# INTEL 430MX PCISET
# 82371MX MOBILE PCI I/O IDE XCELERATOR (MPIIX)

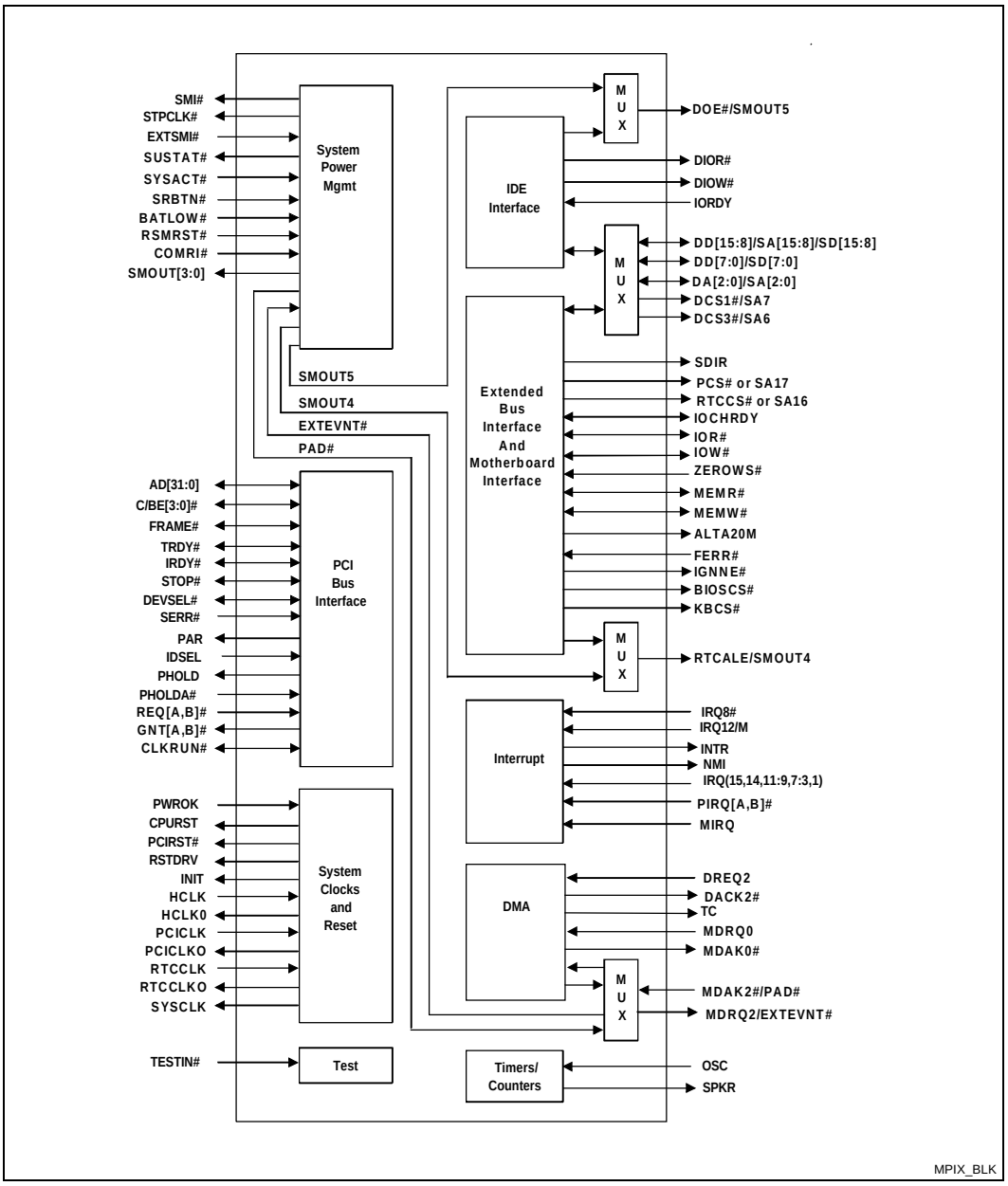
- **Provides a Bridge Between the PCI Bus and Extended I/O Bus**
  - **PCI Bus; 25–33 MHz**
  - **Extended I/O Bus; 7.5–8.33 MHz**
- **System Power Management (Intel SMM Support)**
  - **Programmable System Management Interrupt (SMI)—Hardware/Software Events, EXTSMI#**
  - **Programmable CPU Clock Control (STPCLK#) with Auto Clock Throttle**
  - **Peripheral Device Power Management (Local Standby)**
  - **Suspend State Support (Suspend-to-DRAM and Suspend-to-Disk)**
- **Enhanced DMA Functions**
  - **Two 8237 DMA Controllers**
  - **Fast Type F DMA**
  - **Compatible DMA Transfers**
  - **PC/PCI DMA Expansion for Docking Support**
- **Fast IDE Interface**
  - **PIO Mode 4 Transfers**
  - **2x16-Bit Posted Write Buffer and 1x32-Bit Read Prefetch Buffer**
- **Plug-n-Play Port for Motherboard Devices**
  - **3 Steerable DMA Channels**
  - **1 Steerable Interrupt Line (Plus 2 Steerable PCI Interrupts)**
  - **1 Programmable Chip Select**
- **Functionality of One 82C54 Timer**
  - **System Timer**
  - **Refresh Request**
  - **Speaker Tone Output**
- **Functionality of Two 82C59 Interrupt Controllers**
  - **14 Interrupts Supported**
  - **Independently Programmable for Edge/Level Sensitivity**
- **X-Bus Peripheral Support**
  - **Chip Select Decode**
  - **Controls Lower X-Bus Data Byte Transceiver**
- **Non-Maskable Interrupts (NMI)**
  - **PCI System Error Reporting**
- **NAND Tree for Board-Level ATE Testing**
- **176-Pin TQFP**

The 82371MX PCI I/O IDE Xcelerator (MPIIX) provides the bridge between the PCI bus and the ISA-like Extended I/O expansion bus. In addition, the 82371MX has an IDE interface that supports two IDE devices providing an interface for IDE hard disks and CD ROMs. The MPIIX integrates many common I/O functions found in ISA based PC systems—a seven-channel DMA controller, two 82C59 interrupt controllers, an 8254 timer/counter, Intel SMM power management support, and control logic for NMI generation. Chip select decoding is provided for BIOS, real time clock, and keyboard controller. Edge/Level interrupts and interrupt steering are supported for PCI plug and play compatibility.

The MPIIX also provides the Extended I/O Bus for a direct connection to Super I/O devices providing a complete PC-compatible I/O solution. MPIIX also provides support for the “Mobile PC/PCI” DMA Expansion protocol that enables the implementation of Docking Stations with full ISA and PCI capability without running the full ISA bus across the docking connector. For motherboard Plug-n-Play compatibility, the 82371MX also provides three steerable DMA channels, up to three steerable interrupt lines, and a programmable chip select. The interrupt lines can be routed to any of the available ISA interrupts.

The MPIIX’s power management function supports SMI# interrupt sources, extensive clock control (including Auto Clock Throttling), peripheral power idle detection with access traps, system Suspend-to-DRAM and Suspend-to-Disk.

*Other brands and names are the property of their respective owners.
Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products. Intel retains the right to make changes to those specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.
COPYRIGHT © INTEL CORPORATION, 1996
April 1996 Order Number: 290525-001




**82371MX MPIIX Block Diagram**

# CONTENTS

**1.0. ARCHITECTURE OVERVIEW....9**

**2.0 SIGNAL DESCRIPTION....11**
2.1. PCI Interface Signals....11
2.2. IDE Interface Signals....13
2.3. Extended I/O Bus Signals....14
2.4. Motherboard I/O Device Interface Signals....15
2.5. DMA Signals....17
2.6. Interrupt Controller Signals....17
2.7. System Power Management (SMM) Signals....19
2.8. System Clock And Reset Signals....20
2.9. Test Signals....21

**3.0. REGISTER DESCRIPTION....22**
3.1. Register Access....22
3.2. PCI Configuration Registers....27
3.2.1. VID—VENDOR IDENTIFICATION REGISTER....27
3.2.2. DID—DEVICE IDENTIFICATION REGISTER....27
3.2.3. COM—COMMAND REGISTER....28
3.2.4. DS—DEVICE STATUS REGISTER....28
3.2.5. RID—REVISION IDENTIFICATION REGISTER....29
3.2.6. CLASSC—CLASS CODE REGISTER....29
3.2.7. HEDT—HEADER TYPE REGISTER....30
3.2.8. SPPE—SERIAL & PARALLEL PORT ENABLE REGISTER....30
3.2.9. ECRT— EXTENDED I/O CONTROLLER RECOVERY TIMER REGISTER....31
3.2.10. BIOSE — BIOS ENABLE REGISTER....31
3.2.11. FDCE—FDC ENABLE REGISTER....32
3.2.12. PIRQRC [A,B]—PIRQX ROUTE CONTROL REGISTERS....33
3.2.13. MSTAT—MISCELLANEOUS STATUS REGISTER....33
3.2.14. IDETIM—IDE TIMING REGISTER....34
3.2.15. MIRQRC—MOTHERBOARD DEVICE IRQ ROUTE CONTROL REGISTER....35
3.2.16. MDMARC[2:0]—MOTHERBOARD DEVICE DMA ROUTE CONTROL REGISTERS....36
3.2.17. AUDIOE—AUDIO ENABLE REGISTER....37
3.2.18. DMADS—DMA CH[7:5] DATA SIZE REGISTER....37
3.2.19. PCIDMAE—PCI DMA ENABLE REGISTER....37
3.2.20. PCIDMA[A,B]—PCI DMA AND PCI DMA EXPANSION REGISTER....38
3.2.21. PMAC[1:0]—PROGRAMMABLE MEMORY ADDRESS CONTROL REGISTERS....39
3.2.22. PMAM[1:0]—PROGRAMMABLE MEMORY ADDRESS MASK REGISTERS....40
3.2.23. PARE—PROGRAMMABLE ADDRESS RANGE ENABLE REGISTER....40
3.2.24. PCSC—PROGRAMMABLE CHIP SELECT CONTROL REGISTER....41

3.2.25. PAC[5:1]—PROGRAMMABLE ADDRESS CONTROL REGISTER .....41
3.2.26. PAMA—PROGRAMMABLE ADDRESS MASK A REGISTER .....41
3.2.27. PAMB—PROGRAMMABLE ADDRESS MASK B REGISTER .....42
3.2.28. IOCA—I/O CONFIGURATION ADDRESS REGISTER .....42
3.2.29. PAMC—PROGRAMMABLE ADDRESS MASK C REGISTER .....43
3.2.30. PADE[2:0]—PERIPHERAL ACCESS DETECT ENABLE REGISTERS .....43
3.2.31. LTADEV3—Local Trap Address for Device 3 Register .....44
3.2.32. LTMDEV3—Local Trap Mask for Device 3 Register .....44
3.2.33. LTSMIE—Local Trap SMI Enable Register .....44
3.2.34. LTSMIS—Local Trap SMI Status Register .....45
3.2.35. LSBSMIE—Local Standby SMI Enable Register .....45
3.2.36. LSBTRE—Local Standby Timer Reload Enable Register .....46
3.2.37. LSBSMIS—Local Standby SMI Status Register .....47
3.2.38. LSTBTIDE—Local Standby IDE Timer Register .....47
3.2.39. LSBTAUD—Local Standby Audio Timer Register .....48
3.2.40. LSBTCOM—Local Standby COM Timer Register .....48
3.2.41. LSBTDEV1—Local Standby Device 1 Timer Register .....48
3.2.42. LSBTDEV2—Local Standby Device 2 Timer Register .....49
3.2.43. LSBTDEV3—Local Standby Device 3 Timer Register .....49
3.2.44. SESMIT—Software/EXTSMI# SMI Delay Timer Register .....49
3.2.45. SUSSMIT—Suspend SMI Delay Timer Register .....50
3.2.46. GSBTMR—Global Standby Timer Register .....50
3.2.47. CLKTHSBYT — Clock Throttle Standby Timer Register .....50
3.2.48. SYSMGNTC—System Management Control Register .....51
3.2.49. SYSSMIE—System SMI Enable Register .....51
3.2.50. MISCSMIE—Misc SMI Enable Register .....52
3.2.51. GSMIE—Global SMI Enable Register .....52
3.2.52. SYSSMIS—System SMI STATUS Register .....53
3.2.53. MISCSMIS—Miscellaneous SMI STATUS Register .....53
3.2.54. GSMIS — Global SMI STATUS Register .....54
3.2.55. SUSRSMC1—Suspend/Resume Control 1 Register .....54
3.2.56. SUSRSMC2—Suspend/Resume Control 2 Register .....55
3.2.57. SMOUTC—SMOUT Control Register .....55
3.2.58. SYSEVNTE0—System EVENT Enable 0 Register .....56
3.2.59. SYSEVNTE1—System EVENT Enable 1 Register .....56
3.2.60. SYSEVNTE2—System EVENT Enable 2 Register .....57
3.2.61. BSTCLKT — Burst Count Timer Register .....57
3.2.62. CLKC—Clock Control Register .....58
3.2.63. STPCLKLT—STPCLK# Low Timer Register .....58
3.2.64. STPCLKHT—STPCLK# High Timer Count .....59
3.2.65. STPBRKE0—Stop Break Event Enable 0 Register .....59
3.2.66. STPBRKE1—Stop Break Event Enable 1 Register .....60

3.2.67. STPBRKE2—Stop Break Event Enable 2 Register ..... 60
3.2.68. SHDW—Shadow Register Access Port ..... 61
3.2.69. BSTCLKEE[6:0]—Burst Clock Event Enable Registers ..... 63
3.2.70. CLKTHLBRKEE[6:0]—Clock Throttle Break Event Enable Registers ..... 64
3.3. ISA Compatible Registers ..... 64
3.3.1. DMA REGISTERS ..... 64
3.3.1.1. DCOM—DMA Command Register ..... 65
3.3.1.2. DCM—DMA Channel Mode Register ..... 65
3.3.1.3. DR—DMA Request Register ..... 66
3.3.1.4. Mask Register—Write Single Mask Bit ..... 66
3.3.1.5. Mask Register—Write All Mask Bits ..... 67
3.3.1.6. DS—DMA Status Register ..... 68
3.3.1.7. DMA Base and Current Address Registers (8237 Compatible Segment) ..... 68
3.3.1.8. DMA Base and Current Byte/Word Count Registers (Compatible Segment) ..... 69
3.3.1.9. DMA Memory Low Page Registers ..... 69
3.3.1.10. DMA Clear Byte Pointer Register ..... 70
3.3.1.11. DMC—DMA Master Clear Register ..... 70
3.3.1.12. DCLM—DMA Clear Mask Register ..... 70
3.3.2. TIMER/COUNTER REGISTERS ..... 71
3.3.2.1. TCW—Timer Control Word Register ..... 71
3.3.2.2. Interval Timer Status Byte Format Register ..... 73
3.3.2.3. Counter Access Ports Register ..... 73
3.3.3. INTERRUPT CONTROLLER REGISTERS ..... 74
3.3.3.1. ICW1—Initialization Command Word 1 Register ..... 74
3.3.3.2. ICW2—Initialization Command Word 2 Register ..... 75
3.3.3.3. ICW3—Initialization Command Word 3 Register ..... 75
3.3.3.4. ICW3—Initialization Command Word 3 Register ..... 75
3.3.3.5. ICW4—Initialization Command Word 4 Register ..... 76
3.3.3.6. OCW1—Operational Control Word 1 Register ..... 76
3.3.3.7. OCW2—Operational Control Word 2 Register ..... 77
3.3.3.8. OCW3—Operational Control Word 3 Register ..... 77
3.3.3.9. ELCR1—Edge/Level Triggered Register ..... 78
3.3.3.10. ELCR2—Edge/Level Triggered Register ..... 79
3.3.4. RESET EXTENDED I/O-BUS IRQ12 AND IRQ1 REGISTER ..... 79
3.3.5. NMI REGISTERS ..... 80
3.3.5.1. NMISC—NMI Status and Control Register ..... 80
3.3.5.2. NMI Enable and Real-Time Clock Address Register ..... 81
3.3.5.3. Coprocessor Error Register ..... 81
3.3.5.4. RC—Reset Control Register ..... 81
3.3.5.5. Port 92 Register ..... 82
3.4. Advanced Power Management Registers ..... 83
3.4.1. APMC—ADVANCED POWER MANAGEMENT CONTROL PORT ..... 83

3.4.2. APMS—ADVANCED POWER MANAGEMENT STATUS PORT ....83

**4.0. FUNCTIONAL DESCRIPTION....84**
4.1. Memory And I/O Address Map ....84
4.1.1. I/O ACCESSES....84
4.1.2. BIOS MEMORY ACCESS....84
4.1.3. PERIPHERAL CHIP SELECTS....85
4.2. PCI Interface....86
4.2.1. TRANSACTION TERMINATION ....86
4.2.2. PARITY SUPPORT ....86
4.2.3. PCI ARBITRATION ....87
4.2.4. PCI CLOCK CONTROL (CLKRUN#) ....87
4.3. Extended I/O Bus ....87
4.3.1. EXTENDED I/O BUS CYCLES FOR MPIIX AS A MASTER (PCI MASTER INITIATED) ....88
4.3.2. EXTENDED I/O BUS DMA (8-BIT AND 16-BIT TRANSFERS) ....89
4.4. DMA Controller....90
4.4.1. TYPE F TIMING ....90
4.4.2. DMA buffer for PCI DMA type F transfers ....91
4.4.3. EXTENDED I/O BUS DMA ARBITRATION ....91
4.4.4. PCI DMA....91
4.4.4.1. PCI DMA Expansion Protocol ....91
4.4.4.2. PCI DMA Expansion Cycles ....92
4.4.4.3. Normal DMA Cycle ....94
4.4.4.4. Normal DMA Cycle with Terminal Count ....95
4.4.4.5. Verify DMA Cycle....96
4.4.4.6. Verify DMA Cycle with Terminal Count ....97
4.5. IDE Interface....98
4.5.1. ATA REGISTER BLOCK DECODE ....98
4.5.2. ENHANCED TIMING MODES ....99
4.5.2.1. IORDY masking ....100
4.5.2.2. PIO 32 bit IDE data port mode ....100
4.6. Interval Timer ....100
4.7. Interrupts....101
4.7.1. PROGRAMMING THE INTERRUPT CONTROLLER ....101
4.7.1.1. Edge and Level Triggered Mode ....102
4.7.2. INTERRUPT STEERING ....102
4.7.3. MOUSE FUNCTION....103
4.7.4. COPROCESSOR ERROR FUNCTION ....103
4.7.5. NMI SUPPORT ....103
4.8. Power Management Support ....104
4.8.1. SMI GENERATION....106
4.8.1.1. SMI Enables ....106

4.8.1.2. SMI Request Status ..... 106
4.8.1.3. SMI# Signal Generation ..... 108
4.8.1.4. SMI SOURCES ..... 108
4.8.2. CPU POWER MANAGEMENT (CPU, DRAM, L2 CACHE, DATAPATH) ..... 110
4.8.2.1. Stop Clock ..... 110
4.8.2.2. Software Control of STPCLK# ..... 112
4.8.2.3. Emulating Clock Division (Clock Throttling) ..... 112
4.8.2.4. STPCLK Control State Machine ..... 113
4.8.2.5. Auto Clock Throttle (ACT) Feature ..... 114
4.8.3. LOCAL STANDBY (PERIPHERAL MANAGEMENT) ..... 117
4.8.3.1. Local Standby Sequence ..... 117
4.8.3.2. Access Ranges ..... 118
4.8.3.3. Idle Timers ..... 118
4.8.3.4. Access Traps ..... 119
4.8.3.5. SMOUT Programmable Outputs ..... 119
4.8.4. SUSPEND ..... 119
4.8.4.1. Suspend mode selects ..... 120
4.8.4.2. Suspend SMI# Requests (SRBTN# and BATLOW#) ..... 120
4.8.4.3. Suspend Status (SUSTAT#) Signal and Register ..... 121
4.8.4.4. POWER PLANE CONTROL ..... 122
4.8.4.5. SHADOW REGISTERS ..... 123
4.8.5. SUMMARY OF TIMER RANGES ..... 123
4.9. Reset Support ..... 124

**5.0. PINOUT AND PACKAGE INFORMATION ..... 125**
5.1. Pinout Information ..... 125
5.2. Package Information ..... 129

**6.0. TESTABILITY ..... 130**

## 1.0. ARCHITECTURE OVERVIEW

This section provides a brief overview of the MPIIX. More detailed descriptions are provided in the Signal Description, Register Description, and Functional Description sections.

**Power Management.** Flexible power management capabilites of the MPIIX permit the operating system and system software to efficiently manage the use of system resources. Various low power states are supported while providing the best performance to the user. MPIIX uses several mechanisms to help the power management software initiate and manage the transitions between the power managed states. These include, System Event Monitors such as Idle Timers to identify peripheral and system-wide idle and wake-up conditions, Intel's System Management Interrupt (SMI) support, Advanced Power Management (APM) interface, Pentium® Processor STPCLK# Clock Control, and Low Power Suspend/Resume hardware.

**Docking Support.** MPIIX provides the mechanisms necessary to implement a docking solution that supports both PCI and ISA in the docking station. DMA information is sent across the PCI bus according to the PC/PCI DMA expansion protocol. All ISA IRQx lines are provided. All cycles intended for the MPIIX are positively decoded so that the bus bridge in a docking station can be the subtractive decode agent.

**Fast IDE Interface.** The MPIIX supports one IDE connector on the motherboard (up to 2 devices) and PIO IDE transfers up to 14 Mbytes/sec. The IDE interface has a 2-word write poster and read prefetcher for optimal transfers.

**Plug-n-Play Interface.** The MPIIX provides a Plug-n-Play interface for motherboard devices consisting of 3 steerable DMA channels, 1 steerable interrupt line, and 1 programmable chip select. Each steerable DMA channel supports TYPE F transfers and can use a 4-byte buffer.

**PCI Bus Interface.** The MPIIX provides both a master and slave interface to the PCI bus. As a PCI master, the MPIIX runs cycles on behalf of DMA. As a PCI slave, the MPIIX accepts cycles initiated by PCI masters targeted for the MPIIX's internal register set or the Extended I/O bus. The MPIIX directly supports the PCI Bus running at either 25 MHz or 30 MHz.

**Extended I/O Bus.** The MPIIX incorporates an 8-bit ISA-like interface for motherboard devices such as Multi-Function I/O, Keyboard Controller, Audio Chip, ROM or Flash memory, and a Real Time Clock. MPIIX also includes a 16-bit IDE interface. All cycles to this interface are positively decoded. One programmable Chip Select I/O range, PCS#, and 5 additional programmable I/O ranges are provided for other devices on the Extended I/O bus.

**DMA.** The DMA controller incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels. Channels [3:0] are hardwired to 8-bit, count by bytes transfers, and channels [7:5] can be programmed to either 16-bit, count by words transfers, or 8-bit transfers. All seven channels support fast DMA type F timings using the steerable DMA channels.

**Timer.** The timer block contains three counters that are equivalent in function to those found in one 82C54 programmable interval timer. These counters provide the system timer function and speaker tone. The 14.31818 MHz oscillator input provides the clock source for the counters.

**Interrupt Controller.** The MPIIX provides an ISA compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The two interrupt controllers are cascaded so that 14 external and 1 internal interrupts are possible.

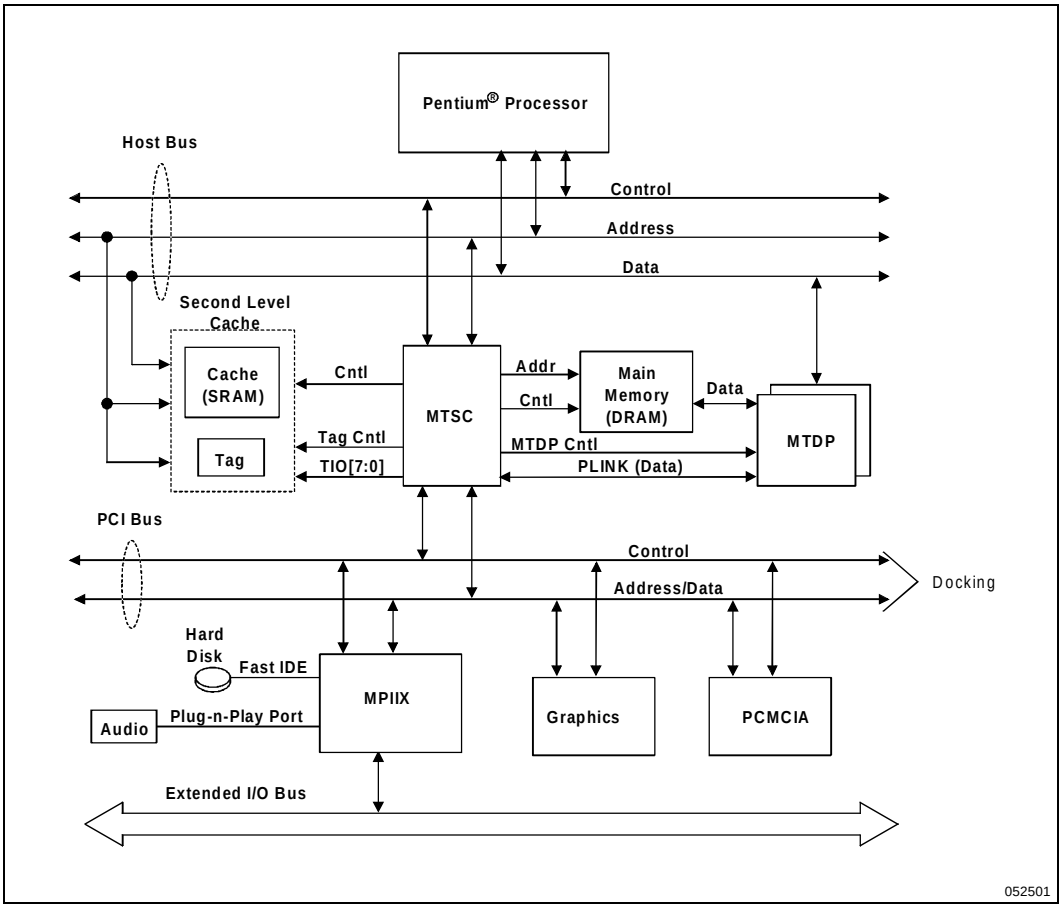



**Figure 1. Intel 430MX PCIset PCIset System**

## 2.0 SIGNAL DESCRIPTION

This section provides a detailed description of each signal. The signals are arranged in functional groups according to their associated interface.

The '#' symbol at the end of a signal name indicates that the active, or asserted state occurs when the signal is at a low voltage level. When '#' is not present after the signal name, the signal is asserted when at the high voltage level.

The terms assertion and negation are used extensively. This is done to avoid confusion when working with a mixture of 'active-low' and 'active-high' signals. The term **assert**, or **assertion** indicates that a signal is active, independent of whether that level is represented by a high or low voltage. The term **negate**, or **negation** indicates that a signal is inactive.

Certain signals are used to drive other signals with different functions through external buffers or transceivers. Both functions have been noted in the descriptions below, with the signal whose function is being described in **bold** font. The actual name given to the pin is the signal driven by MTSC.

The "**PCIRST#**" column indicates the state of the signals during reset.

The following notations are used to describe the signal type.

**I** Input is a standard input-only signal.

**O** Totem pole output is a standard active driver.

**o/d** Open drain.

**t/s** Tri-State is a bi-directional, tri-state input/output pin.

**s/t/s** Sustained tri-state is an active low tri-state signal owned and driven by one and only one agent at a time. The agent that drives a s/t/s pin low must drive it high for at least one clock before letting it float. A new agent can not start driving a s/t/s signal any sooner than one clock after the previous owner tri-states it. An external pull-up is required to sustain the inactive state until another agent drives it and must be provided by the central resource.

**3.3V** Indicates a standard 3.3V low voltage TTL interface.

**5/3V** Indicates that this signal is normally 5V, but will be powered by the RTC voltage on the VDDR "resume well" power supply pin during the suspend state at normal 3.3 volts.

