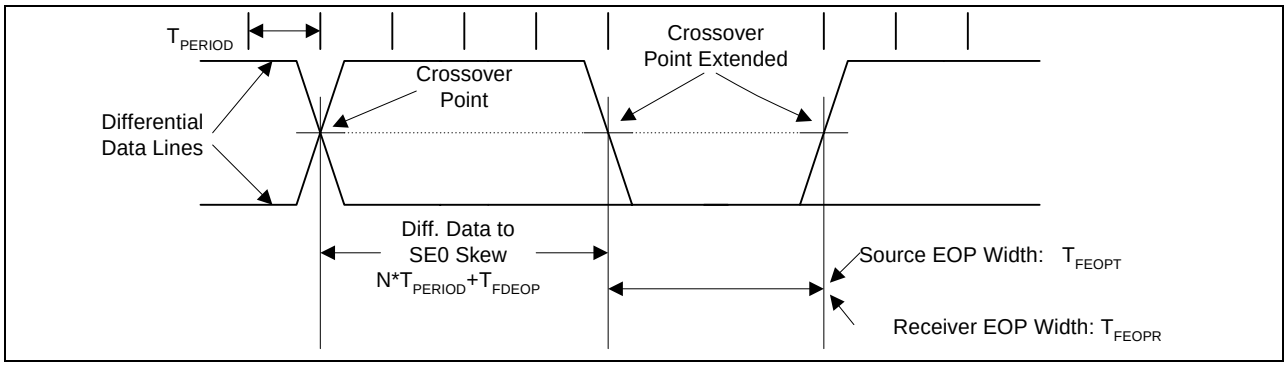


Figure 13-2. Differential Data Jitter



335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

Figure 13-3. Differential to EOP Transition Skew and EOP Width

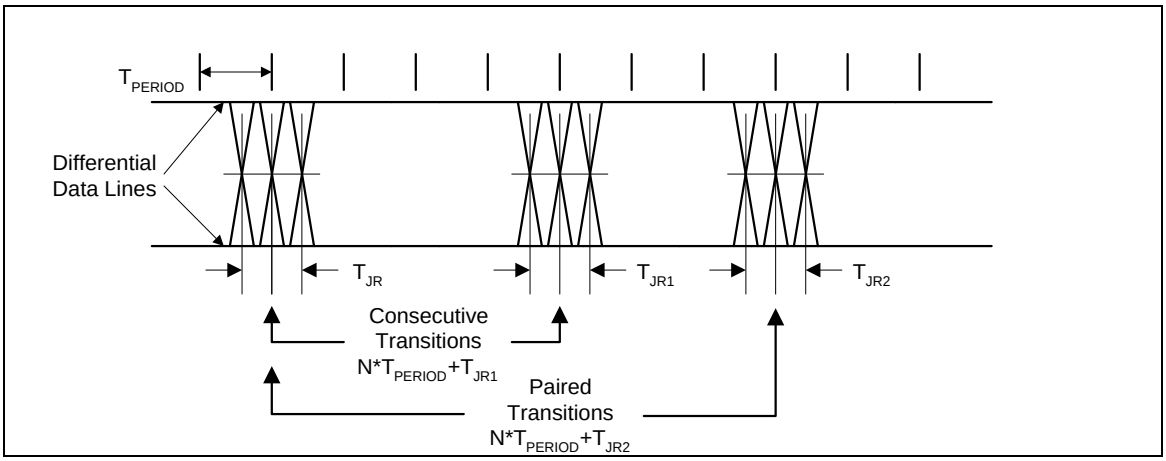


Figure 13-4. Receiver Jitter Tolerance

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.4.4 USB Port AC/DC Specification Notes

1. Measured at A plug.
2. Measured at A receptacle.
3. Measured at B receptacle.
4. Measured at A or B connector.
5. Measured with RL of 1.425KΩ to 3.6V.
6. Measured with RL of 14.25KΩ to GND.
7. Timing difference between the differential data signals.
8. Measured at crossover point of differential data signals.
9. The maximum load specification is the maximum effective capacitive load allowed that meets the target hub V_{BUS} droop of 330 mV.
10. Excluding the first transition from the Idle state.
11. The two transitions should be a (nominal) bit time apart.
12. For both transitions of differential signaling.
13. Must accept as valid EOP.
14. Single-ended capacitance of D+ or D- is the capacitance of D+/D- to all other conductors and, if present, shield in the cable. That is, to measure the single-ended capacitance of D+, short D-, VBUS, GND, and the shield line together and measure the capacitance of D+ to the other conductors.
15. For high power devices (non-hubs) when enabled for remote wakeup.
16. Measured from 10% to 90% of the data signal.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

13.4.5 PCI Bus AC Specifications

Operating Conditions: VDD2: 2.5V ± 0.2V, VDD3: 3.3V ± 0.3V, T_A = 0°C to 70°C
All typical values are at VDD2 = 2.5V, VDD3 = 3.3V and T_A = 25°C