**pu** Internal Pull-Up

**pd** Internal Pull-Down

**bk** Internal Bus Keeper

### 2.1. PCI Interface Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| AD[31:0] | I/O 5V | Tri-state | **PCI ADDRESS/DATA:** The standard PCI address and data lines. The address is driven with FRAME# assertion and data is driven or received in following clocks. |
| C/BE[3:0]# | I/O 5V | Tri-state | **BUS COMMAND AND BYTE ENABLES:** The command is driven with FRAME# assertion. Byte enables corresponding to supplied or requested data is driven on following clocks. |

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| FRAME# | I/O (s/t/s) 5V | Tri-state | **FRAME:** Assertion indicates the address phase of a PCI transfer. Negation indicates that one more data transfer is desired by the cycle initiator. This signal requires a 2.7 KΩ pullup resistor. |
| TRDY# | I/O (s/t/s) 5V | Tri-state | **TARGET READY:** Asserted when the target is ready for a data transfer. This signal requires a 2.7 KΩ pullup resistor. |
| IRDY# | I/O (s/t/s) 5V | Tri-state | **INITIATOR READY:** Asserted when the initiator is ready for a data transfer. This signal requires a 2.7 KΩ pullup resistor. |
| STOP# | I/O (s/t/s) 5V | Tri-state | **STOP:** Asserted by the target to request the master to stop the current transaction. This signal requires a 2.7 KΩ pullup resistor. |
| IDSEL | I 5V | | **INITIALIZATION DEVICE SELECT:** IDSEL is used as a chip select during configuration read and write transactions. |
| DEVSEL# | I/O (s/t/s) 5V | Tri-state | **DEVICE SELECT:** The MPIIX asserts DEVSEL# to claim a PCI transaction through positive decoding. This signal requires a 2.7 KΩ pullup resistor. |
| PAR | O 5V | Tri-state | **CALCULATED PARITY SIGNAL:** PAR is "even" parity and is calculated on 36 bits—AD[31:0] plus C/BE[3:0]#. |
| SERR# | I 5V | | **SYSTEM ERROR:** SERR# can be pulsed active by any PCI device that detects a system error condition. Upon sampling SERR# active, the MPIIX can be programmed to generate a non-maskable interrupt (NMI) to the CPU. This signal requires a 2.7 KΩ pullup resistor. |
| PHOLD# | O 5V | Tri-state | **PCI HOLD:** The MPIIX asserts this signal to request the PCI Bus. |
| PHLDA# | I 5V | | **PCI HOLD ACKNOWLEDGE:** The MTSC asserts this signal to grant the PCI Bus to the MPIIX. |
| REQ[A,B]# | I 5V | | **REQUEST A AND B:** PC/PCI requests for PCI DMA on a dedicated DMA channel or for PC/PCI DMA expansion. These signals should not be used for standard PCI Bus Masters. |
| GNT[A,B]# | O 5V | Tri-state | **GRANT A AND B:** PC/ PCI grants for PCI DMA on a dedicated DMA channel or for PC/PCI DMA expansion. These signals should not be used for standard PCI Bus Masters. |
| CLKRUN# | I/O 5V | Tri-state | **CLOCK RUN:** CLKRUN# is an asynchronous request to start the PCI clock. This signal also indicates PCI clock status. |
| PCIRST# | O 5V | Low | **PCI RESET:** See System Clock and Reset Signal section. |

## 2.2. IDE Interface Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| **DD[15:8]**/ SA[15:8]/ SD[15:8] | **I/O** O 5V TTL 8mA | Undefined | **DISK DATA**: These signals directly drive the corresponding signals on the IDE connector. In addition, these signals are externally buffered to produce the SA[15:8] signals (see separate descriptions). |
| **DD[7:0]**/ SD[7:0] | **I/O** I/O 5V TTL 8mA | Tri-state | **DISK DATA**: These signals directly drive the corresponding signals on the IDE connector. In addition, these signals are externally buffered to produce the SD[7:0] signals (see separate descriptions). |
| DIOR# | O 5V TTL 8mA | High | **DISK I/O READ**: This signal directly drives the corresponding signal on the IDE connector. |
| DIOW# | O 5V TTL 8mA | High | **DISK I/O WRITE**: This signal directly drives the corresponding signal on the IDE connector. |
| IORDY | I 5V pu8KΩ | | **IO CHANNEL READY**: This input signal is directly driven by the corresponding signal on the IDE connector. |
| **DA[2:0]/** SA[2:0] | O 5V TTL 8mA | Undefined | **DISK ADDRESS:** These address signals directly drive the DA[2:0] signals on the IDE connector and are used to indicate which byte in the ATA command block or control block is being addressed. These pins are multiplexed with SA[2:0]. |
| **DCS1#, DCS3#** / SA7,SA6 | O 5V TTL 8mA | Undefined | **DISK CHIP SELECTS:** DCS1# controls the ATA command register block and corresponds to CS1FX# on the IDE connector. DCS3# controls the ATA control register block and corresponds to CS3FX# on the IDE connector. These pins are multiplexed with SA[7,6]. |
| **DOE#** / SMOUT5 | O 5V TTL 4mA | High | **DISK OUTPUT ENABLE:** This signal controls the OE# of the IDE isolation buffers. SMOUT5 is configured to enable this function via the SMOUT Control Register. |

## 2.3. Extended I/O Bus Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| SYSCLK | O<br>5V<br>TTL<br>8mA | Active | **SYSTEM CLOCK:** SYSCLK is the reference clock for the Extended I/O Bus and drives the bus directly. SYSCLK is generated by dividing PCICLK by 3 or 4. The SYSCLK frequencies supported are 6.25 MHz, 7.5 MHz and 8.33 MHz. SYSCLK is a divided down version of PCICLK.<br><br>**Hardware strapping option**<br>SYSCLK is tri-stated when PWROK is negated. The value of SYSCLK is sampled on the assertion of PWROK: If sampled low, the ISA clock divisor is 3 (for 25 MHz PCI). Otherwise, the divisor is 4 (for 30 MHz PCI). The default value (divide-by-4) is determined by an internal pull-up resistor (50 KΩ). This pullup is disabled after reset. |
| IOCHRDY | I<br>5V<br>pu8KΩ | | **I/O CHANNEL READY:** Resources on the Extended I/O Bus negate IOCHRDY to indicate that additional time (wait-states) is required to complete the cycle. This signal is normally high. IOCHRDY is an input when the MPIIX owns the Extended I/O Bus and the CPU or a PCI agent is accessing an Extended I/O slave or during DMA transfers. |
| IOR# | O<br>5V<br>TTL<br>8mA | High | **I/O READ:** IOR# is the command to an Extended I/O Bus slave device that the slave may drive data on the Extended I/O data bus (SD[15:0]). |
| IOW# | O<br>5V<br>TTL<br>8mA | High | **I/O WRITE:** IOW# is the command to an Extended I/O Bus slave device that the slave may latch data from the Extended I/O data bus (SD[15:0]). |
| **SA[7:0]**,<br>DCS1#,<br>DCS3#,<br>DA[2:0]<br><br>**SA[15:8]/**<br>DD[15:8]/<br>SD[15:8]<br><br>**SA[17,16]**,<br>PCS#,<br>RTCCS# | O<br>5V<br>TTL<br>8mA<br><br>O<br><br>O<br>5V | Undefined<br><br>Undefined<br><br>Undefined | **SYSTEM ADDRESS BUS:** These address output signals define the selection with the granularity of one byte. For I/O accesses, only SA[15:0] are used. SA[17:0] are outputs during memory cycles to the Extended I/O Bus BIOS range. SA[17:0] are at an unknown state during PCIRST#. SA[15:0] are driven to 0 durintg DMA cycles to the Extended I/O Bus. SA[17:16] are driven to 1 following PCIRST# and during DMA cycles to the Extended I/O Bus. |
| **SD[15:8]/**<br>DD[15:8]/<br>SA[15:8] | I/O<br>5V | Undefined | **SYSTEM DATA BUS:** SD[15:8] provide the higher byte of the data path to DMA devices residing on the Extended I/O Bus. SD[15:8] are not available to memory or I/O devices on the Extended I/O Bus. |
| MEMR# | O<br>5V<br>TTL<br>8mA | High | **MEMORY READ:** MEMR# is the command to the BIOS memory that it may drive data onto the Extended I/O data bus. |

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| MEMW# | O<br>5V<br>TTL<br>8mA | High | **MEMORY WRITE:** MEMW# is the command to the BIOS memory that it may latch data from the Extended I/O data bus. |
| ZEROWS# | I<br>5V<br>st<br>pu8KΩ<br>TTL | | **ZERO WAIT STATES:** An Extended I/O Bus slave asserts ZEROWS# after its address and command signals have been decoded to indicate that the current cycle can be shortened. An 8-bit ISA memory cycle can be reduced to three SYSCLKs. |
| **SD[7:0]**/<br>DD[7:0] | I/O<br>5V<br>pu8KΩ<br>TTL<br>8mA | Tri-state | **SYSTEM DATA:** SD[7:0] provide the 8-bit data path for devices residing on the Extended I/O Bus. The MPIIX tri-states these signals during PCIRST#. |
| SDIR | O<br>5V<br>TTL<br>4mA | Low | **SYSTEM ADDRESS TRANSCEIVER DIRECTION**: This signal controls the direction of the '245 transceivers that interface the DD[15:0] signals to the SA[15:8] and SD[7:0] signals. Default condition is high (transmit). |

## 2.4. Motherboard I/O Device Interface Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| SA17/<br>**PCS#** | O<br>5V<br>TTL<br>8mA | Undefined | **PROGRAMMABLE CHIP SELECT.** PCS# is asserted for Extended I/O Bus I/O cycles that are generated by PCI masters, if the access is in the address range programmed into the PCSC Register. The Extended I/O Bus buffer signals are enabled when the chip select is asserted (i.e., it is assumed that the peripheral that is selected via this pin resides on the Extended I/O Bus). PCS# can be used to control the isolation buffer to the Plug-n-Play port isolation buffer. |
| BIOSCS# | O<br>5V<br>TTL<br>4mA | Undefined | **BIOS CHIP SELECT:** BIOSCS# is asserted during read or write accesses to BIOS. BIOSCS# is driven combinatorially from the Extended I/O Bus addresses SA[17:0], except during DMA. During DMA cycles, BIOSCS# is not generated. |
| KBCS# | O<br>5V<br>TTL<br>4mA | Undefined | **KEYBOARD CONTROLLER CHIP SELECT:** KBCS# is asserted during I/O read or write accesses to KBC locations 60h, 62h, 64h, and 66h. For DMA cycles, KBCS# is never asserted. |
| SA16/<br>**RTCCS#** | O<br>5V<br>TTL<br>8mA | Undefined | **REAL TIME CLOCK CHIP SELECT:** RTCCS# is asserted during read or write accesses to RTC location 71h, 73h, 75h, and 77h. RTCCS# can be tied to a pair of external OR gates to generate the real time clock read and write command signals. |

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| **RTCALE**/ SMOUT4 | O 5V TTL 4mA | High | **REAL TIME CLOCK ADDRESS LATCH:** RTCALE is used to latch the appropriate memory address into the RTC. A write to port 70h, 72h, 74h, or 76h with the appropriate RTC memory address that will be written to or read from, causes RTCALE to be asserted. RTCALE is asserted based on IOW# falling and remains asserted for two SYSCLKs. |
| SPKR | O 5V TTL 8mA | Low | **SPEAKER DRIVE:** The SPKR signal is the output of counter 2. |
| OSC | I 5V TTL | | **OSCILLATOR:** OSC is the 14.31818 MHz ISA clock signal. It is used by the internal 8254 Timer. |
| FERR# | I 3.3V pu50KΩ | | **NUMERIC COPROCESSOR ERROR:** This signal is tied to the coprocessor error signal on the CPU. IGNNE# is only used if the MPIIX coprocessor error reporting function is enabled in the FDC Enable Register. If FERR# is asserted, the MPIIX generates an internal IRQ13 to its interrupt controller unit. The MPIIX then asserts the INTR output to the CPU. FERR# is also used to gate the IGNNE# signal to ensure that IGNNE# is not asserted to the CPU unless FERR# is active. FERR# has a weak internal pull-up used to ensure a high level when the coprocessor error function is disabled. |
| IGNNE# | od 3.3V | High | **IGNORE ERROR:** This signal is connected to the ignore error pin on the CPU. IGNNE# is only used if the MPIIX coprocessor error reporting function is enabled in the FDC Enable Register. If FERR# is asserted, indicating a coprocessor error, a write to the Coprocessor Error Register (F0h) causes the IGNNE# to be asserted. IGNNE# remains asserted until FERR# is negated. If FERR# is not asserted when the Coprocessor Error Register is written, the IGNNE# signal is not asserted. |
| ALTA20M | O 5V TTL 4mA | Low | **Alternative A20 MASK:** This MPIIX output is externally OR'd with the A20gate from the KBC to generate A20M# to the CPU. A20M# is used to emulate the 1 Mbyte wrap-around. |

## 2.5. DMA Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| **MDRQ[2:0]/** EXTEVNT# | I 5V pd50K Ω | | **MOTHERBOARD DEVICE DMA REQUEST:** These signals can be connected internally to any of DREQ[3:0,7:5]. Each pair of request/acknowledge signals is controlled by a separate register. |
| **MDAK[2:0]#/** PAD# | O 5V 4mA | High | **MOTHERBOARD DEVICE DMA ACKNOWLEDGE:** These signals can be connected internally to any of DACK[3:0,7:5]. Each pair of request/acknowledge signals is controlled by a separate register. MDAK1 or MDAK2 or both can be enabled to re-load the Local Standby Timer for the audio device. |
| DREQ2 | I 5V pu50K Ω TTL | | **DMA REQUEST 2:** DREQ2 is used by the floppy disk controller to request DMA service from the MPIIX's DMA controller. All inactive to active edges are assumed to be asynchronous. The request must remain active until the appropriate DACK2# signal is asserted. |
| DACK2# | O 5V TTL 4mA | High | **DMA ACKNOWLEDGE 2:** DACK2# indicates that a request for DMA service has been granted by the MPIIX. This line should be used to decode the DMA slave device with the IOR# or IOW# line to indicate selection. |
| TC | O 5V TTL 4mA | Tri-state | **TERMINAL COUNT:** The MPIIX asserts TC to DMA slaves as a terminal count indicator. MPIIX asserts TC after a new address has been output, if the byte count expires with that transfer. When all the DMA channels are not in use, TC is negated (low). |

## 2.6. Interrupt Controller Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| IRQ[15,14, 11:9,7:3,1] | I 5V pu8KΩ TTL | | **INTERRUPT REQUEST:** The IRQ signals provide both system board components and docking station Extended I/O Bus I/O devices with a mechanism for asynchronously interrupting the CPU. The assertion mode of these inputs depends on the programming of the two ELCR Registers.<br><br>IRQ1 (as well as IRQ[8#,2,0] and the internal IRQ13) are not programmable through the ELCR Registers. These IRQs are always active high edge triggered. An internal flip-flop latches a low-to-high transition on IRQ1. The MPIIX continues to generate an internal IRQ1 to the 8259 core until a PCIRST# or an I/O read access to port 60h.<br><br>An active IRQ input must remain asserted until after the interrupt is acknowledged. If the IRQ is negated before this time, a DEFAULT IRQ7 occurs when the CPU acknowledges the interrupt. |

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| IRQ8# | I<br>5/3V<br>pu8KΩ<br>CMOS | | **INTERRUPT REQUEST EIGHT SIGNAL:** IRQ8# is always an active low edge triggered interrupt input (i.e. this interrupt can not be modified by software). This signal is monitored by the low power 'Resume Well' circuitry during suspend.<br><br>IRQ8# must remain asserted until after the interrupt is acknowledged. If the input goes inactive before this time, a DEFAULT IRQ7 will occur when the CPU acknowledges the interrupt. |
| IRQ12/M | I<br>5V<br>pu8KΩ<br>TTL | | **INTERRUPT REQUEST / MOUSE INTERRUPT:** In addition to providing the standard interrupt function (see IRQ[15,14,11:9,7:3,1] signal description), this pin can be programmed (via the FDC Enable Register) to provide a mouse interrupt function.<br><br>When the mouse interrupt function is selected, a low-to-high transition on this signal is latched by the MPIIX and an INTR is generated to the CPU as IRQ12. An internal IRQ12 interrupt continues to be generated until a PCIRST# or an I/O read access to address 60h. After a PCIRST#, this signal provides the standard IRQ12 function. |
| MIRQ | I<br>5V | | **MOTHERBOARD DEVICE INTERRUPT REQUEST**: The MIRQ signal can be internally connected to interrupts IRQ[15,14,12:9,7:3]. If MIRQ line and PIRQx# are steered to the same interrupt, the device connected to the MIRQx should produce active high, level interrupts.<br><br>If the MIRQ line is steered to a given IRQ input to the internal 8259, the corresponding IRQ is masked, unless the Route Control register is programmed to allow the interrupts to be shared. This should only be done if the device connected to the MIRQ line and the device connected to the IRQ line both produce active high, level interrupts. |
| PIRQ[A,B]# | I<br>5V<br>pu8KΩ<br>TTL | | **PROGRAMMABLE INTERRUPT REQUEST:** The PIRQx# signals can be shared with interrupts IRQ[15,14,12:9,7:3] as described in the Interrupt Steering section. Each PIRQx# line has a separate Route Control Register. |
| INTR | O<br>3.3V<br>TTL<br>4mA | Low | **CPU INTERRUPT:** INTR is driven by the MPIIX to signal the CPU that an interrupt request is pending and needs to be serviced. The interrupt controller must be programmed following PCIRST# to ensure that INTR is at a known state. |
| NMI | od<br>3.3V<br>TTL<br>4mA | | **NON-MASKABLE INTERRUPT:** NMI is used to force a non-maskable interrupt to the CPU. The MPIIX generates an NMI when SERR# is asserted, depending on how the NMI Status and Control Register is programmed. |

## 2.7. System Power Management (SMM) Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| SMI# | O<br>3.3V<br>TTL<br>4mA | High | **SYSTEM MANAGEMENT INTERRUPT:** SMI# is an active low synchronous output that is asserted by the MPIIX in response to one of many enabled hardware or software events. During CPU Reset (INIT and CPURST), this signal is negated. |
| STPCLK# | O<br>3.3V<br>TTL<br>4mA | High | **STOP CLOCK:** STPCLK# is an active low synchronous output that is asserted by the MPIIX in response to one of many hardware or software events. STPCLK# connects directly to the CPU and is synchronous to PCICLK. |
| SUSTAT# | O<br>5/3V<br>CMOS<br>4mA | High | **SUSPEND STATUS:** This output signal is used to switch off power to all non-critical devices during suspend. Activation of this signal is typically the last step in the SMM code. |
| SYSACT# | I<br>5V<br>pu50KΩ<br>TTL | | **SYSTEM ACTIVITY:** This input signal can be used by system devices such as bridge chips to indicate system activity that is not visible to the MPIIX power management logic. This signal, if enabled through setup software, can be used by MPIIX to prevent the system from entering an idle state. |
| SRBTN# | I<br>5/3V<br>CMOS | | **SUSPEND RESUME BUTTON:** This signal can be enabled to generates an SMI# request. SRBTN# is monitored by the low power “Resume Well” circuitry during suspend. This signal must always be driven to a valid logic level. |
| BATLOW# | I<br>5/3V<br>CMOS | | **BATTERY LOW:** BATLOW# Indicates that battery power is low. Assertion of this signal triggers an SMI, if enabled. This signal is monitored by the low power “Resume Well” circuitry during suspend. MPIIX can be programmed to prevent a resume operation when the BATLOW# signal is active. This signal must always be driven to a valid logic level. Once asserted, this signal must remain asserted until SMI# is generated. |
| RSMRST# | I<br>5/3V<br>CMOS | | **RESUME RESET:** This signal acts as a reset to the low power “Resume Well” circuitry. This signal must always be driven to a valid logic level. |
| **SMOUT[5:0]**/<br>DOE#,<br>RTCALE | O<br>5V<br>TTL<br>4mA | High | **SYSTEM MANAGEMENT OUTPUT ENABLES:** These six programmable outputs can be connected to control the power circuits for various devices in the system. SMOUT5 can be configured to generate a DISK Output Enable. SMOUT4 can be configured to generate RTCALE. |
| COMRI# | I<br>5/3V<br>pu50KΩ<br>CMOS | | **COM RING INDICATE:** A modem connected to the COM port will asserts this signal to wake up a suspended system. This signal is monitored by the low power “Resume Well” circuitry during suspend. |

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| EXTSMI# | I<br>st<br>5/3V<br>pu8KΩ<br>CMOS | | **EXTERNAL SYSTEM MANAGEMENT INTERRUPT:** EXTSMI# is a falling edge triggered input to the MPIIX indicating that an external device is requesting the system to enter SMM mode. When enabled, a low level on EXTSMI# will result in the assertion of the SMI# signal. EXTSMI# is an asynchronous input and should be asserted for a minimum of 32 µsec. |
| **EXTEVNT#**/<br>MDRQ2 | I<br>5V<br>pd50KΩ<br>TTL | | **EXTERNAL EVENT (EXTEVNT#):** The EXTEVNT# signal allows events detected by the external logic to be used as BSTCLK Events or CLKTHL Break Events. MDRQ[2] is multiplexed with the EXTEVNT# signal. A confguration bit selects which signal is enabled on the pin. The power on default is the MDRQ[2] signal. |
| **PAD# /**<br>MDAK2# | O<br>5V<br>TTL<br>4mA | High | **Peripheral Access Decode:** The PAD# signal is asserted by the MPIIX when a PCI memory or I/O address is decoded to be in the same address range as defined by the Peripheral Access Detect Tables and enabled in the Peripheral Access Decode Enable register. MPIIX does not have to be the target of the PCI cycle for the PAD# signal to be asserted. MDAK2# is multiplexed with the PAD# signal. A configuration bit selects which signal is enabled on the pin. The power on default is the MDAK2# signal. |

## 2.8. System Clock And Reset Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| HCLK | I<br>5V<br>TTL | | **HOST CLOCK:** Main system clock used to create clocks for PCI, MTSC, MTDP, and external cache. |
| PCICLK | I<br>5V<br>TTL | | **PCI CLOCK:** PCICLK provides timing for all transactions on the PCI Bus. All other PCI signals are sampled on the rising edge of PCICLK, and all timing parameters are defined with respect to this edge. PCI frequencies of 25–33 MHz are supported. |
| HCLKO | O<br>3.3V<br>TTL<br>4mA | Active | **HOST CLOCK OUT:** Must be buffered to provide CPU, MTSC, TDP, and external L2 cache clocks. |
| PCICLKO | O<br>5V<br>TTL<br>4mA | Active | **PCI CLOCK OUTPUT:** Synchronous divide-by-2 of HCLK. Must be buffered to provide fully loadable PCI clock. |
| RTCCLK | I<br>5/3V<br>CMOS | | **REAL TIME CLOCK INPUT.** |

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| RTCCLKO | O<br>5/3V<br>CMOS<br>4mA | Active | **REAL TIME CLOCK OUTPUT:** Gated RTCCLK to MTSC for suspend refresh operation. |
| SYSCLK | O<br>5V | | **SYSTEM CLOCK:** See Extended I/O Bus Interface Section. |
| PWROK | I<br>5/3V<br>st<br>CMOS | | **POWER OK:** When asserted, PWROK is an indication to the MPIIX that power and PCICLK have been stable for at least 1 ms. PWROK can be driven asynchronously. When PWROK is negated, the MPIIX asserts CPURST, PCIRST# and RSTDRV. When PWROK is asserted, the MPIIX negates CPURST, PCIRST#, and RSTDRV. |
| CPURST | O<br>3.3V<br>TTL<br>4mA | High | **CPU RESET:** The MPIIX asserts CPURST to reset the CPU. The MPIIX asserts CPURST during power-up and when a hard reset sequence is initiated through the RC Register. CPURST is driven synchronously to the rising edge of PCICLK. If a hard reset is initiated through the RC register, the MPIIX resets it's internal registers to the default state. |
| PCIRST# | O<br>5V | Low | **PCI RESET:** The MPIIX asserts PCIRST# to reset devices that reside on the PCI Bus. The MPIIX asserts PCIRST# during power-up and when a hard reset sequence is initiated through the RC Register. PCIRST# is driven inactive a minimum of 1ms after PWROK is driven active. PCIRST# is driven active for a minimum of 1ms when initiated through the RC Register. PCIRST# is driven asynchronously relative to PCICLK. |
| INIT | O<br>3.3V<br>TTL<br>4mA | High | **INITIALIZATION:** The MPIIX asserts INIT if it detects a shut down special cycle on the PCI Bus or if a soft reset is initiated via the RC Register (0CF9h). |
| RSTDRV | O<br>5V<br>TTL<br>8mA | High | **RESET DRIVE:** The MPIIX asserts this signal during a hard reset and during power-up to reset Extended I/O Bus devices. RSTDRV is also asserted for a minimum of 1 ms if a hard reset has been programmed in the RC Register. |

## 2.9. Test Signals

| Signal Name | Type | PCIRST# | Description |
|---|---|---|---|
| TESTIN# | I<br>3.3V | | **TEST INPUT:** The Test signal is used to tri-state all of the MPIIX outputs.This input is sampled on the assertion of PWROK. |

## 3.0. REGISTER DESCRIPTION

There are two groups of MPIIX internal registers—PCI Configuration Registers and ISA Compatible Registers. These registers are discussed in this section.

Some of the MPIIX registers contain reserved bits. Software must deal correctly with fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bit positions are preserved. That is, the values of reserved bit positions must first be read, merged with the new values for other bit positions and then written back.

In addition to reserved bits within a register, the MPIIX contains address locations in the PCI configuration space that are marked "Reserved" (Table 3.1). The MPIIX responds to accesses to these address locations by completing the Host cycle. Software should not write to reserved MPIIX configuration locations in the device-specific region (above address offset 3Fh).

During a hard reset the MPIIX sets its internal registers to predetermined **default** states. The default values are indicated in the individual register descriptions.

The following notation is used to describe register access attributes:

**RO** **Read Only**. If a register is read only, writes have no effect.
**WO** **Write Only**. If a register is write only, reads have no effect.
**R/W** **Read/Write**. A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.
**R/WC** **Read/Write Clear**. A register bit with this attribute can be read and written. However, a write of a 1 clears (sets to 0) the corresponding bit and a write of a 0 has no effect.

## 3.1. Register Access

Table 1 and Table 2 show the I/O assignments for the PCI Configuration Registers and ISA Compatible Registers, respectively. CPU and PCI masters have access to all MPIIX internal registers.

The MPIIX is a single-function device on the PCI Bus. The MPIIX configuration registers are accessed through a mechanism defined for single-function PCI devies in compliance with the PCI Local Bus Specification, Revision 2.0. The configuration registers can only be accessed by PCI masters. For configuration cycles, DEVSEL# is a function of IDSEL and AD[1:0]. DEVSEL# is selected during a configuration cycle only if IDSEL is active and both AD[1:0]=00. IDSEL must be connected to AD12 (device #1). Configuration cycles that target functions 1 through 7 (AD[10:8]=001b through 111b) are ignored (DEVSEL# is not asserted).

The ISA Compatible Registers (e.g., DMA registers, timer/counter registers, interrupt registers, X-Bus registers, and NMI registers) are accessed through I/O space in the normal fashion. PCI master accesses to the ISA Compatible Registers can be 8, 16, 24, or 32 bits. The MPIIX will only respond to the least significant byte (see the IDE section in the Functional Description section for 16-bit IDE register response). On writes the other bytes will not be loaded and on reads the other bytes have invalid data.

There are two power management registers located in normal I/O space. These registers are accessed (by PCI Bus masters) with 8-bit accesses. The other power management registers are located in PCI configuration space.

In general, accesses from CPU or PCI masters to the internal MPIIX registers are not broadcast to the Extended I/O bus. Exceptions to this general rule are read and write accesses to locations 60h, 70–76h, and F0h, and write accesses to ports 80h, 84–86h, 88h, 8C–8Eh. These accesses are broadcast to the Extended I/O Bus.

**Table 1. PCI Configuration Registers**

| Configuration Offset | Mnemonic | Register | Register Access |
|---|---|---|---|
| 00–01h | VID | Vendor Identification | RO |
| 02–03h | DID | Device Identification | RO |
| 04–05h | COM | Command | R/W |
| 06–07h | DS | Device Status | R/WC |
| 08h | RID | Revision Identification | RO |
| 09–0Bh | CLASSC | Class Code | RO |
| 0C–0Dh | — | Reserved | — |
| 0Eh | HEDT | Header Type | RO |
| 0F–48h | — | Reserved | — |
| 49h | SPPE | Serial and Parallel Port Enable | R/W |
| 4A–4Bh | — | Reserved | — |
| 4Ch | ECRT | Extended I/O Controller Recovery Timer | R/W |
| 4Dh | — | Reserved | — |
| 4Eh | BIOSE | BIOS Enable | R/W |
| 4Fh | FDCE | FDC Enable | R/W |
| 50–5Fh | — | Reserved | — |
| 60–61h | PIRQRC[A,B] | PIRQ[A,B]# Route Control | R/W |
| 62–69h | — | Reserved | — |
| 6A–6Bh | MSTAT | Miscellaneous Status | RO |
| 6C–6Dh | IDETIM | IDE Timing Modes | R/W |
| 6E–6Fh | — | Reserved | — |
| 70h | MIRQRC | Motherboard IRQ Route Control | R/W |
| 71–75h | — | Reserved | — |
| 76–78h | MDMARC | Motherboard DMA Route Control | R/W |
| 79–7Dh | — | Reserved | — |
| 7Eh | AUDIOE | Audio Enable | R/W |
| 7Fh | DMADS | DMA CH[7:5] Address Size | R/W |
| 80h | PCIDMAE | PCI DMA Enable | R/W |
| 81–87h | — | Reserved | — |
| 88h | PCIDMAA | PCI DMA and PCI DMA Expansion A | R/W |
| 89h | PCIDMAB | PCI DMA and PCI DMA Expansion B | R/W |
| 8A–8Dh | PMAC[1:0] | Programmable Memory Address Control | R/W |
| 8E–8Fh | PMAM[1:0] | Programmable Memory Address Mask | R/W |
| 90h | PARE | Programmable Address Range Enable | R/W |
| 91h | — | Reserved | — |
| 92–93h | PCSC | Programmable Chip Select Control | R/W |
| 94–95h | PAC1 | Programmable Address Control 1 | R/W |
| 96–97h | PAC2 | Programmable Address Control 2 | R/W |
| 98–99h | PAC3 | Programmable Address Control 3 | R/W |
| 9Ah | PAMA | Programmable Address Mask A | R/W |
| 9Bh | PAMB | Programmable Address Mask B | R/W |

| Configuration Offset | Mnemonic | Register | Register Access |
|---|---|---|---|
| 9C–9Dh | IOCA | I/O Configuration Address | R/W |
| 9E–9Fh | — | Reserved | — |
| A0h–A1h | PAC4 | Programmable Address Control 4 | R/W |
| A2h–A3h | PAC5 | Programmable Address Control 5 | R/W |
| A4h | PAMC | Programmable Address Mask C | R/W |
| A5–A7h | PADE[2:0] | Peripheral Access Detect Enable | R/W |
| A8h–A9h | LTADEV3 | Local Trap Address For Device 3 | R/W |
| AAh | LTMDEV3 | Local Trap Mask For Device 3 | R/W |
| ABh | LTSMIE | Local Trap SMI Enable | R/W |
| AC–ADh | — | Reserved | — |
| AEh | LTSMIS | Local Trap SMI Status | R/W |
| AFh | — | Reserved | — |
| B0h | LSBSMIE | Local Standby SMI Enable | R/W |
| B1h | LSBTRE | Local Standby Timer Reload Enable | R/W |
| B2h | LSBSMIS | Local Standby SMI Status | R/W |
| B3h | — | Reserved | — |
| B4h | LSBTIDE | Local Standby Timer IDE Idle | R/W |
| B5h | LSBTAUD | Local Standby Timer Audio Idle | R/W |
| B6h | LSBTCOM | Local Standby Timer COM Idle | R/W |
| B7h | — | Reserved | — |
| B8h | LSBTDEV1 | Local Standby Timer Device 1 Idle | R/W |
| B9h | LSBTDEV2 | Local Standby Timer Device 2 Idle | R/W |
| BAh | LSBTDEV3 | Local Standby Timer Device 3 Idle | R/W |
| BBh | — | Reserved | — |
| BCh | SESMIT | Software/EXTSMI# SMI Delay Timer | R/W |
| BDh | SUSSMIT | Suspend SMI Delay Timer | R/W |
| BEh | GSBTMR | Global Standby Timer | R/W |
| BFh | CLKTSBYT | Clock Throttle Standby Timer | R/W |
| C0h | SYSMGNTC | System Mangement Control | R/W |
| C1h | SYSSMIE | System SMI Enable | R/W |
| C2h | MISCSMIE | Miscellaneous SMI Enable | R/W |
| C3h | GSMIE | Global SMI Enable | R/W |
| C4–C5h | — | Reserved | — |
| C6h | SYSSMIS | System SMI Status | R/W |
| C7h | MISCSMIS | Miscellaneous SMI Status | R/W |
| C8h | GSMIS | Global SMI Status | R/W |
| C9–CBh | — | Reserved | — |
| CCh | SUSRSMC1 | Suspend/Resume Control 1 | R/W |
| CDh | SUSRSMC2 | Suspend/Resume Control 2 | R/W |
| CEh | SMOUTC | SMOUT Control | R/W |
| CFh | — | Reserved | — |

| Configuration Offset | Mnemonic | Register | Register Access |
|---|---|---|---|
| D0h | SYSEVNTE0 | System Event Enable 0 | R/W |
| D1h | SYSEVNTE1 | System Event Enable 1 | R/W |
| D2h | SYSEVNTE2 | System Event Enable 2 | R/W |
| D3h | BSTCLKT | Burst Count Timer | R/W |
| D4h | CLKC | Clock Control | R/W |
| D5h | — | Reserved | — |
| D6h | STPCLKLT | STPCLK# Low Timer | R/W |
| D7h | STPCLKHT | STPCLK# High Timer | R/W |
| D8h | STPBRKE0 | Stop Break Event Enable 0 | R/W |
| D9h | STPBRKE1 | Stop Break Event Enable 1 | R/W |
| DAh | STPBRKE2 | Stop Break Event Enable 2 | R/W |
| DB–DFh | — | Reserved | — |
| E0h | SHDW | Shadow Register | see description |
| E1–E3h | — | Reserved | — |
| E4–EAh | BSTCLKEE[6:0] | Burst Clock Event Enable | R/W |
| EBh | — | Reserved | — |
| EC–F2h | CLKTHLBRKEE[6:0] | Clock Throttle Break Event Enable | R/W |
| F3–FFh | — | Reserved | — |

**Table 2. ISA-Compatible Registers**

| Address | Address | | | | Type | Name |
|---|---|---|---|---|---|---|
| | FEDC | BA98 | 7654 | 3210 | | |
| 0000h | 0000 | 0000 | 000x | 0000 | R/W | DMA1 CH0 Base and Current Address |
| 0001h | 0000 | 0000 | 000x | 0001 | R/W | DMA1 CH0 Base and Current Count |
| 0002h | 0000 | 0000 | 000x | 0010 | R/W | DMA1 CH1 Base and Current Address |
| 0003h | 0000 | 0000 | 000x | 0011 | R/W | DMA1 CH1 Base and Current Count |
| 0004h | 0000 | 0000 | 000x | 0100 | R/W | DMA1 CH2 Base and Current Address |
| 0005h | 0000 | 0000 | 000x | 0101 | R/W | DMA1 CH2 Base and Current Count |
| 0006h | 0000 | 0000 | 000x | 0110 | R/W | DMA1 CH3 Base and Current Address |
| 0007h | 0000 | 0000 | 000x | 0111 | R/W | DMA1 CH3 Base and Current Count |
| 0008h | 0000 | 0000 | 000x | 1000 | R/W | DMA1 Status(r) Command(w) register |
| 0009h | 0000 | 0000 | 000x | 1001 | wo | DMA1 Write Request register |
| 000Ah | 0000 | 0000 | 000x | 1010 | wo | DMA1 Write Single Mask Bit |
| 000Bh | 0000 | 0000 | 000x | 1011 | wo | DMA1 Write Mode register |
| 000Ch | 0000 | 0000 | 000x | 1100 | wo | DMA1 Clear Byte Pointer |
| 000Dh | 0000 | 0000 | 000x | 1101 | wo | DMA1 Master Clear |
| 000Eh | 0000 | 0000 | 000x | 1110 | wo | DMA1 Clear Mask register |
| 000Fh | 0000 | 0000 | 000x | 1111 | R/W | DMA1 Read/Write All Mask Register Bits |
| 0020h | 0000 | 0000 | 001x | xx00 | R/W | INT 1 Control register |
| 0021h | 0000 | 0000 | 001x | xx01 | R/W | INT 1 Mask register |
| 0040h | 0000 | 0000 | 010x | 0000 | R/W | Timer Counter 1 – Counter 0 Count |

| Address | Address | | | | Type | Name |
|---|---|---|---|---|---|---|
| | FEDC | BA98 | 7654 | 3210 | | |
| 0041h | 0000 | 0000 | 010x | 0001 | R/W | Timer Counter 1 – Counter 1 Count |
| 0042h | 0000 | 0000 | 010x | 0010 | R/W | Timer Counter 1 – Counter 2 Count |
| 0043h | 0000 | 0000 | 010x | 0011 | wo | Timer Counter 1 Command Mode register |
| 0060h[1] | 0000 | 0000 | 0110 | 000 | r | Reset XBus IRQ12/M and IRQ1 |
| 0061h | 0000 | 0000 | 0110 | 0001 | R/W | NMI Status and Control |
| 0070h[1] | 0000 | 0000 | 0111 | 0xx0 | wo | CMOS RAM Address and NMI Mask reg. |
| 0080h[2] | 0000 | 0000 | 100x | 0000 | R/W | DMA Page Register (Reserved) |
| 0081h | 0000 | 0000 | 100x | 0001 | R/W | DMA Channel 2 Page register |
| 0082h | 0000 | 0000 | 1000 | 0010 | R/W | DMA Channel 3 Page register |
| 0083h | 0000 | 0000 | 100x | 0011 | R/W | DMA Channel 1 Page register |
| 0084h[2] | 0000 | 0000 | 100x | 0100 | R/W | DMA Page Register (Reserved) |
| 0085h[2] | 0000 | 0000 | 100x | 0101 | R/W | DMA Page Register (Reserved) |
| 0086h[2] | 0000 | 0000 | 100x | 0110 | R/W | DMA Page Register (Reserved) |
| 0087h | 0000 | 0000 | 100x | 0111 | R/W | DMA Channel 0 Page register |
| 0088h[2] | 0000 | 0000 | 100x | 0100 | R/W | DMA Page Register (Reserved) |
| 0089h | 0000 | 0000 | 100x | 1001 | R/W | DMA Channel 6 Page register |
| 008Ah | 0000 | 0000 | 100x | 1010 | R/W | DMA Channel 7 Page register |
| 008Bh | 0000 | 0000 | 100x | 1011 | R/W | DMA Channel 5 Page register |
| 008Ch[2] | 0000 | 0000 | 100x | 1100 | R/W | DMA Page Register (Reserved) |
| 008Dh[2] | 0000 | 0000 | 100x | 1101 | R/W | DMA Page Register (Reserved) |
| 008Eh[2] | 0000 | 0000 | 100x | 1110 | R/W | DMA Page Register (Reserved) |
| 008Fh | 0000 | 0000 | 100x | 1111 | R/W | DMA low page Register Refresh |
| 0092h | 0000 | 0000 | 1001 | 0010 | R/W | System Control Port |
| 00A0h | 0000 | 0000 | 101x | xx00 | R/W | INT 2 Control register |
| 00A1h | 0000 | 0000 | 101x | xx01 | R/W | INT 2 Mask register |
| 00B2h | 0000 | 0000 | 1011 | 0010 | R/W | Advanced Power Management Control Port |
| 00B3h | 0000 | 0000 | 1011 | 0011 | R/W | Advanced Power Management Status Port |
| 00C0h | 0000 | 0000 | 1100 | 000x | R/W | DMA2 CH0 Base and Current Address |
| 00C2h | 0000 | 0000 | 1100 | 001x | R/W | DMA2 CH0 Base and Current Count |
| 00C4h | 0000 | 0000 | 1100 | 010x | R/W | DMA2 CH1 Base and Current Address |
| 00C6h | 0000 | 0000 | 1100 | 011x | R/W | DMA2 CH1 Base and Current Count |
| 00C8h | 0000 | 0000 | 1100 | 100x | R/W | DMA2 CH2 Base and Current Address |
| 00CAh | 0000 | 0000 | 1100 | 101x | R/W | DMA2 CH2 Base and Current Count |
| 00CCh | 0000 | 0000 | 1100 | 110x | R/W | DMA2 CH3 Base and Current Address |
| 00CEh | 0000 | 0000 | 1100 | 111x | R/W | DMA2 CH3 Base and Current Count |
| 00D0h | 0000 | 0000 | 1101 | 000x | R/W | DMA2 Status(r) Command(w) register |
| 00D2h | 0000 | 0000 | 1101 | 001x | wo | DMA2 Write Request register |
| 00D4h | 0000 | 0000 | 1101 | 010x | wo | DMA2 Write Single Mask Bit |
| 00D6h | 0000 | 0000 | 1101 | 011x | wo | DMA2 Write Mode register |
| 00D8h | 0000 | 0000 | 1101 | 100x | wo | DMA2 Clear Byte Pointer |
| 00DAh | 0000 | 0000 | 1101 | 101x | wo | DMA2 Master Clear |

| Address | Address | | | | Type | Name |
|---|---|---|---|---|---|---|
| | FEDC | BA98 | 7654 | 3210 | | |
| 00DCh | 0000 | 0000 | 1101 | 110x | wo | DMA2 Clear Mask register |
| 00DEh | 0000 | 0000 | 1101 | 111x | R/W | DMA2 Read/Write All Mask Register Bits |
| 00F0h[1] | 0000 | 0000 | 1111 | 0000 | wo | Coprocessor Error |
| 04D0h | 0000 | 0100 | 1101 | 0000 | R/W | INT-1 edge/level control register |
| 04D1h | 0000 | 0100 | 1101 | 0001 | R/W | INT-2 edge/level control register |
| 0CF9h | 0000 | 1100 | 1111 | 1001 | R/W | Control Register |

**Note:**
1. Accesses to these locations are always broadcast to the Extended I/O Bus.
2. Writes to these locations are always broadcast to the Extended I/O Bus.

## 3.2. PCI Configuration Registers

### 3.2.1. VID—VENDOR IDENTIFICATION REGISTER

Address Offset: 01–00h
Default Value: 8086h
Attribute: Read Only

The VID Register contains the vendor identification number. This register, along with the Device Identification Register, uniquely identifies any PCI device. Writes to this register have no effect.

| Bit | Description |
|---|---|
| 15:0 | **Vendor Identification Number.** This is a 16-bit value assigned to Intel. |

### 3.2.2. DID—DEVICE IDENTIFICATION REGISTER

Address Offset: 03–02h
Default Value: 1234h
Attribute: Read Only

The DID Register contains the device identification number. This register, along with the VID Register, define the MPIIX. Writes to this register have no effect.

| Bit | Description |
|---|---|
| 15:0 | **Device Identification Number.** This is a 16-bit value assigned to the MPIIX. |

### 3.2.3. COM—COMMAND REGISTER

Address Offset: 05–04h
Default Value: 0007h
Attribute: Read/Write

This 16-bit register provides basic control over the MPIIX's ability to respond to PCI cycles.

| Bit | Description |
|---|---|
| 15:10 | **Reserved.** Read as 0. |
| 9 | **Fast Back-to-Back Enable (FBE).** Reserved, read as 0. |
| 8 | **SERR# Enable.** Reserved, read as 0. |
| 7:5 | **Reserved.** Read as 0. |
| 4 | **Postable Memory Write Enable (PMWE).** This bit will always be read as a 0. |
| 3 | **Special Cycle Enable (SCE)**. 1=Enable (MPIIX recognizes PCI special cycles—shutdown and stop grant); 0=Disable (MPIIX ignores all PCI special cycles). |
| 2 | **Bus Master Enable (BME).** MPIIX does not support disabling its bus master capability. This bit always reads as 1. |
| 1 | **Memory Space Enable (MSE).** The MPIIX does not support disabling access to main memory. This bit is read as 1. |
| 0 | **I/O Space Enable (IOSE).** The MPIIX does not support disabling its response to PCI I/O cycles. This bit is read as 1. |

### 3.2.4. DS—DEVICE STATUS REGISTER

Address Offset: 06–07h
Default Value: 0280h
Attribute: Read/Write Clear

DSR is a 16-bit status register that reports the occurrence of a PCI master-abort by the MPIIX or a PCI target-abort when the MPIIX is a master. The register also indicates the MPIIX DEVSEL# signal timing.

| Bit | Description |
|---|---|
| 15 | **Parity Error (Not Implemented).** Read as 0. |
| 14 | **SERR# Status (Not Implemented).** Read as 0. |
| 13 | **Master-Abort Status (MA):** When the MPIIX, as a master, generates a master-abort, MA is set to a 1. Software sets MA to 0 by writing a 1 to this bit location. |
| 12 | **Received Target-Abort Status (RTA):** When the MPIIX is a master on the PCI Bus and receives a target-abort, this bit is set to a 1. Software resets RTA to 0 by writing a 1 to this bit location. |
| 11 | **Signaled Target-Abort Status (STA):** This bit is set when the MPIIX Extended I/O bus bridge function is targeted with a transaction that the MPIIX terminates with a target abort. Software resets STA to 0 by writing a 1 to this bit location. |

| Bit | Description |
|---|---|
| 10:9 | **DEVSEL# Timing Status (DEVT):** DEVT=01. The MPIIX always generates DEVSEL# with medium timing for Extended I/O functions. This DEVSEL# timing does not include configuration cycles. |
| 8 | **PERR# Response (Not Implemented).** Read as 0. |
| 7 | **Fast Back to Back—RO.** This bit is read as 1, indicating to the PCI Master that MPIIX, as a target, is capable of accepting fast back-to-back transactions. |
| 6:0 | **Reserved.** Read as 0s. |

### 3.2.5. RID—REVISION IDENTIFICATION REGISTER

Address Offset: 08h
Default Value: Refer to stepping information
Attribute: Read Only

This 8-bit register contains device stepping information. Writes to this register have no effect.

| Bit | Description |
|---|---|
| 7:0 | **Revision ID Byte:** The register is hardwired to the default value. |

### 3.2.6. CLASSC—CLASS CODE REGISTER

Address Offset: 09–0Bh
Default Value: 068000h
Attribute: Read Only

This register contains the device programming interface information related to the Sub-Class Code and Base Class Code definition for the MPIIX. This register also identifies the Base Class Code and the function sub-class in relation to the Base Class Code.

| Bit | Description |
|---|---|
| 23:16 | **Base Class Code (BASEC).** 06h=PCI Bridge device. |
| 15:8 | **Sub-Class Code (SCC)**. 01h=Other bridge device (ISA-like Extended I/O Bus). |
| 7:0 | **Programming Interface (PI).** 00h=No programming interface defined. |

### 3.2.7. HEDT—HEADER TYPE REGISTER

Address Offset: 0Eh
Default Value: 00h
Attribute: Read Only

The HEDT Register identifies the MPIIX as a single-function device.

| Bit | Description |
|---|---|
| 7:0 | **Device Type (DEVICET):** 00h=single-function device. |

### 3.2.8. SPPE—SERIAL & PARALLEL PORT ENABLE REGISTER

Address Offset: 49h
Default Value: 00h
Attribute: Read/Write

This register enables/disables accesses to the Serial Ports and Parallel Ports on the Extended I/O Bus.

| Bit | Description |
|---|---|
| 7 | **Reserved.** |
| 6 | **LPT3 Enable** 1=Enable (forward PCI I/O accesses to 0278–027Fh and 0678–067Bh to the Extended I/O Bus). 0=Disable (confine to PCI). |
| 5 | **LPT2 Enable.** 1=Enable (forward PCI I/O accesses to 0378–037Fh and 0778–077Bh to the Extended I/O Bus). 0=Disable (confine to PCI). |
| 4 | **LPT1 Enable.** 1=Enable (forward PCI I/O accesses to 03BC–03BFh and 07BC–07BFh to the Extended I/O Bus). 0=Disable (confine to PCI). |
| 3 | **COM4 Enable.** 1=Enable (forward PCI I/O accesses to 02E8h–02EFh to the Extended I/O Bus). 0=Disable (confine to PCI). |
| 2 | **COM3 Enable.** 1=Enable (forward PCI I/O accesses to 03E8h–03EFh to the Extended I/O Bus). 0=Disable (confine to PCI). |
| 1 | **COM2 Enable.** 1=Enable (forward PCI I/O accesses to 02F8h–02FFh to the Extended I/O Bus). 0=Disable (confine to PCI). |
| 0 | **COM1 Enable.** 1=Enable (forward PCI I/O accesses to 03F8h–03FFh to the Extended I/O Bus). 0=Disable (confine to PCI). |

### 3.2.9. ECRT— EXTENDED I/O CONTROLLER RECOVERY TIMER REGISTER

Address Offset: 4Ch
Default Value: 48h
Attribute: Read/Write

The I/O recovery mechanism in the MPIIX is used to add additional recovery delay between PCI Master originated 8-bit cycles to the Extended I/O Bus. The MPIIX automatically forces a minimum delay of 3.5 SYSCLKs between back-to-back 8-bit I/O cycles to the Extended I/O bus. This delay is measured from the rising edge of the I/O command (IOR# or IOW#) to the falling edge of the next I/O command . If a delay of greater than 3.5 SYSCLKs is required, the Extended I/O Recovery Time Register can be programmed to increase the delay in increments of SYSCLKs. No additional delay is inserted for back-to-back I/O "sub cycles" generated as a result of byte assembly or disassembly. This register defaults to 8-bit recovery enabled with one SYSCLK clock added to the standard I/O recovery.

| Bit | Description |
|---|---|
| 7 | **Reserved.** |
| 6 | **8-Bit I/O Recovery Enable.** 1=Enable the recovery time programmed in bits [5:3]. 0=Disable recovery times in bits [5:3] and the recovery timing of 3.5 SYSCLKs is inserted. |
| 5:3 | **8-Bit I/O Recovery times.** When bit 6=1, this 3-bit field defines the recovery time for 8-bit I/O.<br>**Bit [5:3] SYSCLK** — **Bit [5:3] SYSCLK**<br>001 1 (default) — 101 5<br>010 2 — 110 6<br>011 3 — 111 7<br>100 4 — 000 8 |
| 2 | **16-Bit I/O Recovery Enable.** Reserved. Read as 0. |
| 1:0 | **16-Bit I/O Recovery Times.** Reserved. Read as 0. |

### 3.2.10. BIOSE — BIOS ENABLE REGISTER

Address Offset: 4Eh
Default Value: 08h
Attribute: Read/Write

This register enables/disables BIOS accesses to the different segments. This register also controls the generation of the BIOSCS# signal.

| Bit | Description |
|---|---|
| 7 | **Extended BIOS Enable.** When bit 7=1 (enabled), PCI master accesses to locations FFFC0000h–FFFDFFFFh are forwarded to the Extended I/O Bus and BIOSCS# is generated. When bit 7=0, the MPIIX does not claim the cycle or generate BIOSCS#. |
| 6 | **Lower BIOS Enable 1.** When bit 6=1 (enabled), PCI master accesses to locations 0E4000h–000EFFFFh (and alias at 4G) are forwarded to the Extended I/O Bus and BIOSCS# is generated. When bit 6=0, the MPIIX does not claim the cycle or generate BIOSCS#. |
| 5 | **Lower BIOS Decode Enable 0.** When bit 5=1 (enabled), PCI master accesses to locations 0E0000h–000E3FFFh (and alias at 4G) are forwarded to the Extended I/O Bus. This region is also used for Kanji BIOS. When bit 5=0, the MPIIX does not claim the cycle. |

| Bit | Description |
|---|---|
| 4 | **Lower BIOS CS# Enable 0.** When bit=1 (enabled), PCI master accesses to locations 0E0000h–000E3FFFh (and alias at 4G) generate BIOSCS#. This region is also used for Kanji BIOS. When bit 4=0, the MPIIX does not generate BIOSCS#. |
| 3 | **F-SEGMENT BIOS ENABLE:** 1=Enable (default). 0=Disable. This bit enables the MPIIX to claim cycles to the F-Segment BIOS. When disabled, MPIIX does not claim these cycles. MPIIX should be programmed to NOT claim cycles to the F-Segment BIOS after the DRAM controller is set to claim PCI cycles to the shadowed F-Segment. |
| 2 | **BIOSCS# Write Protect.** When bit 2=1 (enabled), BIOSCS# is asserted for BIOS memory read and write cycles in the decoded BIOS region. When bit 2=0, BIOSCS# is only asserted for BIOS read cycles (MPIIX does not claim the write cycle). |
| 1:0 | **Reserved.** |

### 3.2.11. FDCE—FDC ENABLE REGISTER

Address Offset: 4Fh
Default Value: 21h
Attribute: Read/Write

This register enables/disables accesses to the Floppy Disk on the Extended I/O Bus. This register also enable/disables coprocessor Error Function, IRQ12/Mouse Function, the DOE# Disk Output Enable signal (multiplexed with the SMOUT5 signal), and the RTCALE enable signal (multiplexed with the SMOUT4 signal).

| Bit | Description |
|---|---|
| 7 | **Coprocessor Error function Enable.** 1=Enable. The FERR# input, when asserted, triggers IRQ13 (internal). FERR# is also used to gate the IGNNE# output. |
| 6 | **IRQ12/M Mouse Function Enable.** 1=Mouse function. 0=Standard IRQ12 interrupt function. |
| 5 | **System Management Output 5/Disk Output Enable.** 1=DOE# function on the SMOUT5/DOE# signal. 0=SMOUT5 function on SMOUT5/DOE# (signal reflects the logic level of the SMOUT5 bit in the SMOUTC Register). |
| 4 | **System Management Output 4/RTCALE Enable.** 1=RTCALE function on the SMOUT4/RTCALE signal. 0=SMOUT4 function on SMOUT4/RTCALE (signal reflects the logic level of the SMOUT4 bit in the SMOUTC Register). |
| 3 | **Motherboard DMA 2 Disable:** 1=PAD# function is enabled on the MDAK2#/PAD# pin and the EXTEVNT# function is enabled on the MDRQ2/EXTEVNT# pin. 0 (default)=MDAK2# function is enabled on the MDAK2#/PAD# pin and the MDRQ2 function is enabled on the MDRQ2/EXTEVNT# pin. |
| 2 | **Reserved.** |
| 1 | **Floppy Secondary Address Enable.** When bit 1=1 (Enable), PCI accesses to 0370–0375h and 0377h are forwarded to the Extended I/O Bus. When bit 1=0, MPIIX does not claim these PCI cycles. |
| 0 | **Floppy Primary Address Enable.** When bit 0=1 (Enable), PCI accesses to 03F0–03F5h and 03F7h are forwarded to the Extended I/O Bus. When bit 0=0, MPIIX does not claim these PCI cycles. |

### 3.2.12. PIRQRC [A,B]—PIRQX ROUTE CONTROL REGISTERS

Address Offset : 60h (PIRQRCA) – 61h (PIRQRCB)
Default Value: 80h
Attribute: Read/Write

These registers control the routing of PIRQ[A,B] to the IRQ inputs of the interrupt controller. Each PIRQx# can be independently routed to any one of 11 interrupts. Both PIRQx# lines can be routed to the same IRQx input. Note, that the IRQ selected through bits [3:0] must be set to level sensitive mode in the corresponding ELCR Register. When a PIRQ# line is routed to a given IRQ input to the internal 8259, the corresponding IRQ is masked

| Bit | Description |
|---|---|
| 7 | **Interrupt Routing Enable.** 0=enable. 1=disable. |
| 6:4 | **Reserved.** Read as 0s. |
| 3:0 | **Interrupt Routing:** When bit 7=0, this field selects the routing of the PIRQx to one of the interrupt controller interrupt inputs. |

| Bits [3:0] | IRQ Routing | Bits [3:0] | IRQ Routing |
|---|---|---|---|
| 0 0 0 0 | Reserved | 1 0 0 0 | Reserved |
| 0 0 0 1 | Reserved | 1 0 0 1 | IRQ9 |
| 0 0 1 0 | Reserved | 1 0 1 0 | IRQ10 |
| 0 0 1 1 | IRQ3 | 1 0 1 1 | IRQ11 |
| 0 1 0 0 | IRQ4 | 1 1 0 0 | IRQ12 |
| 0 1 0 1 | IRQ5 | 1 1 0 1 | Reserved |
| 0 1 1 0 | IRQ6 | 1 1 1 0 | IRQ14 |
| 0 1 1 1 | IRQ7 | 1 1 1 1 | IRQ15 |

### 3.2.13. MSTAT—MISCELLANEOUS STATUS REGISTER

Address Offset: 6Bh–6Ah
Default Value: xxxx xxxx xxxx x00S (S=Strapping option)
Attribute: Read Only

This register reports the hardware strapping options selected for the Extended I/O Bus clock divisor.

| Bit | Description |
|---|---|
| 15:3 | **Reserved**. Software should not rely on any particular value in this field. |
| 2:1 | **Reserved.** Read as 0. |
| 0 | **ISA Clock Divisor Status:** This bit reports the strapping option on the SYSCLK signal. 1=clock divisor of 3 (PCICLK=25 MHz). 0 (default)=Clock divisor of 4 (PCICLK=33 MHz). Note that, for PCICLK=30 MHz, a clock divisor of 4 must be selected and produces a SYSCLK of 7.5 MHz. |

### 3.2.14. IDETIM—IDE TIMING REGISTER

Address Offset: 6D–6Ch: Primary/Secondary Channel
Default Value: 0000h
Attribute: Read/Write

This register controls the MPIIX's IDE interface and selects the timing characteristics of the PCI Local Bus IDE cycle.

| Bit | Description |
|---|---|
| 15 | **IDE Decode Enable (IDE):** When bit 15=1 (Enable), PCI I/O accesses to the IDE ATA register blocks (command block and control block) are forwarded to the IDE interface. When bit 15=0, MPIIX does not claim the PCI cycle. |
| 14 | **Primary or Secondary IDE Address Decode.** 0=Primary. 1=Secondary. Bit 15 must be 1 to decode IDE cycles on PCI. |
| 13:12 | **IORDY Sample Point (ISP).** This field determines the number of clocks between DIOx# assertion and the first IORDY# sample point.<br>**Bits [13:12] Number of Clocks**<br>00 5<br>01 4<br>10 3<br>11 2 |
| 11:10 | **Reserved.** |
| 9:8 | **Recovery Time (RCT).** This field determines the minimum number of clocks between the last IORDY# sample point and the DIOx# strobe of the next cycle.<br>**Bits [9:8] Number of Clocks**<br>00 4<br>01 3<br>10 2<br>11 1 |
| 7 | **Reserved.** |
| 6 | **Prefetch and Posting Enable (PPE1).** When this bit is set, prefetch and posting to the IDE data port is enabled for drive 1. |
| 5 | **IORDY Sample Point Enable Drive Select 1 (IE1).** When IE1=0, IORDY sampling is disabled for Drive 1. The internal IORDY signal is forced asserted guaranteeing that IORDY is sampled asserted at the first sample point as specified by the ISP field in this register.<br>When IE1=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 1, all accesses to the enabled I/O address range sample IORDY. The IORDY sample point is specified by the ISP field in this register. |

| Bit | Description |
|---|---|
| 4 | **Fast Timing Bank Drive Select 1 (TIME1).** When TIME1=0, accesses to the data port of the enabled I/O address range use the 16-bit compatible timing PCI local bus path.<br><br>When TIME1=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 1, then accesses to the data port of the enabled I/O address range use the fast timing bank PCI local bus IDE path. Accesses to the data port use fast timing only if bit 7 of this register (DTE1) is 0. Accesses to all non-data ports of the enabled I/O address range use the 8-bit compatible timing PCI local bus path. |
| 3 | **Reserved.** |
| 2 | **Prefetch and Posting Enable (PPE0).** 1=Enable prefetch and posting to the IDE data port for drive 0. 0=Disable. |
| 1 | **IORDY Sample Point Enable Drive Select 0 (IE0).** When IE0=0, IORDY sampling is disabled for Drive 0. The internal IORDY signal is forced asserted guaranteeing that IORDY is sampled asserted at the first sample point as specified by the ISP field in this register.<br><br>When IE0=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 0, all accesses to the enabled I/O address range sample IORDY. The IORDY sample point is specified by the ISP field in this register. |
| 0 | **Fast Timing Bank Drive Select 0 (TIME0).** When TIME0=0, accesses to the data port of the enabled I/O address range use the 16-bit compatible timing PCI local bus path.<br><br>When TIME0=1 and the currently selected drive (via a copy of bit 4 of 1x6h) is Drive 0, then accesses to the data port of the enabled I/O address range use the fast timing bank PCI local bus IDE path. Accesses to the data port use fast timing only if bit 3 of this register (DTE0) is 0. Accesses to all non-data ports of the enabled I/O address range use the 8-bit compatible timing PCI local bus path. |

### 3.2.15. MIRQRC—MOTHERBOARD DEVICE IRQ ROUTE CONTROL REGISTER

Address Offset : 70h
Default Value: 80h
Attribute: R/W

This register controls the routing of MIRQ to the IRQ inputs. MIRQ# can be routed to any one of 11 interrupts. When a MIRQ line and a PIRQ# line are steered to the same interrupt, the device connected to the MIRQ line must be set for active high, level interrupts. In this case, the corresponding interrupt pin is masked. Bit 6 of that motherboard device IRA Route Control Register must be programmed to a 0.

| Bit | Description |
|---|---|
| 7 | **Interrupt Routing Enable:** 0=Enable routing, 1=Disable routing. |
| 6 | **MIRQ/IRQx Sharing Enable:** 0=Disable sharing, 1=Enable sharing. When sharing is disabled and bit 7 of this register is 0, the interrupt specified by bits [3:0] is masked. Interrupt sharing should only be enabled when the device connected to the MIRQ line and the device connected to the IRQ line both produce active high, level interrupts. |
| 5:4 | **Reserved.** Read as 0's. |

| Bit | Description |
|---|---|
| 3:0 | **Interrupt Routing:** When bit 7=0, this field selects the routing of the MIRQ to one of the interrupt controller interrupt inputs.<br>**Bits [3:0] — IRQ Routing — Bits [3:0] — IRQ Routing**<br>0000 Reserved 1000 Reserved<br>0001 Reserved 1001 IRQ9<br>0010 Reserved 1010 IRQ10<br>0011 IRQ3 1011 IRQ11<br>0100 IRQ4 1100 IRQ12<br>0101 IRQ5 1101 Reserved<br>0110 IRQ6 1110 IRQ14<br>0111 IRQ7 1111 IRQ15 |

### 3.2.16. MDMARC[2:0]—MOTHERBOARD DEVICE DMA ROUTE CONTROL REGISTERS

Address Offset : 76h (MDMARC0), 77h (MDMARC1), 78h (MDMARC2),
Default Value: 04h
Attribute: R/W

These registers control the routing of the MDRQ[2:0] and MDAK[2:0]# signals to the DREQ and DACK# signals on the 8237 DMA controllers.