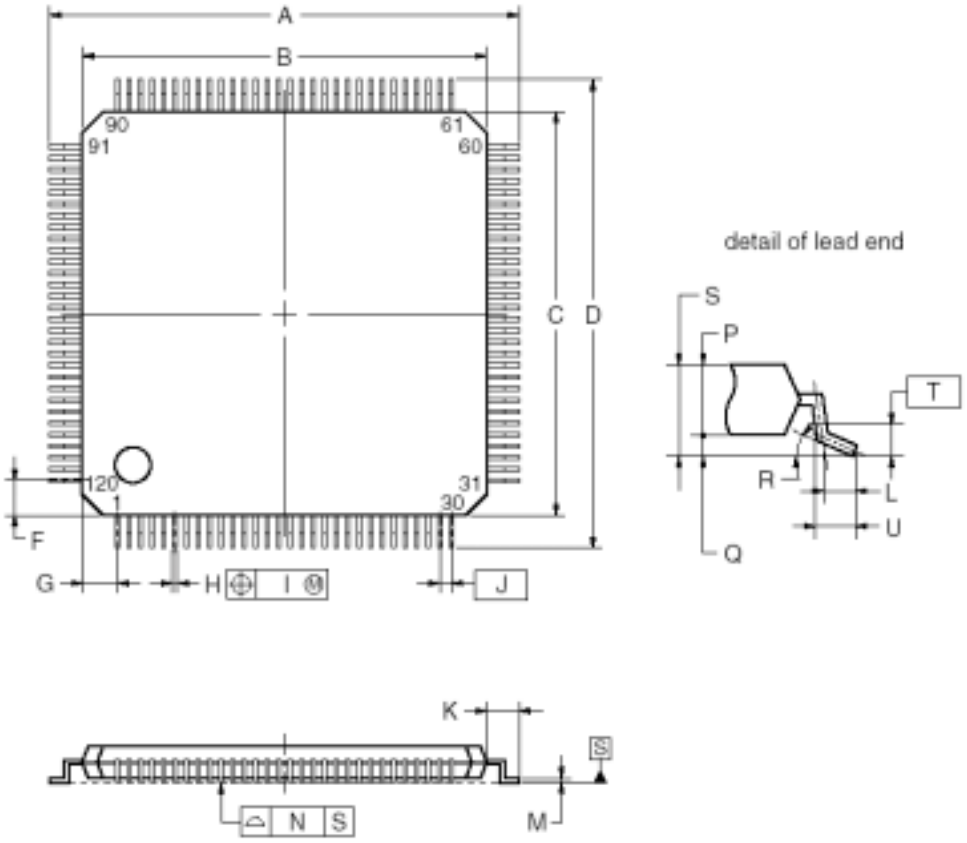
Symbol	Parameter	Conditions	Min	Max	Unit	Notes
T_{CYC}	PCI CLK Cycle Time		30	∞	ns	
T_{VAL}	CLK to signal valid delay – bussed signals		2	11	ns	2, 3
$T_{VAL}(ptp)$	CLK to signal valid delay – point to point		2	12	ns	2, 3
T_{ON}	Float to Active Delay		2		ns	7
T_{OFF}	Active to Float Delay			28	ns	7
T_{SU}	Input Setup to CLK – bussed signals		6		ns	3, 8
$T_{SU}(ptp)$	Input Setup to CLK – point to point		10,12		ns	3
T_H	Input Hold from CLK		0		ns	
T_{RST}	Reset active time after power stable		1		ms	5
$T_{RST-CLK}$	Reset active time after CLK stable		100		µs	5
$T_{RST-OFF}$	Reset active to Output Float delay			40	ns	5, 6, 7
T_{RHFA}	RST# high to first configuration access		2^{25}		clocks	9
T_{RHFF}	RST# high to first FRAME# assertion		5		clocks	

Notes:
2. For parts compliant to the 5V signaling environment:
   Minimum times are evaluated with 0pF equivalent load; maximum times are evaluated with 50pF equivalent load. Actual test capacitance may vary, but results must be correlated to these specifications. Note that faster buffers may exhibit some ring back when attached to a 50pF lump load which should be of no consequence as long as the output buffers are in full compliance with slew rate and V/I curve specifications.
   For parts compliant to the 3.3V signaling environment:
   Minimum times are evaluated with the same load used for slew rate measurement; maximum times are evaluated with a parallel RC load of 25 ohms and 10pF.
3. REQ# and GNT# are point-to-point signals and have different output valid delay and input setup times than do bused signals. GNT# has a setup of 10; REQ# has a setup of 12. All other signals are bused.
5. CLK is stable when it meets the PCI CLK specifications. RST# is asserted and deasserted asynchronously with respect to CLK.
6. All output drivers must be asynchronously floated when RST# is active.
7. For purposes of Active/Float timing measurements, the Hi-Z or "off" state is defined to be when the total current delivered through the component pin is less than or equal to the leakage current specification.
8. Setup time applies only when the device is not driving the pin. Devices cannot drive and receive signals at the same time.
9. At 33 MHz, the device needs to be ready to accept a configuration access within 1 second after RST# is high.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com

14 Mechanical Drawing

120-PIN PLASTIC TQFP (FINE PITCH) (14x14)



P120GC-40-YEB Package

Item	Millimeters	Item	Millimeters
A	16.00 ± 0.20	K	1.00 ± 0.20
B	14.00 ± 0.20	L	0.50
C	14.00 ± 0.20	M	0.17, + 0.03/-0.07
D	16.00 ± 0.20	N	0.08
F	1.20	P	1.00 ± 0.05
G	1.20	Q	0.10 ± 0.05
H	0.18 ± 0.05	R	3°, + 4°/- 3°
I	0.07	S	1.20 MAX
J	0.40 (T.P)	T	0.25

Note: Each lead centerline is located within 0.07mm of its true position (T.P) at maximum material condition.

335 Pioneer Way, Mountain View, California 94041
TEL (650) 526-1490 FAX (650) 526-1494
http://www.netchip.com