When a MDRQ/MDAK# pair is programmed for DMA channel 2, then DREQ2/DACK2# pins are masked. If more than one of the three Motherboard DMAs are used, the Motherboard DMAs should be programmed to different compatible DMA channels. Programming more than one Motherboard DMA to the same compatible DMA channel will result in unpredictable device operation.

| Bit | Description |
|---|---|
| 7 | **Type F and DMA Buffer Enable (FAST):** 1=Enable for this channel. 0=Disable. |
| 6:3 | **Reserved.** Read as 0's. |
| 2:0 | **DMA Channel select (CHNL):** This field selects the DMA channel connected to the MDRQ/MDAK# pair.<br>**Bits[2:0] — DMA channel — Bits[2:0] — DMA channel**<br>0 0 0 0 1 0 0 default (disabled)<br>0 0 1 1 1 0 1 5<br>0 1 0 2 1 1 0 6<br>0 1 1 3 1 1 1 7 |

### 3.2.17. AUDIOE—AUDIO ENABLE REGISTER

Address Offset: 7Eh
Default Value: 00h
Attribute: Read/Write

This register enables/disables the audio I/O channel and, when enabled, selects the I/O address.

| Bit | Description |
|---|---|
| 7 | **Audio Enable.** 1=Enable (forwards PCI I/O accesses to the address range with the base address selected by bits [3:2] of this register (2x0–2xFh) to the Extended I/O Bus. 0=Disable. |
| 6:4 | **Reserved.** |
| 3:2 | **Audio I/O Address.** These bits select the I/O address for the audio device, when enabled via bit 7 of this register .<br>**Bits [3:2] I/O Address**<br>00 0220h<br>01 0230h<br>10 0240h<br>11 0250h |
| 1:0 | **Reserved.** |

### 3.2.18. DMADS—DMA CH[7:5] DATA SIZE REGISTER

Address Offset: 7Fh
Default Value: E0h
Attribute: Read/Write

This register selects between 8-bit and 16-bit DMA device data size for DMA channels [7:5].

| Bit | Description |
|---|---|
| 7 | **Channel 7 16/8-Bit I/O Count By Words (CH7DS).** 1=16-bit, count by word. 0=8-bit, count by byte. |
| 6 | **Channel 6 16/8-Bit I/O Count By Words (CH6DS).** 1=16-bit, count by word. 0=8-bit, count by byte. |
| 5 | **Channel 5 16/8-Bit I/O Count By Words (CH5DS).** 1=16-bit, count by word. 0=8-bit, count by byte. |
| 4:0 | **Reserved.** Read as 0. |

### 3.2.19. PCIDMAE—PCI DMA ENABLE REGISTER

Address Offset: 80h
Default Value: 00h
Attribute: Read/Write

This register selects, on a channel by channel basis, whether the device using the DMA channel is on the Extended I/O Bus or PCI Bus (including proliferation of the PCI bus).

| Bit | Description |
|---|---|
| 7 | **PCI/Extended I/O Bus DMA CH 7 (PCICH7).** 1=PCI Bus. 0=Extended I/O Bus. |
| 6 | **PCI/Extended I/O Bus DMA CH 6 (PCICH6).** 1=PCI Bus. 0=Extended I/O Bus. |
| 5 | **PCI/Extended I/O Bus DMA CH 5 (PCICH5).** 1=PCI Bus. 0=Extended I/O Bus. |
| 4 | **Reserved.** |
| 3 | **PCI/Extended I/O Bus DMA CH 3 (PCICH3).** 1=PCI Bus. 0=Extended I/O Bus. |
| 2 | **PCI/Extended I/O Bus DMA CH 2 (PCICH2).** 1=PCI Bus. 0=Extended I/O Bus. |
| 1 | **PCI/Extended I/O Bus DMA CH 1 (PCICH1).** 1=PCI Bus. 0=Extended I/O Bus. |
| 0 | **PCI/Extended I/O Bus DMA CH 0 (PCICH0).** 1=PCI Bus. 0=Extended I/O Bus. |

### 3.2.20. PCIDMA[A,B]—PCI DMA AND PCI DMA EXPANSION REGISTER

Address Offset: 88h (REQA#/GNTA#), 89h (REQB#/GNTB#)
Default Value: 08h (both)
Attribute: Read/Write

The PCI DMA Expansion request lines (REQ[A,B]#/GNT[A,B]#) provide PCI DMA and PCI DMA expansion support. The default value for the registers selects the request/grant signal to control a PCI DMA expansion device using DMA channel 0.

| Bit | Description |
|---|---|
| 7:4 | **Reserved.** |
| 3 | **Expansion.** This bit provides an ability to control multiple DMA channels through a single request/grant field. When the expansion bit is set to 1, the expansion agent is required to pass the channel number to the arbiter when requesting service using the PCI DMA expansion channel passing protocol, thus making the DMA channel field a “don’t care”. The expansion hardware will then route the agent’s REQ#/GNT# pair to the appropriate internal DMA DREQ/DACK# pair, depending on the value of the channel number passed to it. |

| Bit | Description |
|---|---|
| 2:0 | **DMA Channel.** This field indicates what DMA Channel the signal pair controls. This allows the request/grant pair to be software routable to any DMA channel. Valid values for the DMA Channel field are:<br><br>**Bits[3:1] Channel — Bits[3:1] Channel**<br><br>000 DMA Channel 0 — 100 Reserved<br>001 DMA Channel 1 — 101 DMA Channel 5<br>010 DMA Channel 2 — 110 DMA Channel 6<br>011 DMA Channel 3 — 111 DMA Channel 7<br><br>Note that MPIIX does not support the use of the PC/PCI REQ#/GNT# pair for PCI Masters. Do not program the channel field to the reserved value of 100. |

### 3.2.21. PMAC[1:0]—PROGRAMMABLE MEMORY ADDRESS CONTROL REGISTERS

Address Offset: 8A–8Bh (PMAC0), 8C–8Dh (PMAC1)
Default Value: 0000h
Attribute: Read/Write

This register provides the memory addresses to be used as Burst Event, Clock Throttle Break Event, or Peripheral Access Detect. The memory address is programmable in the range between 16 Kbytes and 1 Gbyte. The memory range is programmable between 16 Kbytes and 4 Mbytes using the Programmable Memory Address Mask (PMAM) registers.

Note: The memory address must be aligned to the size of the memory range programmed through the PMAM registers. Thus, if the range is 16 Kbytes, the memory address range can start on any 16-Kbyte address boundary), If the memory range is 4 Mbytes, the memory address range can start on any 4 Mbytes address boundary.

| Bit | Description |
|---|---|
| 15:0 | **Memory Address Control (MAC).** This field is compared against PCI addresses AD[29:14] during memory cycles. The upper 2 addresses (AD[31:30]) must be zero for the address to be decoded. |

### 3.2.22. PMAM[1:0]—PROGRAMMABLE MEMORY ADDRESS MASK REGISTERS

Address Offset: 8Eh (PMAM0) and 8Fh (PMAM1)
Default Value: 00h
Attribute: Read/Write

This register defines the size of the memory address range which will be decoded by MPIIX and used to signal Burst Event, Clock Throttle Break Event or Peripheral Access Detect. The starting address is determined by PMAC register programming.

| Bit | Description |
|---|---|
| 7:0 | **Memory Address Mask (MAM).** This field provides mask bits that are used to determine if AD[21:14] are part of the decode or ignored. Bits [7:0] correspond to AD[21:14], respectively. If the bit is set to 1, the corresponding address bit is not used during the decode. Split ranges are precluded. |

### 3.2.23. PARE—PROGRAMMABLE ADDRESS RANGE ENABLE REGISTER

Address Offset: 90h
Default Value: 00h
Attribute: Read/Write

This register enables/disables the use of the address range defined in the Programmable Address Control Registers (bits[5:1]) and the PCSC Register (bit 0). When enabled, the MPIIX forwards I/O accesses to the address range specified by the corresponding PACx/PCSC Registers to the Extended I/O Bus. When disabled, MPIIX does not claim these PCI I/O accesses. This register also enables PCS# assertion for accesses to enabled PAC1 and PAC2 ranges.

| Bit | Description |
|---|---|
| 7 | **PCS# Enable for Programmable Address Range 2.** 1=Enable (if bit 2=1). 0=Disable. |
| 6 | **PCS# Enable for Programmable Address Range 1.** 1=Enable (if bit 1=1). 0=Disable. |
| 5 | **Programmable Address Range 5 Enable.** 1=Enable. 0=Disable |
| 4 | **Programmable Address Range 4 Enable.** 1=Enable. 0=Disable |
| 3 | **Programmable Address Range 3 Enable.** 1=Enable. 0=Disable |
| 2 | **Programmable Address Range 2 Enable.** 1=Enable. 0=Disable |
| 1 | **Programmable Address Range 1 Enable.** 1=Enable. 0=Disable |
| 0 | **PCS# Address Range Enable.** 1=Enable. 0=Disable |

### 3.2.24. PCSC—PROGRAMMABLE CHIP SELECT CONTROL REGISTER

Address Offset: 92–93h
Default Value: 0000h
Attribute: Read/Write

This register defines a 16-bit I/O address range to be forwarded to the Extended I/O Bus and, if enabled (via the PARE register), the generation of the PCS# signal. Note that the PAMA Register selects an address range of 1, 2, 4, 8, or 16 bytes (split range is precluded).

| Bit | Description |
|---|---|
| 15:0 | **PCS Address (PCSADDR).** PCI addresses AD[15:0] are compared against bits [15:2]. AD[31:16] must be 0s for the address to be decoded. |

### 3.2.25. PAC[5:1]—PROGRAMMABLE ADDRESS CONTROL REGISTER

Address Offset: 94–95h—PAC1 A0–A1h—PAC4
96–97h—PAC2 A2–A3h—PAC5
98–99h—PAC3
Default Value: 0000h
Attribute: Read/Write

This register provides a 16-bit I/O address range to be forwarded to the Extended I/O Bus, if enabled via the PARE Register. Note that the PAM[A,B,C] Register selects an address range of 1, 2, 4, 8, or 16 bytes (split range is precluded).

| Bit | Description |
|---|---|
| 15:0 | **Programmable Address Control (PADDRC).** PCI addresses AD[15:0] are compared against bits [15:2]. AD[31:16] must be 0s for the address to be decoded. |

### 3.2.26. PAMA—PROGRAMMABLE ADDRESS MASK A REGISTER

Address Offset: 9Ah
Default Value: 00h
Attribute: Read/Write

This register selects an address range of 1, 2, 4, 8, or 16 bytes (split range is precluded) for the Programmable Address Control 1 Register (PAC1) and the PCSC Register. The bits in this register are used to mask address bits AD[3:0], respectively during I/O decode.

| Bit | Description |
|---|---|
| 7:4 | **Programmable Address Control 1 Mask (PAC1MASK).** 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |
| 3:0 | **Programmable Chip Select Mask (PCSMASK).** 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |

### 3.2.27. PAMB—PROGRAMMABLE ADDRESS MASK B REGISTER

Address Offset: 9Bh
Default Value: 00h
Attribute: Read/Write

This register selects an address range of 1, 2, 4, 8, or 16 bytes (split range is precluded) for the Programmable Address Control Registers (PAC[3,2]).

| Bit | Description |
|---|---|
| 7:4 | **Programmable Address Control 3 Mask (PAC3MASK).** 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |
| 3:0 | **Programmable Address Control 2 Mask (PAC2MASK).** 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |

### 3.2.28. IOCA—I/O CONFIGURATION ADDRESS REGISTER

Address Offset: 9C–9Dh
Default Value: 00h
Attribute: Read/Write

This register provides an I/O address range to be forwarded to the Extended I/O Bus for accesses to the configuration space of an integrated I/O device. PCI address bits AD[9:1] are compared to bits [9:1] of this register. PCI address bits AD[31:10] must be zero for a decode hit.

| Bit | Description |
|---|---|
| 15:10 | **Reserved.** |
| 9:1 | **I/O Configuration Address (IOCA).** This field defines a 2-byte I/O address space between 0 Kbyte and 1 Kbyte that will be forwarded to the Extended I/O Bus, if enabled via the IOCAE bit. |
| 0 | **I/O Configuration Address Enable (IOCAE).** 1=Enable bits [9:1] of this register. 0=Disable (MPIIX does not claim these PCI cycles). |

### 3.2.29. PAMC—PROGRAMMABLE ADDRESS MASK C REGISTER

Address Offset: A4h
Default Value: 00h
Attribute: Read/Write

This register selects an address range of 1, 2, 4, 8, or 16 bytes (split range is precluded) for the Programmable Address Control Registers (PAC[5,4]).

| Bit | Description |
|---|---|
| 7:4 | **Programmable Address Control 5 Mask (PAC5MASK).** 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |
| 3:0 | **Programmable Address Control 4 Mask (PAC4MASK).** 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |

### 3.2.30. PADE[2:0]—PERIPHERAL ACCESS DETECT ENABLE REGISTERS

Address Offset: A5h—PADE0, A6h—PADE1, A7h—PADE2
Default Value: 00h
Attribute: Read/Write

This register enables the addresses used to assert the PAD# signal. Setting the bits to 1 enables the corresponding memory or IO address to be part of the peripheral activity detection range. If a PCI address is detected in the enabled peripheral activity range, the MPIIX asserts the Peripheral Access Detect (PAD#) signal. Setting the bit to 0 disables this function. Tables 16 and 17 (Section 4.8) provide the address range for where the associated function is determined.

| Bits | PADE2 | PADE1 | PADE0 |
|---|---|---|---|
| 7 | Audio-E. | COM4. | PMAC1. |
| 6 | Audio-D. | COM3. | PMAC0. |
| 5 | Audio-C. | COM2. | PAC5. |
| 4 | Audio-B. | COM1. | PAC4. |
| 3 | Audio-A. | FDC, Secondary Drive. | PAC3. |
| 2 | Parallel Port 3. | FDC, Primary Drive. | PAC2. |
| 1 | Parallel Port 2. | IDE, Secondary Drive. | PAC1. |
| 0 | Parallel Port 1. | IDE, Primary Drive. | PCSC. |

### 3.2.31. LTADEV3—Local Trap Address for Device 3 Register

Address Offset: A8h–A9h
Default Value: 00h
Attribute: Read/Write

This register contains a 16-bit trap I/O address for device 3. The address range for this trap address is selected via the LTMDEV3 Register.

| Bit | Description |
|---|---|
| 15:0 | **LTRP_ADDR_DEV3.** Bits [15:0] correspond to PCI I/O address bits AD[15:0]. Note that AD[31:16] must all be 0s for a match. |

### 3.2.32. LTMDEV3—Local Trap Mask for Device 3 Register

Address Offset: AAh
Default Value: 00h
Attribute: Read/Write

This register selects the COM port access that will be trapped. The register also selects a trap address range of 1, 2, 4, or 8 bytes for Device 3 (split range is precluded).

| Bit | Description |
|---|---|
| 7:4 | **LTRP_COM_SEL.** These bits select the COM port accesses that will be trapped. When a bit is written to a 1, an access to the corresponding Local Trap Address will cause a synchronous SMI#.<br><br>**Bits COM Port Address**<br><br>7 COM4 02E8h–02EFh<br>6 COM3 03E8h–03EFh<br>5 COM2 02F8h–02FFh<br>4 COM1 03F8h–03FFh |
| 3:0 | **LTRP_MASK_DEV3.** This field selects the I/O address trap range for Device 3. 1=corresponding address bit is not used in the address decode. 0=Corresponding address bit is used in the address decode. For example, mask field=0011 selects a 4-byte range. |

### 3.2.33. LTSMIE—Local Trap SMI Enable Register

Address Offset: ABh
Default Value: 00h
Attribute: Read/Write

This register enables the local address trap to cause a synchronous SMI for accesses to the corresponding enabled trap address range. The address range for each bit is defined in Section 4.8.3.2, Access Ranges.

| Bit | Description |
|---|---|
| 7:6 | **Reserved.** |
| 5 | **LTRP_SMI_EN_IDE.** 1=Enable. 0=Disable. |

| Bit | Description |
|---|---|
| 4 | **LTRP_SMI_EN_AUD.** 1=Enable. 0=Disable. |
| 3 | **LTRP_SMI_EN_COM.** 1=Enable. 0=Disable. (The address range is defined by the LTMDEV3 Register.) |
| 2 | **LTRP_SMI_EN_DEV3.** 1=Enable. 0=Disable. (The address range is defined by the LTADEV3 and LTMDEV3 Registers.) |
| 1 | **LTRP_SMI_EN_DEV2.** 1=Enable. 0=Disable. (The address range is defined by the PAC1 and PAMA Registers.) |
| 0 | **LTRP_SMI_EN_DEV1.** 1=Enable. 0=Disable. (The address range is defined by the PCSC and PAMA Registers.) |

#### 3.2.34. LTSMIS—Local Trap SMI Status Register

Address Offset: AEh
Default Value: 00h
Attribute: Read/Write

This register indicates that an access to the corresponding enabled local trap caused an SMI# request. The traps are enabled via the LTSMIE Register. The MPIIX sets the request status bits to a 1. Software clears a bit by writing a 0 to it. If MPIIX is setting the bit to a 1 at the same time that software is setting it to 0, the bit is set to 1. The address range for each bit is defined in Section 4.8.3.2, Access Ranges.

| Bit | Description |
|---|---|
| 7:6 | **Reserved.** |
| 5 | **LTRP_STAT_IDE.** |
| 4 | **LTRP_STAT_AUD.** |
| 3 | **LTRP_STAT_COM.** The address range is defined by the LTMDEV3 Register. |
| 2 | **LTRP_STAT_DEV3.** The address range is defined by LTADEV3 and LTMDEV3 Registers. |
| 1 | **LTRP_STAT_DEV2.** The address range is defined by PAC1 and PAMA Registers. |
| 0 | **LTRP_STAT_DEV1.** The address range is by the PCSC and PAMA Registers. |

#### 3.2.35. LSBSMIE—Local Standby SMI Enable Register

Address Offset: B0h
Default Value: 00h
Attribute: Read/Write

When a bit in this register is set to 1, the corresponding Local Standby timer is reloaded with the initial count and begins to count down. An access to the corresponding local trap address reloads the timer. When the timer expires, an SMI# is generated, if enabled in the GSMIE Register. When a bit is set to 0, the corresponding timer does not count down. The address range for each bit is defined in Section 4.8.3.2, Access Ranges.

| Bit | Description |
|---|---|

| Bit | Description |
|---|---|
| 7:6 | **Reserved.** |
| 5 | **LSTBY_SMI_EN_IDE.** |
| 4 | **LSTBY_SMI_EN_AUD.** |
| 3 | **LSTBY_SMI_EN_COM.** The address range is defined by the LTMDEV3 Register. |
| 2 | **LSTBY_SMI_EN_DEV3.** The address range is defined by the LTADEV3 and LTMDEV3 Registers. |
| 1 | **LSTBY_SMI_EN_DEV2.** The address range is defined by the PAC1 and PAMA Registers. |
| 0 | **LSTBY_SMI_EN_DEV1.** The address range is by PCSC and PAMA Registers. |

### 3.2.36. LSBTRE—Local Standby Timer Reload Enable Register

Address Offset: B1h
Default Value: 00h
Attribute: Read/Write

This register enables/disables local standby timer reloading. When an access is made to one of the enabled address ranges selected by bits [7:2] (i.e., IDE, Audio, COM port, and Device [3:1]), all six local standby timers are reloaded with their initial count value. This register also enables/disables motherboard DMA activity on MDAK[2,1] to reload the Audio Local Standby Timer.

| Bit | Description |
|---|---|
| 7 | **LSTBY_RLD_IDE Enable.** 1=Enable. 0=Disable. |
| 6 | **LSTBY_RLD_AUD Enable.** 1=Enable. 0=Disable. |
| 5 | **LSTBY_RLD_COM Enable.** 1=Enable. 0=Disable. |
| 4 | **LSTBY_RLD_DEV3 Enable.** 1=Enable. 0=Disable. |
| 3 | **LSTBY_RLD_DEV2 Enable.** 1=Enable. 0=Disable. |
| 2 | **LSTBY_RLD_DEV1 Enable.** 1=Enable. 0=Disable. |
| 1 | **AUD MDAK2 Enable.** 1=Enable. 0=Disable. |
| 0 | **AUD MDAK1 Enable.** 1=Enable. 0=Disable. |

### 3.2.37. LSBSMIS—Local Standby SMI Status Register

Address Offset: B2h
Default Value: 00h
Attribute: Read/Write

The bits in this register indicate that the corresponding Local Standby Timer expired and caused an SMI. SMI generation for the timers are globally enabled via the GSMIE Register and individually enabled via the LSBMIE Register. MPIIX sets the request bits to a 1. Software clears a bit by writing a 0 to it. If MPIIX is setting the bit to a 1 at the same time that software is setting it to 0, the bit is set to 1.

| Bit | Description |
|---|---|
| 7:6 | **Reserved.** |
| 5 | **LSTBY_STAT_IDE.** 1=IDE Local Standby Timer generated an SMI#. |
| 4 | **LSTBY_STAT_AUD.** 1=Audio Local Standby Timer generated an SMI#. |
| 3 | **LSTBY_STAT_COM.** 1=COM Port Local Standby Timer generated an SMI#. |
| 2 | **LSTBY_STAT_DEV3.** 1=Device 3 Local Standby Timer generated an SMI#. |
| 1 | **LSTBY_STAT_DEV2.** 1=Device 2 Local Standby Timer generated an SMI#. |
| 0 | **LSTBY_STAT_DEV1.** 1=Device 1 Local Standby Timer generated an SMI#. |

### 3.2.38. LSBTIDE—Local Standby IDE Timer Register

Address Offset: B4h
Default Value: 00h
Attribute: Read/Write

This register provides the idle time interval for generating an SMI#. An eight second internal clock provides an idle timeout range of 8 sec. to 34 minutes. The timer can be frozen via the SYSMGNTC Register. The timer is reloaded with the count value programmed into this register when there is an access to the enabled device address, the individual enable bit is set in the LSBSMIE Register, or access to a device enabled in the LSBTRE register.

| Bit | Description |
|---|---|
| 7:0 | **LSTBY_TMR_IDE.** This field contains an 8-bit value for the IDE Local Standby Timer. 00h is an illegal programmed value. |

### 3.2.39. LSBTAUD—Local Standby Audio Timer Register

Address Offset: B5h
Default Value: 00h
Attribute: Read/Write

This register provides the idle time interval for generating an SMI#. An eight second internal clock provides an idle timeout range of 8 sec. to 34 minutes. The timer is individually enabled via the LSBSMIE Register. The timer can be frozen via the SYSMGNTC Register. The timer is reloaded with the count value programmed into this register when there is an access to the enabled device address, the individual enable bit is set in the LSBSMIE Register, or access to a device enabled in the LSBTRE register.

| Bit | Description |
|---|---|
| 7:0 | **LSTBY_TMR_AUD.** This field contains an 8-bit count value for the AUDIO Local Standby Timer. 00h is an illegal programmed value. |

### 3.2.40. LSBTCOM—Local Standby COM Timer Register

Address Offset: B6h
Default Value: 00h
Attribute: Read/Write

This register provides the idle time interval for generating an SMI#. An eight second internal clock provides an idle timeout range of 8 sec. to 34 minutes. The timer is individually enabled via the LSBSMIE Register. The timer can be frozen via the SYSMGNTC Register. The timer is reloaded with the count value programmed into this register when there is an access to the enabled device address, the individual enable bit is set in the LSBSMIE Register, or access to a device enabled in the LSBTRE register.

| Bit | Description |
|---|---|
| 7:0 | **LSTBY_TMR_COM.** This field contains an 8-bit count value for the COM port Local Standby Timer. 00h is an illegal programmed value. |

### 3.2.41. LSBTDEV1—Local Standby Device 1 Timer Register

Address Offset: B8h
Default Value: 00h
Attribute: Read/Write

This register provides the idle time interval for generating an SMI#. An eight second internal clock provides an idle timeout range of 8 sec. to 34 minutes. The timer is individually enabled via the LSBSMIE Register. The timer can be frozen via the SYSMGNTC Register. The timer is reloaded with the count value programmed into this register when there is an access to the enabled device address, the individual enable bit is set in the LSBSMIE Register, or access to a device enabled in the LSBTRE register.

| Bit | Description |
|---|---|
| 7:0 | **LSTBY_TMR_DEV1.** This field contains an 8-bit count value for the DEVICE 1 (PCS#) Local Standby Timer. (Programmable Chip Select, PCS#). 00h is an illegal programmed value. |

### 3.2.42. LSBTDEV2—Local Standby Device 2 Timer Register

Address Offset: B9h
Default Value: 00h
Attribute: Read/Write

This register provides the idle time interval for generating an SMI#. An eight second internal clock provides an idle timeout range of 8 sec. to 34 minutes. The timer is individually enabled via the LSBSMIE Register. The timer can be frozen via the SYSMGNTC Register. The timer is reloaded with the count value programmed into this register when there is an access to the enabled device address, the individual enable bit is set in the LSBSMIE Register, or access to a device enabled in the LSBTRE register.

| Bit | Description |
|---|---|
| 7:0 | **LSTBY_TMR_DEV2.** This field contains an 8-bit count value for the Device 2 Local Standby Timer. (Programmable Address Range 1). 00h is an illegal programmed value. |

### 3.2.43. LSBTDEV3—Local Standby Device 3 Timer Register

Address Offset: BAh
Default Value: 00h
Attribute: Read/Write

This register provides the idle time interval for generating an SMI#. An eight second internal clock provides an idle timeout range of 8 sec. to 34 minutes. The timer is individually enabled via the LSBSMIE Register. The timer can be frozen via the SYSMGNTC Register. The timer is reloaded with the count value programmed into this register when there is an access to the enabled device address, the individual enable bit is set in the LSBSMIE Register, or access to a device enabled in the LSBTRE register.

| Bit | Description |
|---|---|
| 7:0 | **LSTBY_TMR_DEV3.** This field contains an 8-bit count value for the Device 3 Local Standby Timer. (LTADEV3 Register). 00h is an illegal programmed value. |

### 3.2.44. SESMIT—Software/EXTSMI# SMI Delay Timer Register

Address Offset: BCh
Default Value: 00h
Attribute: Read/Write

This timer is enabled via the GSMIE Register. When enabled, the timer provides a delay between a software generated SMI# (setting the SWEXT_SMI_EN_SW bit in the GSMIE Register) or the generation of an EXTSMI# (if enabled via the SYSSMIE Register), and the generation of the SMI# to the CPU. A 1 msec internal clock provides a time delay range of 1 msec to 255 msec. The timer is reloaded when an enabled system event occurs (see SYSEVNTE[2:0] Registers). When this timer generates an SMI, the global status bit for this timer (GSMIS Register) and the individual status bit(s) for the source that caused the SMI are set.

| Bit | Description |
|---|---|
| 7:0 | **SWEXT_SMI_DLY_TMR.** This field contains an 8-bit count value for the Software SMI and EXTSMI# SMI Delay Timer. 00h is an illegal programming count. |

### 3.2.45. SUSSMIT—Suspend SMI Delay Timer Register

Address Offset: BDh
Default Value: 00h
Attribute: Read/Write

This timer generates a delay between the hardware generation of a suspend or resume (asserting the SRBTN# signal) or a battery low indication (asserting the BATLOW# signal) and the corresponding SMI generation. A 128 msec internal clock provides a time delay range of 128 msec to 32 sec. For the SRBTN# or BATLOW# signals to activate this delay timer, there individual enable bits must be set to 1 in the MISCSMIE Register. Note that the generation of BATLOW# can bypass this delay timer and immediately generate an SMI by programming this feature in the SUSRSMC1 Register.

| Bit | Description |
|---|---|
| 7:0 | **SUSP_SMI_DLY_TMR.** This field contains an 8-bit count value for the Suspend/Resume Button SRBTN# and Battery Low BATLOW# SMI Delay Timer. 00h is an illegal programming count. |

### 3.2.46. GSBTMR—Global Standby Timer Register

Address Offset: BEh
Default Value: 00h
Attribute: Read/Write

This register provides a global standby timer interval for generating an SMI#. An eight second internal clock provides a standby timeout range of 8 sec to 34 minutes. This timer is enabled via the GSMIE Register. The timer is reloaded when the enable bit is set, the timer expires, any local standby timer is reloaded, or any enabled system event specfied by the SYSEVNTE[2:0] Registers. A status bit in the GSMIS Register indicates that this timer generated the SMI.

| Bit | Description |
|---|---|
| 7:0 | **GSTBY_TMR.** This field contains an 8-bit count value for the Global Standby Timer. 00h is an illegal programmed value. |

### 3.2.47. CLKTHSBYT — Clock Throttle Standby Timer Register

Address Offset: BFh
Default Value: 00h
Attribute: Read/Write

This register provides the count for STPCLK# negation (no CLKTHL Break Event detected). The MPIIX starts to throttle the clock when the timer expires. The timer can be programmed via bit 7 in Clock Control Register (offset D4h) with the granularity of 4 ms or 32 ms. This provides the timer a range of 4 ms to 1 sec, or 32 ms to 8 sec, respectively. The timer is reloaded everytime an enabled CLKTHL Break Event is detected.

| Bit | Description |
|---|---|
| 7:0 | **CLKTHL_STBY_TMR:** This field contains the count value for the STPCLK# deassertion (no CLKTHL Break Event detected). 00h is an illegal programmed count. |

### 3.2.48. SYSMGNTC—System Management Control Register

Address Offset: C0h
Default Value: 04h
Attribute: Read/Write

This register freezes all power management timers, enables/disables all power management functions, and enables/disables the SMI# signal.

| Bit | Description |
|---|---|
| 7:3 | **Reserved.** |
| 2 | **SM_FREEZE.** 1=Freeze all power management timers (timers stop counting but retain the present count). |
| 1 | **SM_EN.** 0=Disable all power management functions. |
| 0 | **SMI_GATE.** 1=Enable SMI. 0=Disable SMI. When enabled, a system management interrupt condition asserts the SMI# signal. When disabled, the SMI# signal is masked and negated. This bit only affects the SMI# signal and does not affect the detection/recording of SMI. Thus, if an SMI is pending when this bit is set to 1, the SMI# signal is asserted. |

### 3.2.49. SYSSMIE—System SMI Enable Register

Address Offset: C1h
Default Value: 00h
Attribute: Read/Write

This register enables the generation of SMI (if enabled via the SYSMGNTC Register) for the associated hardware events (bits [5:0]), and software events (bit 6).

| Bit | Description |
|---|---|
| 7 | **Reserved.** |
| 6 | **APMC_SMI_EN.** 1=Enable (a write to the APMC Register generates an SMI#). 0=Disable |
| 5 | **SWEXT_SMI_EN_EXTSMI.** 1=Enable (The occurrence of an EXTSMI# reloads the Software /EXTSMI# SMI Delay Timer and an SMI is generated after this timer expires.). 0=Disable |
| 4 | **SYS_SMI_EN_IRQ12.** 1=Enable. 0=Disable. |
| 3 | **SYS_SMI_EN_IRQ8.** 1=Enable. 0=Disable. |
| 2 | **SYS_SMI_EN_IRQ4.** 1=Enable. 0=Disable. |
| 1 | **SYS_SMI_EN_IRQ3.** 1=Enable. 0=Disable. |
| 0 | **SYS_SMI_EN_IRQ1.** 1=Enable. 0=Disable. |

### 3.2.50. MISCSMIE—Misc SMI Enable Register

Address Offset: C2h
Default Value: 00h
Attribute: Read/Write

This register enables a PCI interrupt for a write to the APCM Register, and enables the SRBTN# and BATLOW# signals to generate an SMI.

| Bit | Description |
|---|---|
| 7:4 | **Reserved.** |
| 3 | **APM_CALLBACK_EN.** When bit 3=1 (and the APMC SMI is enabled in the SYSSMIE Register), a write to the APMC Register generates a PCI interrupt (PIRQA). PIRQA can be steered to any available interrupt on the interrupt controller. |
| 2 | **SUSP_SMI_EN_SRBTN.** 1=Enable (SRBTN# assertion causes an SMI, if enabled in the GSMIE Register). 0=Disable. |
| 1 | **SUSP_SMI_EN_BATLOW.** 1=Enable (BATLOW# assertion causes an SMI, if enabled in the GSMIE Register). 0=Disable. |
| 0 | **Reserved.** |

### 3.2.51. GSMIE—Global SMI Enable Register

Address Offset: C3h
Default Value: 00h
Attribute: Read/Write

This register provides a master SMI enable for the system events, software SMI# and EXTSMI#, local traps, local standby timers, the global standby timer, and SRBTN# and BATLOW# suspend signals. The register also enables the Software/EXTSMI SMI Delay Timer.

| Bit | Description |
|---|---|
| 7 | **SYS_SMI_EN.** 1=Enable (master enable for system events in the SYSEVNT[2:0] Registers). 0=Disable these system events from generating an SMI. |
| 6 | **SWEXT_SMI_EN.** 1=Enable software SMI# generated by bit 0 of this register. This bit (bit 6) also enables the EXTSMI# signal to cause an SMI, if enabled in the SYSSMIE Register. 0=Disable these SMIs from generating an SMI. |
| 5 | **Reserved.** |
| 4 | **LTRP_SMI_EN.** 1=Enable (master enable for the local traps). 0=Disable (local traps will not cause an SMI). |
| 3 | **LSTBY_SMI_EN.** 1=Enable (master enable for the standby timers). 0=Disable (standby timers will not cause an SMI). |
| 2 | **GSTBY_SMI_EN.** 1=Enable (global standby timer is loaded with intial value, begins counting, and generates an SMI when the counter expires). 0=Disable global standby timer. |
| 1 | **SUSP_SMI_EN.** 1=Enable (SRBTN# and BATLOW# generate an SMI, if individual enable is set in the MISCSMIE Register). 0=Disable SRBTN# ane BATLOW# from generating an SMI. |

| Bit | Description |
|---|---|
| 0 | **SWEXT_SMI_EN_SW.** This bit permits software to generate an SMI. 1=Enable (Software/EXTSMI SMI Delay Timer is reloaded, starts counting, and generates an SMI when the timer expires). |

#### 3.2.52. SYSSMIS—System SMI STATUS Register

Address Offset: C6h
Default Value: 00h
Attribute: Read/Write

This register indicates whether IRQ[12,8,4,3,1], EXTSMI#, or the software SMI (via bit 0 of the GSMIE Register) generated the SMI. The MPIIX sets these bits to a 1 and software sets these bits to 0 by writing a 0 to the individual bit(s). If the MPIIX is setting a bit to 1 at the same time that software is setting it to a 0, the bit is set to 1.

| Bit | Description |
|---|---|
| 7 | **Reserved.** |
| 6 | **SWEXT_STAT_SW.** 1=Software SMI caused an SMI# (setting bit 0 in the GSMIE Register). |
| 5 | **SWEXT_STAT_EXTSMI.** 1=EXTSMI# signal caused an SMI#. |
| 4 | **SYS_STAT_IRQ12.** 1=IRQ12 caused an SMI#. |
| 3 | **SYS_STAT_IRQ8.** 1=IRQ8# caused an SMI#. |
| 2 | **SYS_STAT_IRQ4.** 1=IRQ4 caused an SMI#. |
| 1 | **SYS_STAT_IRQ3.** 1=IRQ3 caused an SMI#. |
| 0 | **SYS_STAT_IRQ1.** 1=IRQ1 caused an SMI#. |

#### 3.2.53. MISCSMIS—Miscellaneous SMI STATUS Register

Address Offset: C7h
Default Value: 00h
Attribute: Read/Write

This register indicates whether SRBTN# and BATLOW caused an SMI. The register also permits power management software to provide status on whether the system is in global standby. Note that the MPIIX sets bits [2,1] to a 1 and software sets these bits to 0 by writing a 0 to the individual bit(s). If the MPIIX is setting bits [2,1] to 1 at the same time that software is setting the bit to a 0, the bit is set to 1.

| Bit | Description |
|---|---|
| 7:4 | **Reserved.** |
| 3 | **SYSTEM_IN_GSTBY.** This bit is set and reset by software to indicate whether the system is in Global Standby. |
| 2 | **SUSP_STAT_SRBTN.** 1=SRBTN# signal caused an SMI#. |
| 1 | **SUSP_STAT_BATLOW.** 1=BATLOW# signal caused an SMI#. |

| Bit | Description |
|---|---|
| 0 | **Reserved.** |

### 3.2.54. GSMIS — Global SMI STATUS Register

Address Offset: C8h
Default Value: 00h
Attribute: Read/Write

This register indicates whether a system event, EXTSMI#, or the software SMI (via bit 0 of the GSMIE Register) generated the SMI. The register also indicates whether a write to the APMC Register, one of the local traps, one of the local standby timers, the global standby timer, or one of the suspend hardware events caused an SMI. The MPIIX sets these bits to a 1 and software sets these bits to 0 by writing a 0 to the individual bit(s). If the MPIIX is setting a bit to 1 at the same time that software is setting it to a 0, the bit is set to 1.

| Bit | Description |
|---|---|
| 7 | **SYS_STAT.** 1=One of the system SMI events in the SYSEVNT[2:0] Registers caused the SMI#. |
| 6 | **SWEXT_STAT.** 1=Software SMI (programming bit 0 of the GSMIE Register) or EXTSMI# caused the SMI. |
| 5 | **APM_STAT.** 1=Write to APMC Register caused SMI. |
| 4 | **LTRP_STAT.** 1=Access to one of the Local Traps caused an SMI. |
| 3 | **LSTBY_STAT.** 1=One of the Local Standby Timers expired and caused an SMI. |
| 2 | **GSTBY_STAT.** 1=Global Standby Timer expired and caused an SMI. |
| 1 | **SUSP_STAT.** 1=SRBTN# or BATLOW# caused an SMI. |
| 0 | **Reserved.** |

### 3.2.55. SUSRSMC1—Suspend/Resume Control 1 Register

Address Offset: CCh
Default Value: 70h
Attribute: Read/Write

The software programmable bits in this register control various suspend/resume functions. This register also enables the BATLOW# signal to bypass the Suspend SMI Delay Timer and immediately generate an SMI.

| Bit | Description |
|---|---|
| 7 | **BATLOW_BYPASS_EN.** When bit 7=1 (and bit 1=1 in the MISCSMIE Register), the BATLOW# input bypasses the Suspend SMI Delay Timer and cause an SMI# directly. When this bit is set to 0, the BATLOW# SMI waits for the timer to expire. |
| 6 | **RSM_MSK_IRQ8.** 1=IRQ8# will not cause a resume. |
| 5 | **RSM_MSK_COMRI.** 1=COMRI# will not cause a resume. |
| 4 | **RSM_MSK_BATLOW.** 1=BATLOW# will not PREVENT a resume. |

| Bit | Description |
|---|---|
| 3 | **SUS_REF.** This bit is set by the power management software at the end of the suspend routine. The SUS_STAT bit is set automatically by MPIIX when the SUS_REF bit is set by software. This bit is shadowed in the MTSC to initiate the suspend refresh. |
| 2 | **SUS_STAT.** This bit can be set by power management software at the end of the suspend routine. In addition, this bit is set to 1 by the MPIIX when the SUS_REF bit is set to 1. This bit is set to 0 by the power managment resume routine. |
| 1:0 | **SUS_MODE.** This field sets the suspend mode.<br>**Bits[1:0] Function**<br>00 Suspend is Disabled<br>01 Reserved (illegal)<br>10 Suspend-to-DRAM<br>11 Suspend-to-Disk |

### 3.2.56. SUSRSMC2—Suspend/Resume Control 2 Register

Address Offset: CDh
Default Value: 00h
Attribute: Read/Write

This register prevents EXTSMI# from causing a resume event.

| Bit | Description |
|---|---|
| 7:1 | **Reserved.** |
| 0 | **RSM_MSK_EXTSMI.** 1=EXTSMI# will not cause a resume event. |

### 3.2.57. SMOUTC—SMOUT Control Register

Address Offset: CEh
Default Value: 3Fh
Attribute: Read/Write

This register controls the SMOUT[5:0] signals.

| Bit | Description |
|---|---|
| 7:6 | **Reserved.** |
| 5:0 | **SMOUT[5:0].** Writing to any of these bits causes that logical level to be driven on the corresponding SMOUTx signal. When the dual function SMOUT5/DOE# signal is configured for Disk Output Enable (DOE#), writing to the SMOUT5 bit has no effect. When the dual function SMOUT4/RTCALE signal is configured for RTCALE, writing to the SMOUT4 bit has no effect. |

### 3.2.58. SYSEVNTE0—System EVENT Enable 0 Register

Address Offset: D0h
Default Value: 00h
Attribute: Read/Write

This register enables hardware events as system events for power management control.

| Bit | Description |
|---|---|
| 7 | **SYS_EVNT_EN_IRQ_7.** 1=Enable IRQ7 as a System Event. |
| 6 | **SYS_EVNT_EN_IRQ_6.** 1=Enable IRQ6 as a System Event. |
| 5 | **SYS_EVNT_EN_IRQ_5.** 1=Enable IRQ5 as a System Event. |
| 4 | **SYS_EVNT_EN_IRQ_4.** 1=Enable IRQ4 as a System Event. |
| 3 | **SYS_EVNT_EN_IRQ_3.** 1=Enable IRQ3 as a System Event. |
| 2 | **Reserved.** |
| 1 | **SYS_EVNT_EN_IRQ_1.** 1=Enable IRQ1 as a System Event. |
| 0 | **SYS_EVNT_EN_IRQ_0.** 1=Enable IRQ0 as a System Event. |

### 3.2.59. SYSEVNTE1—System EVENT Enable 1 Register

Address Offset: D1h
Default Value: 00h
Attribute: Read/Write

This register enables hardware events as system events for power management control.

| Bit | Description |
|---|---|
| 7 | **SYS_EVNT_EN_IRQ_15.** 1=Enable IRQ15 as a System Event. |
| 6 | **SYS_EVNT_EN_IRQ_14.** 1=Enable IRQ14 as a System Event. |
| 5 | **Reserved.** |
| 4 | **SYS_EVNT_EN_IRQ_12.** 1=Enable IRQ12 as a System Event. |
| 3 | **SYS_EVNT_EN_IRQ_11.** 1=Enable IRQ11 as a System Event. |
| 2 | **SYS_EVNT_EN_IRQ_10.** 1=Enable IRQ10 as a System Event. |
| 1 | **SYS_EVNT_EN_IRQ_9.** 1=Enable IRQ9 as a System Event. |
| 0 | **SYS_EVNT_EN_IRQ_8.** 1=Enable IRQ8# as a System Event. |

### 3.2.60. SYSEVNTE2—System EVENT Enable 2 Register

Address Offset: D2h
Default Value: 00h
Attribute: Read/Write

This register enables hardware events as system events for power management control. This register also contains a master enable for all system events.

| Bit | Description |
|---|---|
| 7 | **SYS_EVNT_EN_HWSUS.** 1=Enable BATLOW# and SRBTN# as a System Event. The corresponding BATLOW# and SRBTN# enable bits must be set to 1 in the MISCSMIE Register for these signals to be recognized as system events. |
| 6 | **SYS_EVNT_EN_EXTSMI.** 1=Enable EXTSMI# as a System Event. The EXTSMI# bit must be set to 1 in the SYSSMIE Register for this signal to be recognized as a system event. |
| 5 | **SYS_EVNT_EN_SMI.** 1=Enable SMI# as a System Event. |
| 4 | **SYS_EVNT_EN_NMI.** 1=Enables NMI as a System Event. |
| 3 | **SYS_EVNT_EN_INTR.** 1=Enable INTR as a System Event. |
| 2 | **Reserved.** |
| 1 | **SYS_EVNT_EN_COMRI.** 1=Enable COMRI# as a System Event. |
| 0 | **SYS_EVNT_EN.** 1=Enable System Events (each System Event is individually enabled via SYSEVNT[2:0]). 0=Disable all System Events. An enabled system event causes the Global Standby Timer and the SMI Delay timers to be reloaded. |

### 3.2.61. BSTCLKT — Burst Count Timer Register

Address Offset: D3h
Default Value: 00h
Attribute: Read/Write

This register provides the 8-bit initial count for the Burst Count Timer that controls the STPCLK# negation period after a Burst Clock Event in ACT mode. The timer runs with the granularity of 32 µs giving the timer the range of 32 µs to 8 ms. The timer is reloaded from this register every time an enabled Burst Clock Event is detected. STPCLK# is asserted when the timer expires.

| Bit | Description |
|---|---|
| 7:0 | **STPCLK_LO_TMR.** This field contains the Burst Count Timer count value. |

### 3.2.62. CLKC—Clock Control Register

Address Offset: D4h
Default Value: 00h
Attribute: Read/Write

This register enables the PCICLK to be stopped and enables clock throttling. The register also permits software to control STPCLK#.

| Bit | Description |
|---|---|
| 7 | **Clock Throttle Standby Timer Frequency (CLKTHSBYT) Timer:** This bit selescts the resolution (granularity) of the 8-bit CLKTHSBYT. 0 = 4 ms, 1 = 32 ms. |
| 6:5 | **Reserved.** |
| 4 | **Auto Clock Throttle** (**ACT_MODE_EN):** 1=Enable. 0=Disable. |
| 3:2 | **STPCLK_MODE.** When either bit is set to 1, a read from the APMC Register causes STPCLK# to be asserted. When bits [3:2]=00, reads from the APMC Register have no effect on the STPCLK# function.<br>**Bits[3:2] Function**<br>00 Disable STPCLK# function<br>01 Enable Stop Grant Mode<br>10 Enable Stop Clock Mode<br>11 Reserved |
| 1 | **CLK_THROTTLE_EN.** 1=Enable clock throttling. 0=Disable clock throttling. |
| 0 | **PCI_CLK_CTRL_EN.** 1=Enable (PCI clock can be stopped). 0=Disable. |

### 3.2.63. STPCLKLT—STPCLK# Low Timer Register

Address Offset: D6h
Default Value: 00h
Attribute: Read/Write

The value in this register defines the duration of the STPCLK# asserted period when bit 1 in the CLKC Register is set to 1. The value in this register is loaded into the STPCLK# Timer when STPCLK# is asserted. The STPCLK# timer counts using a 32-us clock with a range of 32 µs to 8 ms.

| Bit | Description |
|---|---|
| 7:0 | **STPCLK_LO_TMR.** Bits [7:0] define the duration of the STPCLK# asserted period during clock throttling. 00h is an illegal programmed count. |

### 3.2.64. STPCLKHT—STPCLK# High Timer Count

Address Offset: D7h
Default Value: 00h
Attribute: Read/Write

The value in this register defines the duration of the STPCLK# negated period when bit 1 in the CLKC Register is set to 1. The value in this register is loaded into the STPCLK# Timer when STPCLK# is negated. The STPCLK# timer counts using a 32-us clock with a range of 32 µs to 8 ms.

| Bit | Description |
|---|---|
| 7:0 | **STPCLK_HI_TMR.** Bits [7:0] define the duration of the STPCLK# negated period during clock throttling. 00h is an illegal programmed count. |

### 3.2.65. STPBRKE0—Stop Break Event Enable 0 Register

Address Offset: D8h
Default Value: 00h
Attribute: Read/Write

This register enables/disables hardware events as break events to restore system clocks. When a break event is enabled, the corresponding hardware event activity restores the CPU clock by negating STPCLK# and reloading the STPCLKHT Register with its intial count.

| Bit | Description |
|---|---|
| 7 | **STPBRK_EN_IRQ7.** 1=Enable IRQ7 as a break event. |
| 6 | **STPBRK_EN_IRQ6.** 1=Enable IRQ6 as a break event. |
| 5 | **STPBRK_EN_IRQ5.** 1=Enable IRQ5 as a break event. |
| 4 | **STPBRK_EN_IRQ4.** 1=Enable IRQ4 as a break event. |
| 3 | **STPBRK_EN_IRQ3.** 1=Enable IRQ3 as a break event. |
| 2 | **Reserved.** |
| 1 | **STPBRK_EN_IRQ1.** 1=Enable IRQ1 as a break event. |
| 0 | **STPBRK_EN_IRQ0.** 1=Enable IRQ0 as a break event. |

### 3.2.66. STPBRKE1—Stop Break Event Enable 1 Register

Address Offset: D9h
Default Value: 00h
Attribute: Read/Write

This register enables/disables hardware events as break events to restore system clocks. When a break event is enabled, the corresponding hardware event activity restores the CPU clock by negating STPCLK# and reloading the STPCLKHT Register with its intial count.

| Bit | Description |
|---|---|
| 7 | **STPBRK_EN_IRQ15.** 1=Enable IRQ15 as a break event. |
| 6 | **STPBRK_EN_IRQ14.** 1=Enable IRQ14 as a break event. |
| 5 | **Reserved.** |
| 4 | **STPBRK_EN_IRQ12.** 1=Enable IRQ12 as a break event. |
| 3 | **STPBRK_EN_IRQ11.** 1=Enable IRQ11 as a break event. |
| 2 | **STPBRK_EN_IRQ10.** 1=Enable IRQ10 as a break event. |
| 1 | **STPBRK_EN_IRQ9.** 1=Enable IRQ9 as a break event. |
| 0 | **STPBRK_EN_IRQ8.** 1=Enable IRQ8# as a break event. |

### 3.2.67. STPBRKE2—Stop Break Event Enable 2 Register

Address Offset: DAh
Default Value: 00h
Attribute: Read/Write

This register enables/disables hardware events as break events to restore system clocks. When a break event is enabled, the corresponding hardware event activity restores the CPU clock by negating STPCLK# and reloading the STPCLKHT Register with its intial count. This register also disables all break events.

| Bit | Description |
|---|---|
| 7 | **STPBRK_EN_HWSUS.** 1=Enable SRBTN# and BATLOW# as break events. For these signals to be recognized as break events, the corresponding SMIs musts be enabled in the MISCSMIE Register. |
| 6 | **STPBRK_EN_EXTSMI.** 1=Enable EXTSMI# as a break event. For this signal to be recognized as break event, SMIs must be enable for EXTSMI# in the SYSSMIE Register. |
| 5 | **STPBRK_EN_SMI.** 1=Enable SMI# as a break event. |
| 4 | **STPBRK_EN_NMI.** 1=Enable NMI as a break event. |
| 3 | **STPBRK_EN_INTR.** 1=Enable INTR as a break event. |
| 2 | **Reserved.** |
| 1 | **STPBRK_EN_COMRI.** 1=Enable COMRI# as a break event. |
| 0 | **STPBRK_EN.** 0=Disable all break events. 1=Break events are enabled by their respective enables. |

### 3.2.68. SHDW—Shadow Register Access Port

Register Location: E0h
Default Value: undefined
Attribute: Read/Write.

MPIIX includes a set of shadow registers for the standard AT write-only registers. In the transition to Suspend mode, the content of these registers is saved so the system state can be restored, when resumed. The shadowed registers can be read through the PCI configuration register.

When written, the SHDW register initializes a counter that points to a shadow register. When the SHDW Register is read, it returns the data from the shadow register pointed to by the counter. The counter increments the count every time software reads the register. Tables 3.3–3.5 define the MPIIX shadow registers with the register counter.

**Table 3. DMA 1 Registers**

| Register Counter | AT I/O address | Description |
|---|---|---|
| **Master DMA** | | |
| 00 | 00 | Channel 0 Base Address Register (low byte). |
| 01 | 00 | Channel 0 Base Address Register (high byte). |
| 02 | 01 | Channel 0 Base Word Count Register (low byte). |
| 03 | 01 | Channel 0 Base Word Count Register (high byte). |
| 04 | 02 | Channel 1 Base Address Register (low byte). |
| 05 | 02 | Channel 1 Base Address Register (high byte). |
| 06 | 03 | Channel 1 Base Word Count Register (low byte). |
| 07 | 03 | Channel 1 Base Word Count Register (high byte). |
| 08 | 04 | Channel 2 Base Address Register (low byte). |
| 09 | 04 | Channel 2 Base Address Register (high byte). |
| 0A | 05 | Channel 2 Base Word Count Register (low byte). |
| 0B | 05 | Channel 2 Base Word Count Register (high byte). |
| 0C | 06 | Channel 3 Base Address Register (low byte). |
| 0D | 06 | Channel 3 Base Address Register (high byte). |
| 0E | 07 | Channel 3 Base Word Count Register (low byte). |
| 0F | 07 | Channel 3 Base Word Count Register (high byte). |
| 10 | 08 | DMA1 Command Register. |
| 11 | 0B | Channel 0 Mode Register. |
| 12 | 0B | Channel 1 Mode Register. |
| 13 | 0B | Channel 2 Mode Register. |
| 14 | 0B | Channel 3 Mode Register. |
| 15 | 0F | DMA1 Mask Register. |

**Table 4. DMA 2 Registers**

| Register Counter | AT I/O address | Description |
|---|---|---|
| | | **Slave DMA** |
| 16 | C4 | Channel 5 Base Address Register (low byte). |
| 17 | C4 | Channel 5 Base Address Register (high byte). |
| 18 | C6 | Channel 5 Base Word Count Register (low byte). |
| 19 | C6 | Channel 5 Base Word Count Register (high byte). |
| 1A | C8 | Channel 6 Base Address Register (low byte) |
| 1B | C8 | Channel 6 Base Address Register (high byte). |
| 1C | CA | Channel 6 Base Word Count Register (low byte). |
| 1D | CA | Channel 6 Base Word Count Register (high byte). |
| 1E | CC | Channel 7 Base Address Register (low byte). |
| 1F | CC | Channel 7 Base Address Register (high byte). |
| 20 | CD | Channel 7 Base Word Count Register (low byte). |
| 21 | CD | Channel 7 Base Word Count Register (high byte). |
| 22 | D0 | DMA2 Command Register. |
| 23 | D6 | Channel 5 Mode Register. |
| 24 | D6 | Channel 6 Mode Register. |
| 25 | D6 | Channel 7 Mode Register. |
| 26 | DE | DMA2 Mask Register. |

NOTE: The Base Address Registers, the Base Word Counter Registers, and Mode Register of DMA channel 4 are not shadowed. However, the Mask bit of DMA channel 4 is still shadowed.

**Table 5. Programmable Interrupt Controller and Other Registers**

| Register Counter | AT I/O address | Description |
|---|---|---|
| | | **Interrupt Controller** |
| 27 | 20 | PIC1 ICW1. |
| 28 | 21 | PIC1 ICW2. |
| 29 | 21 | PIC1 ICW3. |
| 2A | 21 | PIC1 ICW4 |
| 2B | 20 | PIC1 OCW2. |
| 2C | A0 | PIC2 ICW1. |
| 2D | A1 | PIC2 ICW2. |
| 2E | A1 | PIC2 ICW3. |

| Register Counter | AT I/O address | Description |
|---|---|---|
| 2F | A1 | PIC2 ICW4. |
| 30 | A0 | PIC2 OCW2. |
| | | OTHER. |
| 31 | 70 | NMI mask / RTC address. |
| 32 | 03FAh | COM1 FIFO Enable Register bits 0, 3, 6, 7 (other bits undefined). |
| 33 | 02FAh | COM2 FIFO Enable Register bits 0, 3, 6, 7 (other bits undefined). |
| 34 | 03EAh | COM3 FIFO Enable Register bits 0, 3, 6, 7 (other bits undefined). |
| 35 | 02EAh | COM4 FIFO Enable Register bits 0, 3, 6, 7 (other bits undefined). |
| 36 | 40h | TIMER 0 Count Register (low byte). |
| 37 | 40h | TIMER 0 Count Register (high byte). |
| 38 | 20h | Master PIC OCW3 Register (bits 0,2,5). |
| 39 | A0h | Slave PIC OCW3 Register (bits 0,2,5). |
| | | |
| | | Total 58 Registers. |

### 3.2.69. BSTCLKEE[6:0]—Burst Clock Event Enable Registers

Address Offset: E4h (BSTCLKEE0) to EAh (BSTCLKEE6)
Default Value: 00h
Attribute: Read/Write

These registers enable various activities as Burst Clock Events. Setting a bit to 1 enables the corresponding activity as a Burst Clock Event. When activity is detected, STPCLK# is negated, if necessary, and the Burst Clock Timer is reloaded. The Burst Clock Event Enable bit (bit 0, BSTCLKEE2 register) globally enables these events.

| Bit | BSTCLK ENVT EN_6 | BSTCLK ENVT EN_5 | BSTCLK ENVT EN_4 | BSTCLK ENVT EN_3 | BSTCLK ENVT EN_2 | BSTCLK ENVT EN_1 | BSTCLK ENVT EN_0 |
|---|---|---|---|---|---|---|---|
| 7 | Audio-E | COM4 | PMAC1 | Reserved | Reserved | IRQ15 | IRQ7 |
| 6 | Audio-D | COM3 | PMAC0 | Reserved | EXTSMI# | IRQ14 | IRQ6 |
| 5 | Audio-C | COM2 | PAC5 | Reserved | SMI# | Reserved | IRQ5 |
| 4 | Audio-B | COM1 | PAC4 | Reserved | Reserved | IRQ12 | IRQ4 |
| 3 | Audio-A | FDC-S | PAC3 | Reserved | Reserved | IRQ11 | IRQ3 |
| 2 | Parallel-3 | FDC-P | PAC2 | Reserved | Reserved | IRQ10 | Reserved |
| 1 | Parallel-2 | IDE-S | PAC1 | EXTEVNT# | COMRI# | IRQ9 | IRQ1 |
| 0 | Parallel-1 | IDE-P | PCSC | PHLDA# | Burst Clock | IRQ8# | IRQ0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Event Enable | | |

### 3.2.70. CLKTHLBRKEE[6:0]—Clock Throttle Break Event Enable Registers

Address Offset: ECh (CLKTHLBRKEE0) to F2h (CLKTHLBRKEE6)
Default Value: 00h
Attribute: Read/Write

These registers enable various activities as Clock Throttle Break Events. Setting a bit to 1 enables the corresponding activity as a Clock Throttle Break Event. When activity is detected, STPCLK# is negated, if necessary, and the Clock Throttle Standby Timer reloaded. The CLKTHLBRKE Enable bit (bit 0, CLKTHLBRKEE2 register) globally enables these events.

| Bits | CLKTHL BRKEVNT EN_6 | CLKTHL BRKEVNT EN_5 | CLKTHL BRKEVNT EN_4 | CLKTHL BRKEVNT EN_3 | CLKTHL BRKEVNT EN_2 | CLKTHL BRKEVNT EN_1 | CLKTHL BRKEVNT EN_0 |
|---|---|---|---|---|---|---|---|
| 7 | Audio-E | Serial-4 | PMAC1 | Reserved | BATLOW# /SRBTN# | IRQ15 | IRQ7 |
| 6 | Audio-D | Serial-3 | PMAC0 | Reserved | EXTSMI# | IRQ14 | IRQ6 |
| 5 | Audio-C | Serial-2 | PAC5 | Reserved | SMI# | Reserved | IRQ5 |
| 4 | Audio-B | Serial-1 | PAC4 | Reserved | NMI | IRQ12 | IRQ4 |
| 3 | Audio -A | FDC-S | PAC3 | Reserved | INTR | IRQ11 | IRQ3 |
| 2 | Parallel-3 | FDC-P | PAC2 | Reserved | Reserved | IRQ10 | Reserved |
| 1 | Parallel-2 | IDE-S | PAC1 | EXTEVNT# | COMRI# | IRQ9 | IRQ1 |
| 0 | Parallel-1 | IDE-P | PCSC | PHLDA# | CLKTHL BRKEVNT Enable | IRQ8# | IRQ0 |

## 3.3. ISA Compatible Registers

The ISA Compatible registers contain the DMA, timer/counter, and interrupt registers. This group also contains the NMI, and reset registers.

### 3.3.1. DMA REGISTERS

The MPIIX contains DMA circuitry that incorporates the functionality of two 82C37 DMA controllers (DMA1 and DMA2). The DMA registers control the operation of the DMA controllers and are all accessible via the PCI Bus interface. This section describes the DMA registers. Unless otherwise stated, a PCIRST sets each register to its default value.

#### 3.3.1.1. DCOM—DMA Command Register

Register Location: Channels 0–3—08h
Channels 4–7—0D0h
Default Value: 00h
Attribute: Write Only

This 8-bit register controls the configuration of the DMA. Note that disabling channels 4-7 also disables channels 0-3, since channels 0-3 are cascaded onto channel 4.

| Bit | Description |
|---|---|
| 7 | **DACK# ACTIVE Level (DACK[3:0,7:5]#).** This bit is hardwired to 0. DACK[3:0,7:5]# are always active low. |
| 6 | **DREQ Sense Assert Level (DREQ[3:0,7:5]).** This bit is hardwired to 0. DREQ[3:0,7:5] are always active high. |
| 5 | **Reserved.** Must be 0. |
| 4 | **DMA Group Arbitration Priority.** 1=Rotating priority. 0=Fixed priority. |
| 3 | **Reserved**. Must be 0. |
| 2 | **DMA Channel Group Enable.** 1=Disable. 0=Enable. |
| 1:0 | **Reserved**. Must be 0. |

#### 3.3.1.2. DCM—DMA Channel Mode Register

Register Location: Channels 0–3—0Bh
Channels 4–7—0D6h
Default Value: Bits[7:2]=0, Bits[1:0]=undefined
Attribute: Write Only

Each channel has a DMA Channel Mode Register. The Channel Mode Registers provide control over DMA transfer type, transfer mode, address increment/decrement, and autoinitialization. This register is set to its default state upon PCIRST or Master Clear.

| Bit | Description |
|---|---|
| 7:6 | **DMA Transfer Mode:** Each DMA channel can be programmed in one of four different modes. Note that channels programmed for block or cascade mode or channels that are used for PCI DMA can not be programmed for type F timing mode.<br>**Bits[7:6] Transfer Mode**<br>00 Demand mode<br>01 Single mode<br>10 Block mode<br>11 Cascade mode |
| 5 | **Address Increment/Decrement Select:** 0=Increment; 1=Decrement. |
| 4 | **Autoinitialize Enable:** 1=Enable; 0=Disable. |

| Bit | Description |
|---|---|
| 3:2 | **DMA Transfer Type:** When Bits [7:6]=11, the transfer type bits are irrelevant.<br>**Bits[3:2] Transfer Type**<br>00 Verify transfer<br>01 Write transfer<br>10 Read transfer<br>11 Illegal |
| 1:0 | **DMA Channel Select:** Bits [1:0] select the DMA Channel Mode Register written to by bits [7:2].<br>**Bits[1:0] Channel**<br>00 Channel 0 (4)<br>01 Channel 1 (5)<br>10 Channel 2 (6)<br>11 Channel 3 (7) |

#### 3.3.1.3. DR—DMA Request Register

Register Location: Channels 0–3—09h
Channels 4–7—0D2h
Default Value: Reserved
Attribute: Write Only

Writes to these register address locations are claimed by the MPIIX but have no effect.

| Bit | Description |
|---|---|
| 7:3 | **Reserved**. Must be 0. |
| 2 | **DMA Channel Service Request.** Reserved. |
| 1:0 | **DMA Channel Select.** Reserved. |

#### 3.3.1.4. Mask Register—Write Single Mask Bit

Register Location: Channels 0–3—0Ah
Channels 4–7—0D4h
Default Value: Bits[1:0]=undefined, Bit 2=1, Bits[7:3]=0
Attribute: Write Only

A channel's mask bit is automatically set when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for autoinitialization). Setting the entire register disables all DMA requests until a clear mask register instruction allows them to occur. This instruction format is similar to the format used with the DMA Request Register. Masking DMA channel 4 (DMA controller 2, channel 0) also masks DMA channels [3:0]. The fields in this register are set to 1 following a PCIRST or Master Clear.

| Bit | Description |
|---|---|
| 7:3 | **Reserved**. Must be 0. |
| 2 | **Channel Mask Select.** 1=Disable DREQ for the selected channel. 0=Enable DREQ for the selected channel. |

| Bit | Description |
|---|---|
| 1:0 | **DMA Channel Select.** Bits [1:0] select the DMA Channel Mode Register to program with bit 2.<br>**Bits [1:0] Channel**<br>00 0 (4)<br>01 1 (5)<br>10 2 (6)<br>11 3 (7) |

#### 3.3.1.5. Mask Register—Write All Mask Bits

Register Location: Channels 0–3—0Fh
Channels 4–7—0DEh
Default Value: Bit[3:0]=1, Bit[7:4]=0
Attribute: Read/Write

A channel's mask bit is automatically set to 1 when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for autoinitialization). Setting bits [3:0] to 1 disables all DMA requests until a clear mask register instruction enables the requests. Note that, masking DMA channel 4 (DMA controller 2, channel 0), masks DMA channels [3:0].

| Bit | Description |
|---|---|
| 7:4 | **Reserved**. Must be 0. |
| 3:0 | **Channel Mask Bits.** 1=Disable the corresponding DREQ(s); 0=Enable the corresponding DREQ(s). Bits [3:0] are set to 1 upon PCIRST or Master Clear.<br>**Bit Channel**<br>0 0 (4)<br>1 1 (5)<br>2 2 (6)<br>3 3 (7) |

#### 3.3.1.6. DS—DMA Status Register

Register Location: Channels 0–3—08h
Channels 4–7—0D0h
Default Value: 00h
Attribute: Read Only

Each DMA controller has a read-only DMA Status Register that indicates which channels have reached terminal count and which channels have a pending DMA request.

| Bit | Description |
|---|---|
| 7:4 | **Channel Request Status.** When a valid DMA request is pending for a channel (on its DREQ signal line), the corresponding bit is set to 1. When a DMA request is not pending for a particular channel, the corresponding bit is set to 0. Note that channel 4 is used to cascade the two DMA controllers together and is not used for DMA transfers, so the response for a read of DMA2 status for channel 4 is irrelevant.<br>**Bit** — **Channel**<br>4 — 0<br>5 — 1 (5)<br>6 — 2 (6)<br>7 — 3 (7) |
| 3:0 | **Channel Terminal Count Status.** 1=TC is reached; 0=TC is not reached.<br>**Bit** — **Channel**<br>0 — 0<br>1 — 1 (5)<br>2 — 2 (6)<br>3 — 3 (7) |

#### 3.3.1.7. DMA Base and Current Address Registers (8237 Compatible Segment)

Register Location: DMA Channel 0—000h DMA Channel 4—0C0h
DMA Channel 1—002h DMA Channel 5—0C4h
DMA Channel 2—004h DMA Channel 6—0C8h
DMA Channel 3—006h DMA Channel 7—0CCh
Default Value: Undefined
Attribute: Read/Write

This Register works in conjunction with the Low Page Register. After an autoinitialization, this register retains the original programmed value. Autoinitialize takes place after a TC. The address register is automatically incremented or decremented after each transfer. This register is read/written in successive 8-bit bytes. The programmer must issue the "Clear Byte Pointer Flip-Flop" command to reset the internal byte pointer and correctly align the write prior to programming the Current Address Register. Autoinitialize takes place only after a TC.

| Bit | Description |
|---|---|
| 15:0 | **Base and Current Address [15:0].** These bits represent the 16 least significant address bits used during DMA transfers. Together with the DMA Low Page Register, they form the ISA-compatible 24-bit DMA address. Upon PCIRST or Master Clear, the value of these bits is undefined. |

#### 3.3.1.8. DMA Base and Current Byte/Word Count Registers (Compatible Segment)

| | | |
|---|---|---|
| Register Location: | DMA Channel 0—001h | DMA Channel 4—0C2h |
| | DMA Channel 1—003h | DMA Channel 5—0C6h |
| | DMA Channel 2—005h | DMA Channel 6—0CAh |
| | DMA Channel 3—007h | DMA Channel 7—0CEh |
| | Default Value: Undefined | |
| Attribute: | Read/Write | |

This register determines the number of transfers to be performed. The actual number of transfers is one more than the number programmed in the Current Byte/Word Count Register When the value in the register is decremented from zero to FFFFh, a TC is generated. After an autoinitialization, this register retains the original programmed value. Autoinitialize can only occur when a TC occurs. If it is not autoinitialized, this register has a count of FFFFh after TC.

For transfers to/from an 8-bit I/O, the Byte/Word count indicates the number of bytes to be transferred. This applies to DMA channels 0-3. For transfers to/from a 16-bit I/O, with shifted address, the Byte/Word count indicates the number of 16-bit words to be transferred. This applies to DMA channels 5–7.

| Bit | Description |
|---|---|
| 15:0 | **Base and Current Byte/ Word Count.** This field represents the 16-byte/word count bits used when counting down a DMA transfer. Upon PCIRST or Master Clear, the value of these bits is undefined. |

#### 3.3.1.9. DMA Memory Low Page Registers

| | | |
|---|---|---|
| Register Location: | DMA Channel 0—087h | DMA Channel 5—08Bh |
| | DMA Channel 1—083h | DMA Channel 6—089h |
| | DMA Channel 2—081h | DMA Channel 7—08Ah |
| | DMA Channel 3—082h | |
| | Default Value: Undefined | |
| Access: | Read/Write | |

Each channel has an 8-bit Low Page Register. The DMA memory Low Page Register contains bits 23–16 of the 24-bit address. The register works in conjunction with the DMA controller's Current Address Register to define the complete (24-bit) address for the DMA channel. This 8-bit register is read or written directly. This register is static throughout the DMA transfer. Following an autoinitialization, this register retains the original programmed value. Autoinitialize takes place only after a TC.

| Bit | Description |
|---|---|
| 7:0 | **DMA Low Page [23:16].** These bits represent address bits [23:16] of the 24-bit DMA address. Upon PCIRST or Master Clear, the value of these bits is undefined. |

#### 3.3.1.10. DMA Clear Byte Pointer Register

Register Location: Channels 0–3—00Ch
Channels 4–7—0D8h
Default Value: All bits undefined
Attribute: Write Only

Writing to this register executes the Clear Byte Pointer Command. This command is executed prior to reading/writing a new address or word count to the DMA. The command initializes the byte pointer flip-flop to a known state so that subsequent accesses to register contents address upper and lower bytes in the correct sequence. The Clear Byte Pointer Command (or CPURST or the Master Clear Command) clears the internal latch used to address the upper or lower byte of the 16-bit Address and Word Count Registers.

| Bit | Description |
|---|---|
| 7:0 | **Clear Byte Pointer.** No specific pattern. Command enabled with a write to the I/O port address. |

#### 3.3.1.11. DMC—DMA Master Clear Register

Register Location: Channel 0–3—00Dh
Channel 4–7—0DAh
Default Value: All bits undefined
Attribute: Write Only

This software instruction has the same effect as the hardware Reset. The Command, Status, Request, and Internal First/Last Flip-Flop registers are cleared and the Mask Register is set. The DMA controller enters the idle cycle. There are two independent Master Clear Commands; 0Dh acts on channels 0-3, and 0DAh acts on channels 4-7.

| Bit | Description |
|---|---|
| 7:0 | **Master Clear.** No specific pattern. Command enabled with a write to the I/O port address |

#### 3.3.1.12. DCLM—DMA Clear Mask Register

Register Location: Channel 0–3—00Eh
Channel 4–7—0DCh
Default Value: All bits undefined
Attribute: Write Only

This command clears the mask bits of all four channels.

| Bit | Description |
|---|---|
| 7:0 | **Clear Mask Register.** No specific pattern. Command enabled with a write to the I/O port address. |

### 3.3.2. TIMER/COUNTER REGISTERS

#### 3.3.2.1. TCW—Timer Control Word Register

Register Location: 043h
Default Value: All bits undefined
Attribute: Write Only

The Timer Control Word Register specifies the counter selection, the operating mode, the counter byte programming order and size of the count value, and whether the counter counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count can be written at any time. The new value takes effect according to the programmed mode.

| Bit | Description |
|---|---|
| 7:6 | **Counter Select.**<br>**Bit [7:6] Function**<br>00 Counter 0 select<br>01 Counter 1 select<br>10 Counter 2 select<br>11 Read Back Command |
| 5:4 | **Read/Write Select.**<br>**Bit [5:4] Function**<br>00 Counter Latch Command<br>01 R/W Least Significant Byte (LSB)<br>10 R/W Most Significant Byte (MSB)<br>11 R/W LSB then MSB |
| 3:1 | **Counter Mode Selection.** Bits [3:1] select one of six possible counter modes.<br>**Bit [3:1] Mode Function**<br>000 0 Out signal on end of count (=0)<br>001 1 Hardware retriggerable one-shot<br>X10 2 Rate generator (divide by n counter)<br>X11 3 Square wave output<br>100 4 Software triggered strobe<br>101 5 Hardware triggered strobe |
| 0 | **Binary/BCD Countdown Select.** 0=Binary countdown. The largest possible binary count is $2^{16}$. 1=Binary coded decimal (BCD) count is used. The largest BCD count allowed is $10^4$. |

**Read Back Command**

The Read Back Command is used to determine the count value, programmed mode, and current states of the OUT pin and Null count flag of the selected counter or counters. The Read Back Command is written to the Timer Control Word Register which latches the current states of the above mentioned variables. The value of the counter and its status may then be read by I/O access to the counter address. Note that The Timer Counter Register bit definitions are different during the Read Back Command than for a normal Timer Counter Register write.

| Bit | Description |
|---|---|
| 7:6 | **Read Back Command:** When bits[7:6]=11, the Read Back Command is selected during a write to the Timer Control Word Register. Following the Read Back Command, I/O reads from the selected counter's I/O addresses produce the current latch status, the current latched count, or both if bits 4 and 5 are both 0. |
| 5 | **Latch Count of Selected Counters:** When bit 5=0, the current count value of the selected counters will be latched. When bit 5=1, the count will not be latched. |
| 4 | **Latch Status of Selected Counters:** When bit 4=0, the status of the selected counters will be latched. When bit 4=1, the status will not be latched. The status byte format is described in Section 4.3.3, Interval Timer Status Byte Format Register. |
| 3 | **Counter 2 Select:** When bit 3=1, Counter 2 is selected for the latch command selected with bits 4 and 5. When bit 3=0, status and/or count will not be latched. |
| 2 | **Counter 1 Select:** When bit 2=1, Counter 1 is selected for the latch command selected with bits 4 and 5. When bit 2=0, status and/or count will not be latched. |
| 1 | **Counter 0 Select:** When bit 1=1, Counter 0 is selected for the latch command selected with bits 4 and 5. When bit 1=0, status and/or count will not be latched. |
| 0 | **Reserved**. Must be 0. |

**Counter Latch Command**

The Counter Latch Command latches the current count value at the time the command is received. If a Counter is latched once and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued. If the counter is programmed for two-byte counts, two bytes must be read. The two bytes do not have to be read successively (read, write, or programming operations for other counters may be inserted between the reads). Note that the Timer Counter Register bit definitions are different during the Counter Latch Command than for a normal Timer Counter Register write. Note that, If a counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine that also reads from that same counter. Otherwise, an incorrect count will be read.

| Bit | Description |
|---|---|
| 7:6 | **Counter Selection.** Bits 6 and 7 are used to select the counter for latching.<br>**Bit [7:6] Function**<br>00 latch counter 0 select<br>01 latch counter 1 select<br>10 latch counter 2 select<br>11 Read Back Command select (do not use for counter latch command) |
| 5:4 | **Counter Latch Command.** When bits[5:4]=00, the Counter Latch Command is selected during a write to the Timer Control Word Register. Following the Counter Latch Command, I/O reads from the selected counter's I/O addresses produce the current latched count. |
| 3:0 | **Reserved**. Must be 0. |

#### 3.3.2.2. Interval Timer Status Byte Format Register

Register Location: Counter 0—040h
Counter 1—041h
Counter 2—042h
Default Value: Bits[6:0]=Undefined, Bit 7=0
Attribute: Read Only

Each counter's status byte can be read following an Interval Timer Read Back Command. If latch status is chosen (bit 4=0, Read Back Command) as a read back option for a given counter, the next read from the counter's Counter Access Ports Register returns the status byte.

| Bit | Description |
|---|---|
| 7 | **Counter OUT Pin State:** 1=Pin is 1; 0=Pin is 0. |
| 6 | **Count Register Status:** This bit indicates when the last count written to the Count Register (CR) has been loaded into the counting element (CE). 0=Count has been transferred from CR to CE and is available for reading. 1=Count has not been transferred from CR to CE and is not yet available for reading. |
| 5:4 | **Read/Write Selection Status:** Bits[5:4] reflect the read/write selection made through bits[5:4] of the Control Register.<br>**Bit[5:4] Function**<br>00 Counter Latch Command<br>01 R/W Least Significant Byte (LSB)<br>10 R/W Most Significant Byte (MSB)<br>11 R/W LSB then MSB |
| 3:1 | **Mode Selection Status:** Bits[3:1] return the counter mode programming.<br>**Bit[3:1] Mode Selected Bit[3:1] Mode Selected**<br>000 0 X11 3<br>001 1 100 4<br>X10 2 101 5 |
| 0 | **Countdown Type Status:** 0=Binary countdown; 1=Binary coded decimal (BCD) countdown. |

#### 3.3.2.3. Counter Access Ports Register

Register Location: Counter 0, System Timer—040h
Counter 1, Refresh Request—041h
Counter 2, Speaker Tone—042h
Default Value: All bits undefined
Attribute: Read/Write

Each of these I/O ports is used for writing count values to the Count Registers; reading the current count value from the counter by either an I/O read, after a counter-latch command, or after a Read Back Command; and reading the status byte following a Read Back Command.

| Bit | Description |
|---|---|
| 7:0 | **Counter Port bit[x].** Each counter I/O port address is used to program the 16-bit Count Register. The order of programming, either LSB only, MSB only, or LSB then MSB, is defined with the Interval Counter Control Register at I/O port address 043h. The counter I/O port is also used to read the current count from the Count Register, and return the status of the counter programming following a Read Back Command. |

### 3.3.3. INTERRUPT CONTROLLER REGISTERS

The MPIIX contains an ISA-compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The interrupt registers control the operation of the interrupt controller and can be accessed from the PCI Bus via PCI I/O space.

#### 3.3.3.1. ICW1—Initialization Command Word 1 Register

Register Location: INT CNTRL-1—020h
INT CNTRL-2—0A0h
Default Value: All bits undefined
Attribute: Write Only

A write to Initialization Command Word 1 starts the interrupt controller initialization sequence. Addresses 020h and 0A0h are referred to as the base addresses of CNTRL-1 and CNTRL-2, respectively. An I/O write to the CNTRL-1 or CNTRL-2 base address with bit 4 equal to 1 is interpreted as ICW1. For MPIIX-based ISA systems, three I/O writes to "base address + 1" must follow the ICW1. The first write to "base address + 1" performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence during which the following automatically occur:

1. The Interrupt Mask register is cleared.
2. IRQ7 input is assigned priority 7.
3. The slave mode address is set to 7.
4. Special Mask Mode is cleared and Status Read is set to IRR.
5. If IC4 was set to 0, then all functions selected by ICW4 are set to 0. However, ICW4 must be programmed in the MPIIX implementation of this interrupt controller, and IC4 must be set to a 1.

| Bit | Description |
|---|---|
| 7:5 | **ICW/OCW select:** These bits should be 000 when programming the MPIIX. |
| 4 | **ICW/OCW select:** Bit 4 must be a 1 to select ICW1. After the fixed initialization sequence to ICW1, ICW2, ICW3, and ICW4, the controller base address is used to write to OCW2 and OCW3. Bit 4 is a 0 on writes to these registers. A 1 on this bit at any time will force the interrupt controller to interpret the write as an ICW1. The controller will then expect to see ICW2, ICW3, and ICW4. |
| 3 | **Edge/Level Bank Select (LTIM):** This bit is disabled. Its function is replaced by the Edge/Level Triggered Control (ELCR) Registers. |
| 2 | **ADI:** Ignored for the MPIIX. |
| 1 | **Single or Cascade (SNGL):** This bit must be programmed to a 0. |
| 0 | **ICW4 Write Required (IC4):** This bit must be set to a 1. |

#### 3.3.3.2. ICW2—Initialization Command Word 2 Register

Register Location: INT CNTRL-1—021h
INT CNTRL-2—0A1h
Default Value: All bits undefined
Attribute: Write Only

ICW2 is used to initialize the interrupt controller with the five most significant bits of the interrupt vector address.

| Bit | Description |
|---|---|
| 7:3 | **Interrupt Vector Base Address:** Bits [7:3] define the base address in the interrupt vector table for the interrupt routines associated with each interrupt request level input. |
| 2:0 | **Interrupt Request Level:** Must be programmed to all 0s. |

#### 3.3.3.3. ICW3—Initialization Command Word 3 Register

Register Location: INT CNTRL-1—021h
Default Value: All bits undefined
Attribute: Write Only

The meaning of ICW3 differs between CNTRL-1 and CNTRL-2. On CNTRL-1, the master controller, ICW3 indicates which CNTRL-1 IRQ line physically connects the INTR output of CNTRL-2 to CNTRL-1.

| Bit | Description |
|---|---|
| 7:3 | **Reserved:** Must be programmed to all 0s. |
| 2 | **Cascaded Mode Enable:** This bit must be programmed to 1 selecting cascade mode. |
| 1:0 | **Reserved:** Must be programmed to all 0s. |

#### 3.3.3.4. ICW3—Initialization Command Word 3 Register

Register Location: INT CNTRL-2—0A1h
Default Value: All bits undefined
Attribute: Write Only

On CNTRL-2 (the slave controller), ICW3 is the slave identification code broadcast by CNTRL-1.

| Bit | Description |
|---|---|
| 7:3 | **Reserved:** Must be programmed to all 0s. |
| 2:0 | **Slave Identification Code:** Must be programmed to 010b. |

#### 3.3.3.5. ICW4—Initialization Command Word 4 Register

Register Location: INT CNTRL-1—021h
INT CNTRL-2—0A1h
Default Value: 01h
Attribute: Write Only

Both MPIIX interrupt controllers must have ICW4 programmed as part of their initialization sequence.

| Bit | Description |
|---|---|
| 7:5 | **Reserved:** Must be programmed to all 0s. |
| 4 | **Special Fully Nested Mode (SFNM):** Bit 4, SFNM, should normally be disabled by writing a 0 to this bit. If SFNM=1, the special fully nested mode is programmed. |
| 3 | **Buffered mode (BUF):** Must be programmed to 0 selecting non-buffered mode. |
| 2 | **Master/Slave in Buffered Mode:** Should always be programmed to 0. Bit not used. |
| 1 | **AEOI (Automatic End of Interrupt):** This bit should normally be programmed to 0. This is the normal end of interrupt. If this bit is 1, the automatic end of interrupt mode is programmed. |
| 0 | **Microprocessor Mode:** Must be programmed to 1 indicating an Intel Architecture-based system. |

#### 3.3.3.6. OCW1—Operational Control Word 1 Register

Register Location: INT CNTRL-1—021h
INT CNTRL-2—0A1h
Default Value: 00h
Attribute: Read/Write

OCW1 sets and clears the mask bits in the Interrupt Mask Register (IMR). Each interrupt request line may be selectively masked or unmasked any time after initialization. The IMR stores the interrupt line mask bits. The IMR operates on the IRR. Masking of a higher priority input does not affect the interrupt request lines of lower priority. Unlike status reads of the ISR and IRR, for reading the IMR, no OCW3 is needed. The output data bus contains the IMR when an I/O read is active and the I/O address is 021h or 0A1h (OCW1). All writes to OCW1 must occur following the ICW1-ICW4 initialization sequence, since the same I/O ports are used for OCW1, ICW2, ICW3 and ICW4.

| Bit | Description |
|---|---|
| 7:0 | **Interrupt Request Mask (Mask [7:0]).** When a 1 is written to any bit in this register, the corresponding IRQx line is masked. For example, if bit 4 is set to a 1, then IRQ4 is masked. Interrupt requests on IRQ4 do not set channel 4's interrupt request register (IRR) bit as long is the channel is masked. When a 0 is written to any bit in this register, the corresponding IRQx is unmasked. Note that masking IRQ2 on CNTRL-1 also masks the interrupt requests from CNTRL-2, which is physically cascaded to IRQ2. |

#### 3.3.3.7. OCW2—Operational Control Word 2 Register

Register Location: INT CNTRL-1—020h
INT CNTRL-2—0A0h
Default Value: Bit[4:0]=undefined, Bit[7:5]=001
Attribute: Write Only

OCW2 controls both the Rotate Mode and the End of Interrupt Mode. Following a PCIRST or ICW initialization, the controller enters the fully nested mode of operation. Both rotation mode and specific EOI mode are disabled following initialization.

| Bit | Description |
|---|---|
| 7:5 | **Rotate and EOI Codes.** R, SL, EOI – These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations is listed above under the bit definition.<br><br>**Bits [7:5] Function**<br><br>001 Non-specific EOI command<br>011 Specific EOI Command<br>101 Rotate on Non-Specific EOI Command<br>100 Rotate in Auto EOI Mode (Set)<br>000 Rotate in Auto EOI Mode (Clear)<br>111 *Rotate on Specific EOI Command<br>110 *Set Priority Command<br>010 No Operation<br><br>* L0–L2 Are Used |
| 4:3 | **OCW2 Select.** Must be programmed to 00 selecting OCW2. |
| 2:0 | **Interrupt Level Select (L2, L1, L0).** L2, L1, and L0 determine the interrupt level acted upon when the SL bit is active. A simple binary code, outlined above, selects the channel for the command to act upon. When the SL bit is inactive, these bits do not have a defined function; programming L2, L1 and L0 to 0 is sufficient in this case.<br><br>**Bit [2:0] Interrupt Level Bit [2:0] Interrupt Level**<br><br>000 IRQ 0(8) 100 IRQ 4(12)<br>001 IRQ 1(9) 101 IRQ 5(13)<br>010 IRQ 2(10) 110 IRQ 6(14)<br>011 IRQ 3(11) 111 IRQ 7(15) |

#### 3.3.3.8. OCW3—Operational Control Word 3 Register

Register Location: INT CNTRL-1—020h
INT CNTRL-2—0A0h
Default Value: Bit[6,0]=0, Bit[7,4:2]=undefined, Bit[5,1]=1
Attribute: Read/Write

OCW3 serves three important functions—Enable Special Mask Mode, Poll Mode control, and IRR/ISR register read control.

| Bit | Description |
|---|---|
| 7 | **Reserved**. Must be 0. |

| Bit | Description |
|---|---|
| 6 | **Special Mask Mode (SMM):** If ESMM=1 and SMM=1, the interrupt controller enters Special Mask Mode. If ESMM=1 and SMM=0, the interrupt controller is in normal mask mode. When ESMM=0, SMM has no effect. |
| 5 | **Enable Special Mask Mode (ESMM):** 1=Enable SMM bit; 0=Disable SMM bit. |
| 4:3 | **OCW3 Select:** Must be programmed to 01 selecting OCW3. |
| 2 | **Poll Mode Command:** 0=Disable Poll Mode Command. When bit 2=1, the next I/O read to the interrupt controller is treated as an interrupt acknowledge cycle indicating highest priority request. |
| 1:0 | **Register Read Command:** Bits [1:0] provide control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). When bit 1=0, bit 0 doesl not affect the register read selection. When bit 1=1, bit 0 selects the register status returned following an OCW3 read. If bit 0=0, the IRR will be read. If bit 0=1, the ISR will be read. Following ICW initialization, the default OCW3 port address read will be "read IRR". To retain the current selection (read ISR or read IRR), always write a 0 to bit 1 when programming this register. The selected register can be read repeatedly without reprogramming OCW3. To select a new status register, OCW3 must be reprogrammed prior to attempting the read.<br><br>**Bit[1:0] Function**<br>00 No Action<br>01 No Action<br>10 Read IRQ Register<br>11 Read IS Register |

#### 3.3.3.9. ELCR1—Edge/Level Triggered Register

Register Location: INT CNTRL-1—4D0h
Default Value: 00h
Attribute: Read/Write

ELCR1 register allows IRQ3 - IRQ7 to be edge or level programmable on an interrupt by interrupt basis. IRQ0, IRQ1 and IRQ2 are not programmable and are always edge sensitive.

| Bit | Description |
|---|---|
| 7 | **IRQ7 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 6 | **IRQ6 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 5 | **IRQ5 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 4 | **IRQ4 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 3 | **IRQ3 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 2:0 | **Reserved:** Must be 0. |

#### 3.3.3.10. ELCR2—Edge/Level Triggered Register

Register Location: INT CNTRL-2—4D1h
Default Value: 00h
Attribute: Read/Write

ELCR2 register allows IRQ[15,14,12:9] to be edge or level programmable on an interrupt by interrupt basis. Note that, IRQ[13,8#] are not programmable and are always edge sensitive.

| Bit | Description |
|---|---|
| 7 | **IRQ15 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 6 | **IRQ14 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 5 | **Reserved:** Must be 0. |
| 4 | **IRQ12 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 3 | **IRQ11 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 2 | **IRQ10 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 1 | **IRQ9 ECL:** 0 = edge triggered mode; 1 = level sensitive mode. |
| 0 | **Reserved:** Must be 0. |

### 3.3.4. RESET EXTENDED I/O-BUS IRQ12 AND IRQ1 REGISTER

Register Location: 60h
Default Value: N/A
Attribute: Read only

This register clears the mouse interrupt function and the keyboard interrupt (IRQ1). Reads to this address are monitored by the MPIIX. When the mouse interrupt function is enabled (FDC Enable Register), the mouse interrupt function is provided on the IRQ12/M input signal. In this mode, a mouse interrupt generates an interrupt through IRQ12 to the Host CPU. A read of 60h releases IRQ12. Reads/writes flow through to the Extended I/O Bus. A read of this address always clears the keyboard interrupt (IRQ1).

| Bit | Description |
|---|---|
| 7:0 | **Reset IRQ12 and IRQ1**. No specific pattern. A read of address 60h executes the command. |

### 3.3.5. NMI REGISTERS

The NMI logic incorporates two different 8-bit registers. The CPU reads the NMISC Register to determine the NMI source (bits set to a 1). After the NMI interrupt routine processes the interrupt, software clears the NMI status bits by setting the corresponding enable/disable bit to a 1. The NMI Enable and Real-Time Clock Register can mask the NMI signal and disable/enable all NMI sources.

To ensure that all NMI requests are serviced, the NMI service routine software flow should be as follows:

1. NMI is detected by the processor on the rising edge of the NMI input.
2. The processor will read the status stored in port 061h to determine what sources caused the NMI. The processor may then set to 0 the register bits controlling the sources that it has determined to be active. Between the time the processor reads the NMI sources and sets them to a 0, an NMI may have been generated by another source. The level of NMI will then remain active. This new NMI source will not be recognized by the processor because there was no edge on NMI.
3. The processor must then disable all NMIs by setting bit 7 of port 070h to a 1 and then enable all NMIs by setting bit 7 of port 070h to a 0. This will cause the NMI output to transition low then high if there are any pending NMI sources. The CPU's NMI input logic will then register a new NMI.

#### 3.3.5.1. NMISC—NMI Status and Control Register

Register Location: 061h
Default Value: 00h
Attribute: Read/Write

This register reports the status of different system components, control the output of the speaker counter (Counter 2), and gate the counter output that drives the SPKR signal.

| Bit | Description |
|---|---|
| 7 | **SERR# NMI Source Status—RO:** Bit 7 is set if a system board agent (PCI devices or main memory) detects a system board error and pulses the PCI SERR# line. This interrupt source is enabled by setting bit 2 to 0. To reset the interrupt, set bit 2 to 0 and then set it to 1. When writing to port 061h, bit 7 must be 0. |
| 6 | **Reserved:** Read as 0. |
| 5 | **Timer Counter 2 OUT Status—RO:** The Counter 2 OUT signal state is reflected in bit 5. The value on this bit following a read is the current state of the Counter 2 OUT signal. Counter 2 must be programmed following a CPURST for this bit to have a determinate value. When writing to port 061h, bit 5 must be a 0. |
| 4 | **Refresh Cycle Toggle—RO:** The Refresh Cycle Toggle signal toggles from either 0 to 1 or 1 to 0 following every refresh cycle. When writing to port 061h, bit 4 must be a 0. |
| 3 | **Reserved:** Read as 0. |
| 2 | **PCI SERR# Enable—R/W:** 1=Clear and Disable; 0=Enable. |
| 1 | **Speaker Data Enable—R/W:** 0=SPKR output is 0; 1= the SPKR output is the Counter 2 OUT signal value. |
| 0 | **Timer Counter 2 Enable—R/W:** 1=Enable; 0=Disable. |

#### 3.3.5.2. NMI Enable and Real-Time Clock Address Register

Register Location: 070h
Default Value: Bit[6:0]=undefined, Bit 7=1
Attribute: Write Only

This port is shared with the real-time clock. Do not modify the contents of this register without considering the effects on the state of the other bits. Reads and writes to this register address flow through to the Extended I/O Bus.

| Bit | Description |
|---|---|
| 7 | **NMI Enable.** 1=Disable all NMI sources. 0=Enable the NMI interrupt. |
| 6:0 | **Real Time Clock Address.** Used by the Real Time Clock on the Base I/O component to address memory locations. Not used for NMI enabling/disabling. |

#### 3.3.5.3. Coprocessor Error Register

Register Location: F0h
Default Value: Undefined
Attribute: Write only

Writing to this register causes the MPIIX to assert IGNNE#. The MPIIX also negates IRQ13 (internal to the MPIIX). Note, that IGNNE# is not asserted unless FERR# is active. Reads/writes flow through to the Extended I/O Bus.

| Bit | Description |
|---|---|
| 7:0 | No special pattern required: A write to address F0h executes the command. |

#### 3.3.5.4. RC—Reset Control Register

I/O Address: CF9h
Default Value: 00h
Attribute: Read/Write

Bits 1 and 2 in this register are used by the MPIIX to generate a hard reset or a soft reset. To perform a proper reset, bit 2 should be cleared when writing the type of reset (bit 1) to be performed. Then bit 2 should be set to initiate the reset. To perform a soft (hard) reset, first a 00h (02h) is written to CF9h. A second write of 04h (06h) is written to initiate the soft (hard) reset.

| Bit | Description |
|---|---|
| 7:4 | **Reserved.** |
| 3 | **Reserved.** |
| 2 | **Reset CPU (RCPU):** This bit is used to initiate (transitions from 0 to 1) a hard reset (bit 1 in this register is set to 1) or a soft reset to the CPU. During a hard reset, the MPIIX asserts CPURST, PCIRST#, and RSTDRV. The MPIIX initiates a hard reset when this register is programmed for a hard reset or PWROK is asserted. This bit cannot be read as a 1. |

| Bit | Description |
|---|---|
| 1 | **System Reset (SRST):** This bit is used in conjunction with bit 2 in this register to initiate a hard reset. When SRST =1, the MPIIX initiates a hard reset to the CPU when bit 2 in this register transitions from 0 to 1. When SRST=0, the MPIIX initiates a soft reset when bit 2 in this register transitions from 0 to 1. |
| 0 | **Reserved.** |

#### 3.3.5.5. Port 92 Register

Register Location: 92h
Default Value: 00h
Attribute: Read/Write

This register controls the ALTA20 signal and initiates a fast soft reset by generating an INIT to the CPU.

| Bit | Description |
|---|---|
| 7:4 | **Reserved.** Returns 0s when read. |
| 3 | **Power On Password Protection:** Writing a 1 to this bit enables the power-on password protection by inhibiting accesses to the RTC CMOS RAM locations 38–3Fh. This is accomplished by not issuing an RTCCS# for accesses to the data port (71h, 73h, 75h, 77h) after 38–3Fh has been written to the index port (70h, 72h, 74h, 76h). This bit can only be cleared to 0 by turning off system power (PCIRST# asserted) and then turning on system power. |
| 2 | **Reserved.** Returns 0 when read. |
| 1 | **FASTA20.** 0=ALTA20 signal is driven low. 1=ALTA20 signalis driven high. This signal is externally OR'd with the Keyboard Controller A20 signal. This signal is connected to CPU for support of real mode compatible software. |
| 0 | **FASTINIT.** This bit provides a fast software system reset function and is an alternate means to reset the system CPU to effect a mode switch from Protected Virtual Address Mode to the Real Address Mode. FASTINIT provides a faster way of invoking a reset than is provided by the Keyboard controller. Setting this bit to a 1 causes the INIT signal to pulse active (high) for approximately 16 PCI Clocks. Before another INIT pulse can be generated, this bit must be set to 0. |

## 3.4. Advanced Power Management Registers

The APMS and APMC registers are in normal I/O space. The software must only access them with 8-bit accesses.

### 3.4.1. APMC—ADVANCED POWER MANAGEMENT CONTROL PORT

I/O Address: 0B2h
Default Value: 00h
Attribute: Read/Write

This is a Control port used to pass data between the OS and the SMI handler. Writes to this port cause SMI# to be asserted if the APMC_SMI_EN bit is set. If the STPCLK# function is enabled, reads from this port cause STPCLK# to be asserted.

| Bit | Description |
|---|---|
| 7:0 | **APMC[7:0].** APM Control Port. Read/writeable at system I/O address 0B2h. Used to pass an APM command between the OS and the SMI handler. Writes to this port not only store data in the APMC register, but also generates an SMI when the APMC_SMI_EN bit is set. Reads to this port will not generate an SMI. If STPCLK# is enabled in the STPCLK_MODE configuration register, a read from the APMC will cause STPCLK# to be asserted. |

### 3.4.2. APMS—ADVANCED POWER MANAGEMENT STATUS PORT

I/O Address: 0B3h
Default Value: 00h
Attribute: Read/Write

This is a Status port used to pass data between the OS and the SMI handler.

| Bit | Description |
|---|---|
| 7:0 | **APMS[7:0].** APM Status Port. Read/writeable at system address 0B3h. Used to pass data between the OS and the SMI handler. |

# 4.0. FUNCTIONAL DESCRIPTION

This section describes each of the major functions of the MPIIX including the memory and I/O address map, PCI interface, Extended I/O Bus interface, DMA controller, IDE interface, interval timer, interrupts, power management, and reset.

## 4.1. Memory And I/O Address Map

The MPIIX interfaces to two system buses—PCI Bus and the Extended I/O Bus. The MPIIX provides positive decode for certain I/O and memory space accesses on the PCI bus as described in this section. The MPIIX does not support other bus masters on the Extended I/O Bus and, thus, does not provide address decoding on this bus.

### 4.1.1. I/O ACCESSES

The MPIIX provides positive decode for PCI accesses to the PCI Configuration Registers, power management registers, and the ISA-Compatible registers. For details concerning accessing these registers, see the Register Description section.

PCI I/O access that are claimed by MPIIX, and are not part of the MPIIX internal register set, are forwarded to the Extended I/O Bus. These accesses are targeted for peripherals residing on the Extended I/O Bus.

### 4.1.2. BIOS MEMORY ACCESS

The MPIIX supports 256 Kbytes of BIOS space. This includes the normal 128 Kbyte space plus an additional 128 Kbyte BIOS space (known as the extended BIOS area). The BIOSE Register provides BIOS space access control. Access to the lower 64 Kbyte block of the 128 Kbyte space and the extended BIOS space can be individually enabled/disabled. In addition, write protection can be programmed for the entire BIOS space.

The MPIIX only claims PCI memory cycles in the BIOS range, and forwards the cycles to the Extended I/O Bus. The 128 Kbyte BIOS memory space is located at 000E0000–000FFFFFh (top of 1 Mbyte), and is aliased at FFFE0000h (top of 4 GBytes). This 128 Kbyte block is split into two 64 Kbyte blocks. Accesses to the top 64 Kbytes (000F0000–000FFFFFh) will be forwarded to the Extended I/O bus and BIOSCS# is always generated. Accesses to the bottom 64 Kbytes (000E0000–000EFFFFh) are forwarded to the Extended I/O bus and BIOSCS# generated based on the status of bits [6:4] in the BIOSE Register. Accesses to the aliased region at the top of 4 GBytes (FFFF0000h–FFFFFFFFh) will be forwarded to the Extended I/O bus and BIOSCS# will always be generated. Accesses to the aliased region at the top of 4 GBytes (FFFE0000–FFFEFFFFh) are forwarded to the Extended I/O bus and BIOSCS# generated based on the status of bits [6:4] in the BIOSE Register.

The additional 128 Kbyte region resides at FFFC0000–FFFDFFFFh. Memory accesses within this region are forwarded to the Extended I/O bus and BIOSCS# is generated if bit 7 in the BIOSE Register is enabled.

#### 4.1.3. PERIPHERAL CHIP SELECTS

The MPIIX generates chip selects for PCI initiated cycles to the BIOS, keyboard controller and real time clock. The MPIIX also generates a programmable chip select for a peripheral device and generates RTCALE (address latch enable) for the RTC. The chip selects are generated combinatorially from the SA(15:0) bus.

Table 3 lists the I/O and memory addresses that are positively decoded by MPIIX and, where applicable, shows the corresponding chip select generation. Chip selects and optional address ranges can be enabled/disabled through their corresponding configuration register. In general, the addresses shown in the table do not reside in the MPIIX itself. Addresses 60h and 70h are exceptions since particular bits from these registers reside in the MPIIX.

**Table 3. Extended I/O Bus Decode**

| Address | Type | Name | Encoded Chip Select |
|---|---|---|---|
| 0060h, 62h, 64h, 66h | R/W | Keyboard Controller | KBCCS# |
| 70h, 72h, 74h, 76h | W | Real Time Clock Address | RTCALE# |
| 0071h, 73h, 75h, 77h | R/W | Real Time Clock Data | RTCCS# |
| 0201h | R/W | Audio Port | |
| 02x0–02xFh | R/W | Audio Port x = 2, 3, 4, or 5 | |
| 0388–038Bh | R/W | Audio Port | |
| 03BC–03BFh, 07BC–07BEh | | Parallel Port, LPT1 or EPP/ECP | |
| 0378–037Fh, 0778–077Ah | | Parallel Port, LPT2 or EPP/ECP | |
| 0278–027Fh, 0678–067Ah | | Parallel Port, LPT3 or EPP/ECP | |
| 03F8–03FFh | R/W | Serial Port, COM1 | |
| 02F8–02FFh | R/W | Serial Port, COM2 | |
| 03E8–03EFh | R/W | Serial Port, COM3 | |
| 02E8–02EFh | R/W | Serial Port, COM4 | |
| 03F0–03F5h, 03F7h | R/W | FDC Primary | |
| 0370–0375h, 0377h | R/W | FDC Secondary | |
| 03F6h | R/W | IDE Primary | |
| 0376h | R/W | IDE Secondary | |
| ADDR (10-bit) | R/W | I/O Config Address (2-byte range) | |
| ADDR + MASK | R/W | Prog Chip Select PCS# | PCS# |
| ADDR + MASK | R/W | Prog I/O Range #1 | |
| ADDR + MASK | R/W | Prog I/O Range #2 | |
| ADDR + MASK | R/W | Prog I/O Range #3 | |
| ADDR + MASK | R/W | Prog I/O Range #4 | |
| ADDR + MASK | R/W | Prog I/O Range #5 | |

| Address | Type | Name | Encoded Chip Select |
|---|---|---|---|
| 000E0000–000E3FFFh | R/W | Kanji BIOS (at top of 1MB) | BIOSCS# (optional) |
| FFFE0000–FFFE3FFFh | | Kanji BIOS (at top of 4 GB) | BIOSCS# (optional) |
| 000E0000–000FFFFFh | R/W | BIOS Memory (128 KB at top of 1MB) | BIOSCS# |
| FFFE0000–FFFFFFFFh | R/W | BIOS Memory (aliased 128 KB region at top of 4 GB) | BIOSCS# |
| FFFC0000–FFFDFFFFh | R/W | BIOS Memory (additional 128 KB region at top of 4 GB) | BIOSCS# |

## 4.2. PCI Interface

The MPIIX incorporates a fully PCI Bus compatible master and slave interface. As a PCI master, the MPIIX runs cycles on behalf of DMA. As a PCI slave, the PIIX accepts cycles intitiated by PCI masters targeted for the MPIIX's internal registers or the Extended I/O Bus. The MPIIX directly supports the PCI interface running at either 25 Mhz, 30 Mhz, or 33 MHz.

Bus commands indicate to the slave the type of transaction the master is requesting. Bus Commands are encoded on the C/BE[3:0]# lines during the address phase of a PCI cycle.

### 4.2.1. TRANSACTION TERMINATION

The MPIIX supports both Master-initiated Termination as well as Target-initiated Termination.

**MPIIX As Master—Master-Initiated Termination**

The MPIIX supports three forms of Master-initiated Termination:

1. Normal termination of a completed transaction.
2. Normal termination of an incomplete transaction due to timeout.
3. Abnormal termination due to no slave responding to the transaction (master abort).

**MPIIX As a Master—Response to Target-Initiated Termination**

MPIIX's response as a master-to-target-termination, including target abort, retry, and disconnect.

**MPIIX As a Target—Target-Initiated Termination**

The MPIIX supports three forms of Target-initiated Termination (target abort, retry, and disconnect).

### 4.2.2. PARITY SUPPORT

As a master, the MPIIX generates address parity for read and write cycles, and data parity for write cycles. As a slave, the MPIIX generates data parity for read cycles. The MPIIX does not check parity and does not generate SERR#.

#### 4.2.3. PCI ARBITRATION

The MPIIX requests the use of the PCI Bus on behalf of Extended I/O DMA or PCI DMA. The MPIIX arbitrates for the PCI Bus through the PHOLD# and PHLDA# signals. The PCI arbiter is assumed to be integrated in the host-to-PCI bridge.

Extended I/O Bus DMA slave devices assert DREQ2 or MDRQ[2:0]# to gain access to the PCI Bus. The MPIIX in response asserts PHOLD# to the PCI arbiter. For Extended I/O DMA devices, MPIIX keeps DACK2# or MDAK[2:0]# negated until the MPIIX has ownership of the PCI Bus and Memory. The PCI arbiter asserts PHLDA# to the MPIIX after its PCI buffers are emptied to PCI bus. The MPIIX gives ownership of the bus (PCI and Memory) to the DMA controller after sampling PHLDA# asserted.

PCI DMA agents will use the PC/PCI REQ[A,B]# signals to gain access to the PCI bus. A PCI DMA agent can be a PCI DMA slave, or a PCI expansion bridge that requests ownership of the PCI bus on behalf of an ISA DMA slave or ISA master. The MPIIX obtains the PCI bus through the PHOLD#/PHLDA# sequence as described for Extended I/O bus DMA. MPIIX then grants the bus to the PCI DMA agent by driving the appropriate GNT[A,B]# signal as described later in the DMA chapter.

#### 4.2.4. PCI CLOCK CONTROL (CLKRUN#)

MPIIX contains extensive power management capabilities. To provide power management on the PCI Bus, MPIIX implements PCI clock control using the CLKRUN# signal. The three main states in the clocking protocol are:

- **Clock Running**: The clock is running and the bus is operational.
- **About to Stop:** The central resource has indicated on the CLKRUN# line that the clock is about to stop.
- **Clock Stopped:** The clock is stopped with CLKRUN# being monitored for a restart.

MPIIX serves as the CLKRUN# Central Resource in accordance with the PCI Local Bus PCI Mobile Design Guide, Revision 1.0.

## 4.3. Extended I/O Bus

The MPIIX incorporates a subset of the ISA Bus called the Extended I/O Bus that is designed to interface to 5V peripherals that reside on the main system board. All PCI cycles intended for the Extended I/O bus are positively decoded to allow a docking station bridge to claim subtractively decoded PCI cycles. The MPIIX does not support bus masters on the Extended I/O Bus. The Extended I/O Bus provides the following support for motherboard devices:

- IDE (16-bit) Interface with isolation buffer control.
- DMA between Extended I/O devices and PCI memory with steerable DMA channels (3).
- All ISA IRQ signals and 1 steerable Interrupt.
- BIOS ROM or Flash 256 Kbyte (8-bit) Interface.
- 6 Programmable I/O Decode Ranges.
- Audio (8-bit I/O and 8- or 16-bit DMA) Interface.
- Super-I/O (8-bit I/O and 8- or 16-bit DMA) Interface
- Keyboard Controller (8-bit) Interface.
- Real Time Clock (8-bit) Interface.
- Programmable chip select with isolation buffer control.

The Extended I/O Bus interface also provides byte swap logic, I/O recovery support, wait-state generation, SYSCLK generation, and standard ISA port 92h Fast A20Gate and Fast CPU INIT.

All PCI cycles intended for the Extended I/O bus are positively decoded to allow a docking station bridge to claim subtractively decoded PCI cycles. There are 5 programmable I/O address ranges for positive decode special function I/O ports (PAC[5:1] Registers).

Access to the enabled BIOS range generates the BIOSCS# signal, if enabled in the BIOS Enable Register. This is an 8-bit memory access and it is the only memory access supported on the Extended I/O bus. Access to the standard 8-bit Keyboard Controller port generates the KBCS# signal. Access to the standard 8-bit Real Time Clock ports generates an RTCCS#, RTCALE.

When the Programmable Chip Select Address Range is enabled, any access to that range generates the PCS# signal. This PCS# signal can be used as the output enable for the address and data isolation buffer to the PCS# port. The SDIR signal is used to control the direction of the data transceiver. When the peripheral connected to the PCS# port is powered down or not in use, the buffers are tri-stated.

The PCS# signal can also be generated (if enabled) for Programmable Address Ranges 1 and 2. This allows the system designer to generate chip selects for up to 3 non-contiguous I/O ranges using simple address decoding.

MPIIX supports an audio interface by providing a steerable DMA REQ/ACK pair, a steerable interrupt and decoding of the Audio chip ports. MPIIX supports Soundblaster Pro compatibility through the use of an external audio chip.

MPIIX supports mulit-function I/O devices including 8-bit access to standard serial ports, parallel ports, and Floppy Disk Controller. A dedicated DMA signal pair (DREQ2/DACK2#) is used for the Floppy Controller interface.

### 4.3.1. EXTENDED I/O BUS CYCLES FOR MPIIX AS A MASTER (PCI MASTER INITIATED)

The Extended I/O Bus interface supports the following types of cycles:

- PCI master initiated I/O cycle to positively decoded peripherals.
- PCI master initiated I/O cycle to the IDE interface.
- PCI master initiated memory cycle to 256 Kbyte BIOS region only.
- DMA compatible cycles between PCI memory (include main system DRAM) and Extended I/O Bus I/O
- Enhanced DMA cycles between PCI memory and Extended I/O Bus I/O

The EIO bus supports only 8-bit I/O ports. If a multi-byte PCI access is performed to a device on the EIO bus, the cycle will be assembled from (read) or disassembled to (write) the sequential addresses in the PCI cycle.

The MPIIX generates the Extended I/O Bus system clock (SYSCLK). SYSCLK is a divided down version of the PCICLK and has a frequency of 7.5 or 8.33 MHz, depending on the PCICLK frequency. The clock divisor value is determined according to the strapping options as described in the SYSCLK signal description. MPIIX has an internal pull-up resistor that sets the default divisor to 4. (This resister is disabled after reset.) When MPIIX stops the PCICLK, SYSCLK is also stopped.

The MPIIX adds wait-states to MPIIX master cycles (not including DMA) to the Extended I/O Bus, if IOCHRDY is sampled active. Wait states will be added as long as IOCHRDY is negated. The MPIIX shortens MPIIX master cycles (not including DMA) to the Extended I/O Bus, if ZEROWS# is sampled active. Note that, If IOCHRDY and ZEROWS# are sampled active at the same time, IOCHRDY takes precedence and wait-states are added.

The I/O recovery mechanism in the MPIIX is used to add additional recovery delay between PCI Master originated 8-bit I/O cycles to the Extended I/O Bus. See the EXRT Register description for details.

MPIIX provides the Fast A20Gate bit 1 in the Port 92h Register and ALTA20M output signal. This signal is externally OR'd with the A20M from the Keyboard Controller and then sent to the CPU. MPIIX also provides the Fast CPU INIT bit 0 in the Port 92h Register and the INIT signal (Figure 2). This signal is externally OR'd with the RC# signal from the Keyboard Controller.

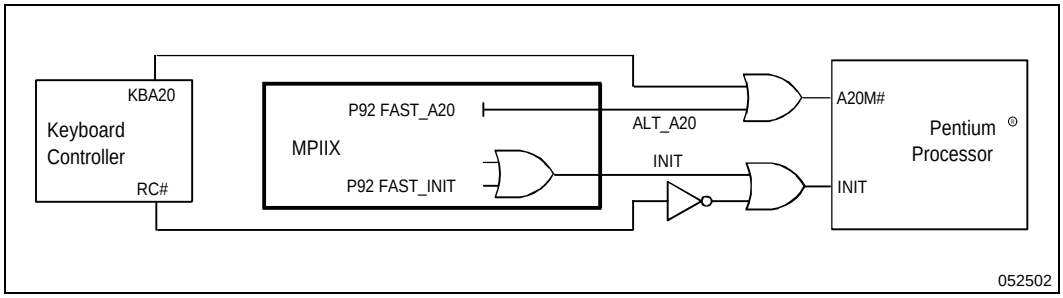



**Figure 2. Fast A20Gate and INIT Connections**

#### 4.3.2. EXTENDED I/O BUS DMA (8-BIT AND 16-BIT TRANSFERS)

The DMA controller in MPIIX will transfer data between PCI memory and Extended I/O bus. Devices on the Extended I/O bus request DMA service through the MDRQ[2:0]/MDAK[2:0]# or DREQ2/DACK2# signal pairs. The MDRQ[2:0]/MDAK[2:0]# signal pairs can be routed to any of the channels on the DMA controller.

For 8-bit DMA transfers, data is read/written on the SD[7:0]. SA[15:0] are driven to 0 which prevents any other I/O devices on the Extended I/O bus from responding to the DMA cycle. MPIIX does not need an AEN signal for systems that only implement 8-bit DMA devices on the Extended I/O bus.

For 16-bit DMA transfers the upper byte of data (SD[15:8]) is multiplexed on the SA[15:8] signal pins. For 16-bit DMA cycles, the MPIIX asserts MDAKx#, IOR# or IOW#, and floats (DMA read) or drives (DMA write) the SA[15:8] signals. For systems that use 16-bit DMA transfers on the Extended I/O bus, the SA[15:8] signals are driven with data by either the MPIIX or the 16-bit DMA device. To prevent 8-bit I/O devices from responding to the address and command from the 16-bit DMA cycle, these 8-bit I/O devices require an AEN signal. The AEN signal can be generated by inverting the MDAKx# signal that is assigned to the 16-bit DMA device.

**NOTE**

For 8-bit I/O read cycles to the 16-bit DMA device, it is possible that the 16-bit DMA device could drive data onto its SD[15:8]. If SA[15:8] are connected directly to the SD[15:8] signals of the 16-bit DMA device, there will be contention. In this case the SD[15:8] signals to/from the 16-bit DMA device will require buffering with a '245 transceiver. The '245 can use MDAKx# for an enable and the MPIIX IOR# signal for the direction. (Note: this is simply a caution since the ISA specification prohibits driving the upper byte on a byte-wide access. However, there have been devices that were in violation of this specification)

## 4.4. DMA Controller

The DMA circuitry incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels (Channels 0-3 and Channels 5-7). DMA Channel 4 is used to cascade the two controllers and will default to cascade mode in the DMA Channel Mode (DCM) Register. The DMA controller for Channels 0-3 is referred to as "DMA-1" and the controller for Channels 4-7 is referred to as "DMA-2".

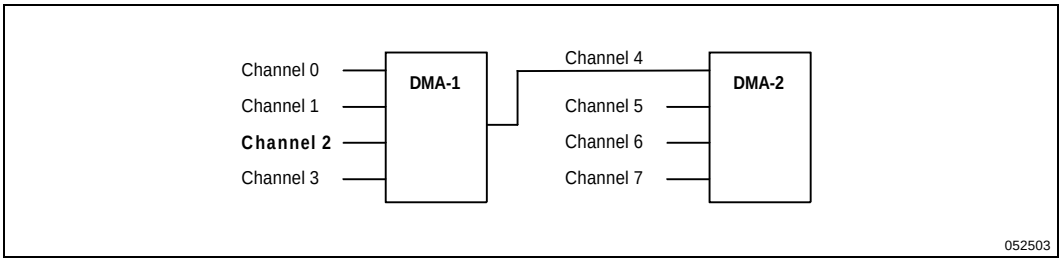



**Figure 3. Internal DMA Controller**

DMA channels [3:0] are hardwired to 8-bit, count-by-bytes transfers, and channels [7:5] are programmable to either 8-bit, count-by-bytes transfers or 16-bit, count-by-words (address shifted) transfers. The MPIIX provides the timing control and data size translation necessary for DMA transfers between PCI memory and the Extended I/O Bus I/O or PCI DMA I/O. ISA Compatible DMA and type F (motherboard devices only) timings are supported.

The MPIIX provides 24-bit addressing in compliance with the ISA Compatible specification. Each channel includes a 16-bit ISA Compatible Current Register which holds the 16 least significant bits of the 24-bit address, and an ISA Compatible Low Page Register which contains the eight most significant bits of address. The DMA controller also permits auto-initialization following a DMA termination.

The DMA controller is at any time either in master mode or slave mode. In master mode, the DMA controller is either servicing a DMA slave's request for DMA cycles, or allowing a 16-bit ISA master (through the use of the PCI DMA Serial Protocol) to use the bus via a cascade mode DMA Channel. NOTE: Masters are not supported on the MPIIX Extended I/O Bus. In slave mode, the MPIIX monitors the PCI bus, decoding and responding to I/O read and write commands that address its registers.

During DMA memory read cycles to the PCI bus, the MPIIX returns undefined data to the Extended I/O Bus, if the PCI cycle is either target aborted or master aborted.

The channels can be programmed for single, block, demand, or cascade transfer modes. Each of the three active transfer modes (single, block, and demand) can perform three different types of transfers (read, write, or verify). Note that memory-to-memory transfers are not supported by the MPIIX.

ISA compatible timing is provided for DMA slave devices that reside on the Extended I/O Bus. All PCI DMA cycles use the 4-byte DMA buffer. The buffer reduces PCI utilization resulting from DMA transfers configured for PCI DMA cycles.

### 4.4.1. TYPE F TIMING

The type F DMA cycles are used with motherboard devices only, through the use of the MDRQ[2:0] and MDAK[2:0]# signals. The type F cycles occur back to back at a minimum repetition rate of 3 SYSCLKs (360ns min). The type F cycles are always performed using the 4-byte DMA buffer.

#### 4.4.2. DMA BUFFER FOR PCI DMA TYPE F TRANSFERS

The DMA buffer referred to above is a 4-byte buffer that is used to reduce the PCI utilization resulting from DMA transfers configured for Type F transfers on the Extended I/O bus and all PCI DMA cycles. The DMA buffer is always used in conjunction with the type F transfers and PCI DMA transfers. For Extended I/O cycles, type F transfers and the DMA buffer are invoked by setting the MDMAx[FAST] register bits for the appropriate channels.

#### 4.4.3. EXTENDED I/O BUS DMA ARBITRATION

PCI masters have default ownership of the Extended I/O Bus. For DMA requests, the MPIIX arbitrates for PCI Bus control through the PCI PHOLD#/PHLDA# signals. If the MPIIX, as a master, performs a PCI transaction on behalf of the DMA controller and that transaction is terminated with a retry, the MPIIX reissues the transaction. The MPIIX always negates PHOLD# for at least two PCICLKs after a DMA channel has been serviced to allow PCI masters access to the Extended I/O bus.

#### 4.4.4. PCI DMA

MPIIX provides support for DMA on PCI using the PC/PCI DMA Protocol. The PCI DMA request/grant pairs (REQ[A:B]# and GNT[A:B]#) can be configured for support of a PCI DMA agent on the same PCI bus that MPIIX is on (PCI bus number 0) or they can be configured to support a PCI DMA Expansion agent on the other side of a secondary bridge such as a PCI-to-ISA bridge. These request grant pairs are configured in the PCIDMA[A,B] Registers (offset 088h and 089h). Note that the PCIDMAE Register must be programmed to indicate that a DMA channel supports a PC/PCI REQ#/GNT# pair rather than an Extended I/O DMA slave.

**PCI DMA agents (PCI Bus number 0):** A PCI DMA master/slave device is configured to request a specific DMA channel (CH[0:3, 5:7]) by selecting the appropriate PC/PCI REQ#/GNT# pair (via the PCIDMA[A,B] Register). For PCI DMA slaves, the slave requests a DMA transfer using PC/PCI REQx# and MPIIX responds by asserting the corresponding PC/PCI GNTx#. The DMA controller then runs the DMA I/O and memory cycles on the PCI bus for the selected channel.

**PCI DMA Expansion Agent (secondary bridge):** The PC/PCI REQ#/GNT# pair can be used by a PCI DMA Expansion agent (secondary bridge) to provide DMA service or ISA Bus master service using the MPIIX DMA controller. When the PC/PCI REQ#/GNT# pair is configured as an expansion agent (via PCIDMA[A,B] Registers), the REQ#/GNT# pair must follow the PC/PCI serial protocol described in the following section.

##### 4.4.4.1. PCI DMA Expansion Protocol

If the expansion bit is set the PCI expansion agent must support the PCI expansion Channel Passing Protocol defined below for both the REQ# and GNT# pins:

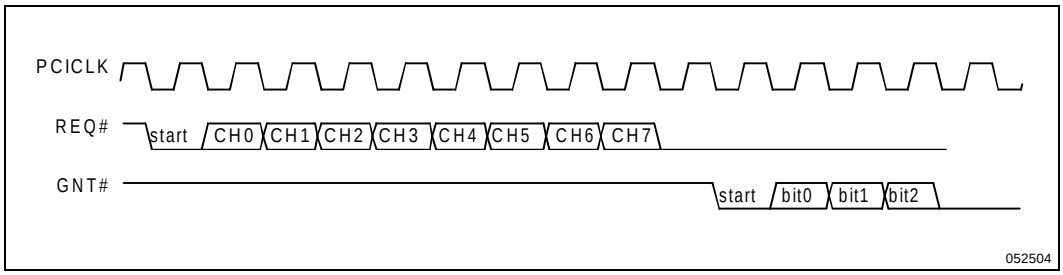



**Figure 4. DMA Serial Channel Passing Protocol**

When a REQ#/GNT# pair has been designated as an expansion REQ#/GNT# pair, the requesting device must encode the channel request information (Figure 4), where CH[0:7] are one clock active high states representing DMA channel requests 0-7 (Note that channel 4 is reserved and must not be asserted). The MPIIX encodes the granted channel on the GNT# line (bits[0:2]). For example bits[0:2]=011 grants DMA channel 6 and bits[0:2]=100 grants DMA channel 1.

All PCI DMA expansion agents MUST use the channel passing protocol described above. They must also operate as follows:

1. If a PCI DMA expansion agent has more than one request active, the agent must resend the request serial protocol after one of the requests has been granted the bus and it has completed its transfer. The expansion device should negate REQ# for two clocks and then transmit the serial channel passing protocol again, even if there are no new requests from the PCI expansion agent to the MPIIX. For example, if a PCI expansion agent had active requests for DMA Channel 1 and a Channel 5, the agent would pass this information to the MPIIX through the expansion channel passing protocol. If, after receiving GNT# (assume for CH5) and having the device finish its transfer (device stops driving request to PCI expansion agent), the agent would then need to re-transmit the expansion channel passing protocol to let the MPIIX know that DMA Channel 1 is still requesting the bus. This must occur, even if DMA Channel 1 is the only request the expansion device has pending.
2. If a PCI DMA expansion agent has a request go inactive before MPIIX asserts GNT#, the agent must resend the expansion channel passing protocol. For example, a PCI expansion agent with DMA Channel 1 and 2 requests pending sends these requests to the MPIIX using the expansion channel passing protocol. If DMA Channel 1 request to the expansion agent goes inactive before a GNT# is received from the MPIIX, the expansion agent must negate its REQ# line for one clock and resend the expansion channel passing information with only DMA Channel 2 active.

   Note that the MPIIX does not support this case because a DREQ going inactive before a DACK# is received is not allowed in the ISA DMA protocol. This requirement is needed to be able to support Plug-n-Play ISA devices that toggle DREQ# lines to determine if those lines are free in the system.
3. If a PCI expansion agent has sent its serial request information and receives a new DMA request before receiving a GNT#, the agent must resend the serial request with the new request active. For example, If a PCI expansion agent has already passed requests for DMA Channel 1 and 2 and receives DREQ 3 active before a GNT# is received, the agent must negate its REQ# line for one clock and resend the expansion channel passing information with all three channels active.

#### 4.4.4.2. PCI DMA Expansion Cycles

MPIIX's support of the Mobile PC/PCI DMA Protocol consists of four types of cycles—Memory to I/O, I/O to Memory, Verify, and ISA master cycles. ISA masters are supported through the use of a DMA Channel that has been programmed for cascade mode. Single Transfer Mode is implicitly supported as the case where the DMA controller negates the DACK#/GNT# signal after one transfer has been completed or the DMA controller toggles DACK# after every transfer. Single transfer mode does not require the requesting device to negate DREQ# after a cycle has completed. Therefore, a PCI DMA agent that uses this mode must also sample the GNT# signal and remove DACK# to the I/O DMA device when GNT# is negated.

The DMA controller generates a two cycle transfer (a load followed by a store) in contrast to the ISA "fly-by" cycle for PC/PCI DMA agents. The memory portion of the cycle generates a PCI memory read or memory write bus cycle with the address representing the selected memory. Refer to the PCI 2.0 specification for timings for the memory portion of a Mobile PC/PCI DMA cycle.

The I/O portion of the DMA cycle generates a PCI I/O cycle to one of four I/O addresses as illustrated in Table 4. Note that these cycles must be qualified by an active GNT# signal to the requesting device.

**Table 4. DMA Cycle vs I/O Address**

| DMA Cycle Type | DMA I/O Address | TC (A2) | PCI Cycle Type |
|---|---|---|---|
| Normal | 00h | 0 | I/O Read/Write |
| Normal TC | 04h | 1 | I/O Read/Write |
| Verify | 0C0h | 0 | I/O Read |
| Verify TC | 0C4h | 1 | I/O Read |

For PCI DMA cycles, the I/O address indicates the type of DMA cycle taking place (whether its a normal or a verify cycle and if the cycle is the last transfer of the buffer). Note that the A2 address line is encoded as the terminal count signal for PCI cycles. A2 is asserted during a PCI I/O cycle to indicate the last DMA transfer. To ensure that non Mobile PC/PCI compliant PCI I/O devices do not confuse Mobile PC/PCI DMA cycles for normal I/O cycles, the addresses used by the PCI DMA cycles correspond to the slave addresses of the Mobile PC/PCI DMA controller.

All PCI DMA I/O ports must be dword aligned and can be either byte or word in size. Thus, PCI DMA I/O ports must always be connected to the lower data lines of the PCI data bus (Table 5). The byte enables also reflect the cycle width during the I/O portion of a PCI DMA cycle (Table 6).

**Table 5. PCI Data Bus vs DMA I/O Port Size**

| PCI DMA I/O Port Size | PCI Data Bus Connection |
|---|---|
| byte | AD[7:0] |
| word | AD[15:0] |

**Table 6. DMA I/O Cycle Width vs BE[3:0]#**

| BE[3:0]# | Description |
|---|---|
| 1110b | 8-bit DMA I/O Cycle |
| 1100b | 16-bit DMA I/O Cycle |

**NOTE:** For verify cycles, BE[3:0]# are “don’t cares”.

Every DMA device (including secondary bus arbiters) must recognize a valid signal on its GNT# combined with the DMA I/O address as its command authorization to initiate a DMA access cycle. MPIIX asserts the DMA I/O device's GNT# signal until the data phase of the I/O portion of the DMA transfer. Note that the AC timings for the I/O portion of a PCI DMA cycle are identical to the timings outlined in the PCI revision 2.0 specification for PCI I/O cycles.

#### 4.4.4.3. Normal DMA Cycle

An example of an entire PCI DMA cycle is illustrated in Figure 5 for a read of a 16-bit PCI DMA port. The Mobile PC/PCI DMA device initiates the cycle by asserting REQx#. When the MPIIX receives the PHLDA# signal, the channel number is passed to the expansion agent serially by the GNT# signal. This is followed by the actual DMA cycle that starts by performing two 16-bit I/O reads to the PCI DMA port, followed by a 32-bit memory write to the memory controller. The I/O reads occur with the PCI I/O address of 00h, while the memory write occurs to the selected memory. Data for the PCI DMA I/O cycles is always transferred on the address/data lines AD[15:0] for this example (16-bit cycle).

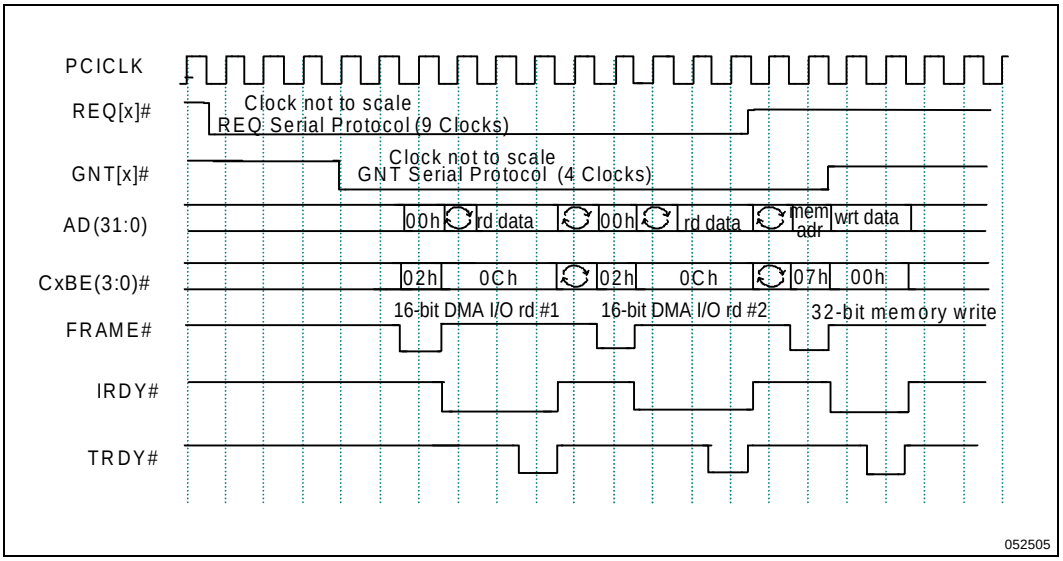



**Figure 5. DMA Write (16-Bit PCI I/O to PCI Memory)**

#### 4.4.4.4. Normal DMA Cycle with Terminal Count

The terminal count protocol for ending demand mode transfers (larger than a single transfer) is shown in Figure 6 for a write transfer from a 16-bit PCI I/O to PCI memory. When a DMA device initiates a multi-byte transfer, the DMA Controller passes a Terminal Count (TC) indication that the device is making its last transfer. The TC indication, that all requesting devices must recognize, is an I/O cycle (read or write, depending on whether the I/O portion was a load or store) by the DMA controller to address 04h. When the DMA I/O device or expansion agent detects a TC, it negates its REQ#. The MPIIX recognizes the removal of the request for service and removes its GNTx#, which cascades back through the PCI DMA agent to the initiating device.

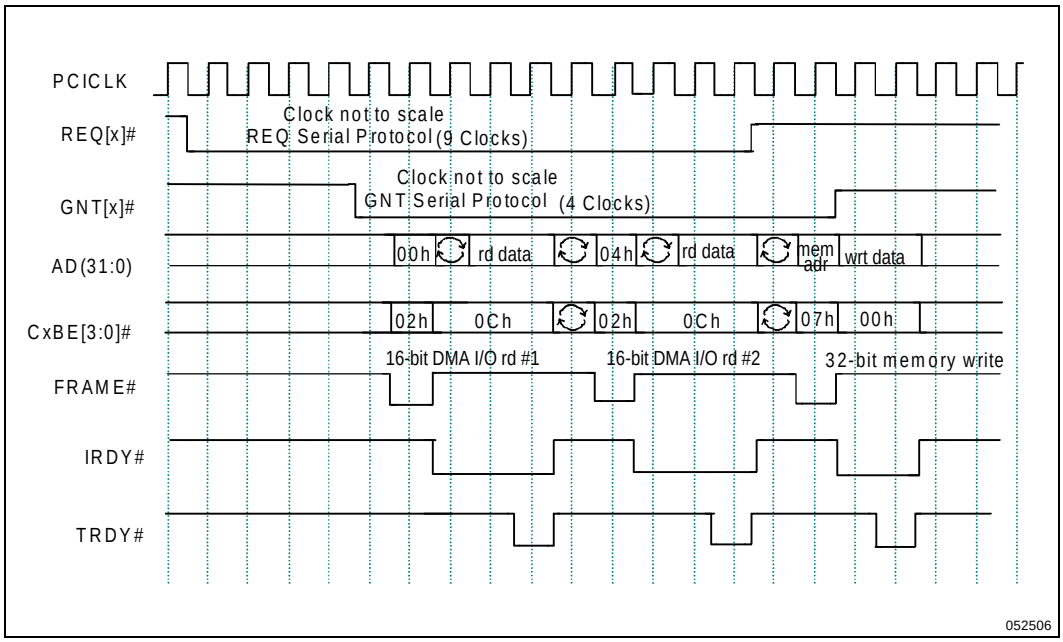



**Figure 6. Terminal Count of DMA Write (16-Bit PCI I/O to PCI Memory)**

#### 4.4.4.5. Verify DMA Cycle

The verify DMA cycle is similar to the normal DMA cycle, except no memory portion of the cycle takes place, and the I/O portion has the attributes of an I/O read cycle with address of 0C0h (Figure 7). Data is not transferred during a verify DMA cycle.

Note that some ISA DMA devices require that the DACK# toggle for each verify cycle. The PCI protocol, however, does not toggle GNT# for each transfer, but can perform multiple verify cycles while GNT# remains active. Bridges to ISA type peripherals should decode the verify cycle such that the GNT# to DACK# translation causes the DACK# to toggle for each transfer. This is only necessary on verify cycle (not normal DMA cycles).

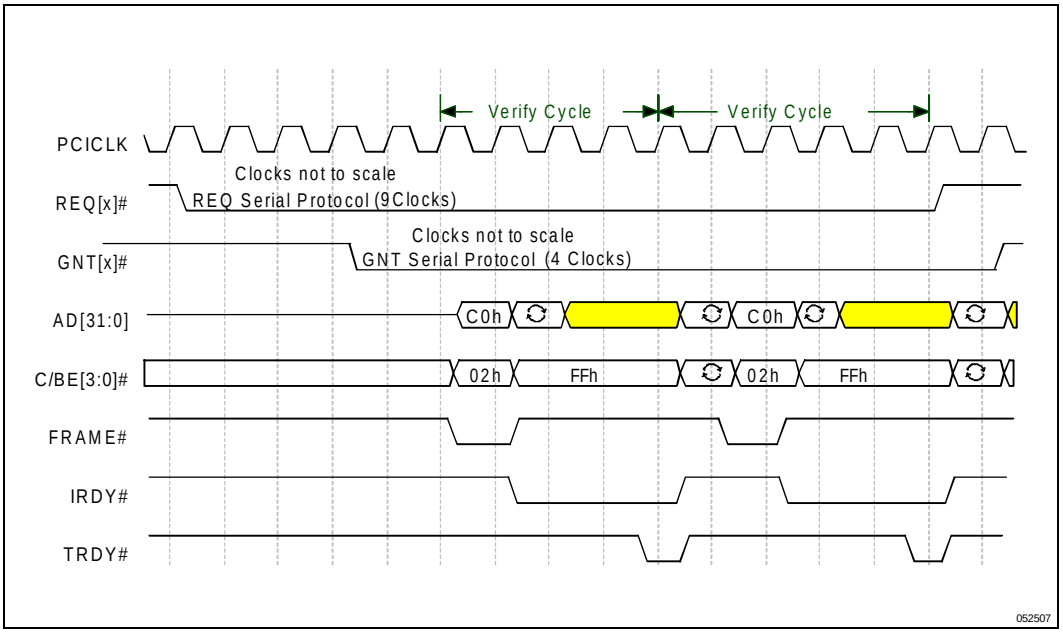



**Figure 7. Verify DMA Cycle**

#### 4.4.4.6. Verify DMA Cycle with Terminal Count

The terminal count during a verify cycle (Figure 8) is similar to the terminal count in a normal DMA cycle, except for the following: only the I/O portion occurs, the PCI command is always an I/O read cycle, no data is transferred on the PCI bus, and the I/O address is 0C4h for the terminal count cycle.

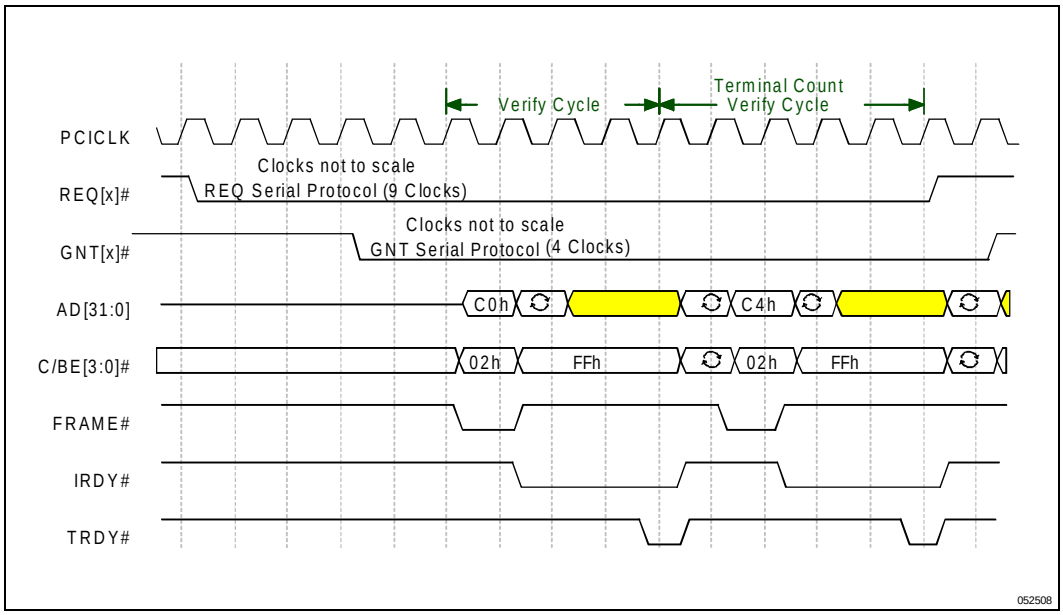



**Figure 8. Verify DMA Cycle With TC**

## 4.5. IDE Interface

The MPIIX integrates a high performance interface from PCI to IDE that is capable of accelerated PIO data transfers. The MPIIX provides an interface for one IDE connector that can be configured as either the primary or secondary IDE connector (Figure 9).

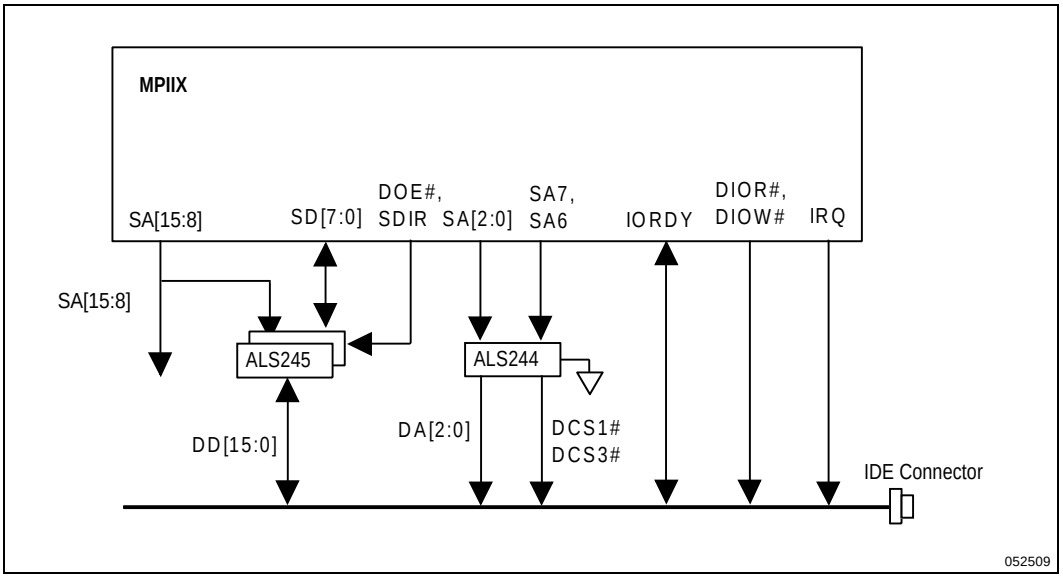



**Figure 9. MPIIX IDE Interface**

The IDE data transfer command strobes, DMA request and grant signals, and IORDY signal interface directly to the MPIIX. The IDE data lines (DD[15:0]) are buffered versions of the SD[7:0] system data lines and the multiplexed SA[15:8] system address lines. The IDE data buffer uses the same direction control signal (SDIR) as the PCS# interface and an output enable (DOE#). (DOE# shares a pin with SMOUT5 and must be configured as the DOE# pin.) The IDE address and chip select output signals are multiplexed on the SA[7,6,2:0] lines.

### 4.5.1. ATA REGISTER BLOCK DECODE

The IDE ATA I/O port is decoded by the MPIIX when the decode is enabled for either the primary connector or the secondary connector in the IDE Timing Register (IDETIM). The actual ATA registers are implemented in the drive itself. An access to the IDE registers results in the assertion of the appropriate chip select for the register. The transaction is then run using compatible timing and using the IDE command strobes (DIOR#, DIOW#).

There are two I/O ranges; the Command block (8-byte range) that corresponds to the DCS1# chip select and the Control block (4-byte range) that corresponds to the DCS3# chip select. The upper 15 bits of the I/O address (SA[17:3]) are decoded as all zeros.

Primary Command Block Offset: 01F0h
Primary Control Block Offset: 03F4h

Secondary Command Block Offset: 0170h
Secondary Control Block Offset: 0374h

Table 7 and Table 8specify the registers as they affect the MPIIX hardware definition.

**Table 7. IDE Legacy I/O port definition: Command Block (DCS1# chip select)**

| I/O Offset | Register Function (Read / Write) | Register Access |
|---|---|---|
| 00h | Data | R/W |
| 01h | Error/Features | R/W |
| 02h | Sector Count | R/W |
| 03h | Sector Number | R/W |
| 04h | Cylinder Low | R/W |
| 05h | Cylinder High | R/W |
| 06h | Drive/Head | R/W |
| 07h | Status/Command | R/W |

The Data Register is accessed as a 16-bit register for PIO transfers (except for ECC bytes). All other registers are accessed as 8-bit quantities.

**Table 8. IDE Legacy I/O port definition: Control Block (DCS3# chip select)**

| I/O Offset | Register Function (Read / Write) | Register Access |
|---|---|---|
| 00h | Reserved. Not Claimed by MPIIX unless FDC is enabled. | reserved |
| 01h | Reserved. Not Claimed by MPIIX unless FDC is enabled. | reserved |
| 02h | Alt Status / Device control | R/W |
| 03h | Reserved. Not Claimed by MPIIX unless FDC is enabled. | R/W |

The MPIIX claims all accesses to the Command Block Range and claims only the Control Block byte 3x4h offset 02h (3x6h) for the selected connector (primary or secondary). Note that the MPIIX only claims cycles to 3x4h offsets 00h, 01h, and 03h (3x4h, 3x5h, 3x7h) if the corresponding Floppy interface is enabled in the FDC Enable Register.

#### 4.5.2. ENHANCED TIMING MODES

The MPIIX includes fast timing modes that target local bus implementations. These timing modes are faster than those possible with ISA based implementations and are controlled with the granularity of the PCI clock. The fast timing modes may be enabled only for the IDE data ports. All other transactions to the IDE registers are run in single transaction mode with compatible timings.

Up to 2 IDE devices may be attached to the IDE connector (drive 0 and drive 1). Only one fast timing mode may be specified, by programming the IDETIM[ISP] and IDETIM[RCT]. This mode may be applied to drive 0, drive 1,

or both, by setting the IDETIM[TIME0] and or IDETIM[TIME1] bits. Transactions targeting the other drive will use compatible timing. The MPIIX snoops bit 4 of byte 6 of the ATA command block for the IDE connector. By keeping a copy of the current drive bits, the correct transaction timing can be determined.

#### 4.5.2.1. IORDY masking

The IORDY signal can be forced asserted on a drive by drive basis by setting the IDETIM[IE0] and IDETIM[IE1] register bits.

#### 4.5.2.2. PIO 32 bit IDE data port mode

If the 32-bit IDE data port mode is enabled, 32-bit accesses to the IDE data port address (default 01F0h primary etc.) result in two back to back 16-bit transactions to IDE.

The 32-bit data port feature is enabled for ALL timings, not just enhanced timing. Note that for compatible timing (mode 0), a Shutdown latency and Startup latency is incurred between the two halves of the requested Dword. This guarantees that the mode 0 IDE device will see the chip selects deassert for at least two clocks in between the two 16-bit reads. The 32-bit mode might speed up CD ROM drives incrementally since the CD ROM drives cannot be used with prefetching.

## 4.6. Interval Timer

The MPIIX contains three counters that are equivalent to those found in the 82C54 programmable interval timer. Each counter output provides a key system function. Counter 0 is connected to interrupt controller IRQ0 and provides a system timer interrupt for a time-of-day, diskette time-out, or other system timing functions. Counter 1 output (typically used to generate refresh requests for the ISA bus ) is reflected in port 61h, bit 4. Counter 2 generates the tone for the speaker. The 14.31818 MHz counters normally use OSC as a clock source.

**Counter 0, System Timer**

This counter functions as the system timer by controlling the state of IRQ0 and is typically programmed for Mode 3 operation. The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value one counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ0 and decrements the count value by two each counter period. The counter negates IRQ0 when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ0 when the count value reaches 0, reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ0.

**Counter 1, Refresh Request**

This counter provides the refresh cycle toggle in port 61h bit 4 and is typically programmed for Mode 2 operation. The counter negates the refresh cycle toggle bit for one counter period (838 ns) during each count cycle. The initial count value is loaded one counter period after being written to the counter I/O address. The counter initially sets the refresh cycle toggle, and negates it for 1 counter period when the count value reaches 1. The counter then sets the refresh cycle toggle and continues counting from the initial count value.

**Counter 2, Speaker Tone**

This counter provides the speaker tone and is typically programmed for Mode 3 operation. The counter provides a speaker frequency equal to the counter clock frequency (1.193 MHz) divided by the initial count value. The speaker must be enabled by a write to port 061h (see NMI Status and Control ports).

## 4.7. Interrupts

The MPIIX provides an ISA compatible interrupt controller which incorporates the functionality of two 82C59 interrupt controllers. The two controllers are cascaded so that 13 external and three internal interrupts are possible. The master interrupt controller provides IRQ [7:0] and the slave interrupt controller provides IRQ [15:8] (Figure 10). The three internal interrupts are used for internal functions only and are not available to the user. IRQ2 is used to cascade the two controllers together. IRQ0 is used as a system timer interrupt and is tied to Interval Timer 1, Counter 0. IRQ13 is connected internally to FERR#. The remaining 13 interrupt lines (IRQ[15,12:9,8#,7:3,1] are available for external system interrupts. Edge or level sense selection is programmable on an individual channel-by-channel basis.

The Interrupt unit also supports interrupt steering. The MPIIX can be programmed to allow the two PCI active low interrupts (PIRQ[A,B]#) to be internally routed to one of 11 interrupts (IRQ[15,14,12:9,7:3]. In addition, an interrupt signal is dedicated to motherboard devices (MIRQ#) may be routed to any of the 11 interrupts.

The Interrupt Controller consists of two separate 82C59 cores. Interrupt Controller 1 (CNTRL-1) and Interrupt Controller 2 (CNTRL-2) are initialized separately and can be programmed to operate in different modes. The default settings are: 80x86 Mode, Edge Sensitive (IRQ[15:0]) Detection, Normal EOI, Non-Buffered Mode, Special Fully Nested Mode disabled, and Cascade Mode. CNTRL-1 is connected as the Master Interrupt Controller and CNTRL-2 is connected as the Slave Interrupt Controller.

Note that IRQ13 is generated internally (as part of the coprocessor error support) by the MPIIX. IRQ12/M is generated internally (as part of the mouse support) when bit 6 in the FDCE is set to a 1. When set to a 0, the standard IRQ12 function is provided and IRQ12 appears externally.

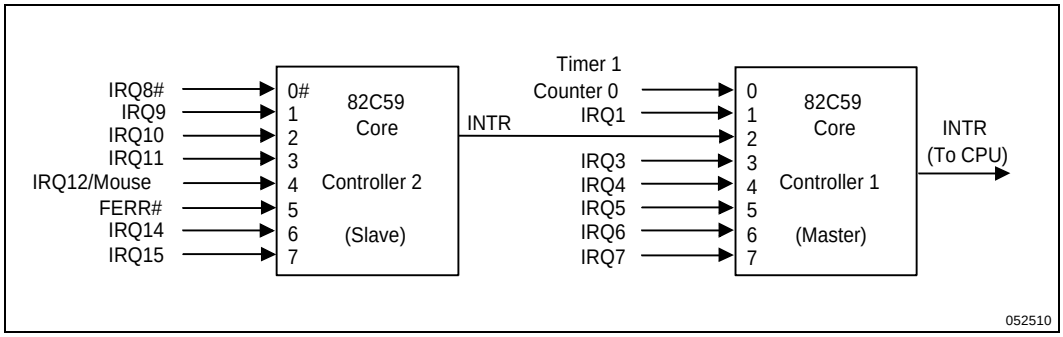



**Figure 10. Block Diagram Of The Interrupt Controller**

### 4.7.1. PROGRAMMING THE INTERRUPT CONTROLLER

The Interrupt Controller accepts two types of command words generated by the CPU or bus master:

**1. Initialization Command Words (ICWs)**: Before normal operation can begin, each Interrupt Controller in the system must be initialized. In the 82C59, this is a two- to four-byte sequence. However, for the MPIIX, each controller must be initialized with a four-byte sequence. This four-byte sequence is required to configure the interrupt controller correctly for the MPIIX implementation. This implementation is ISA-compatible. The four initialization command words are referred to by their acronyms: ICW1, ICW2, ICW3, and ICW4. The base address for each interrupt controller is a fixed location in the I/O memory space, at 0020h for CNTRL-1 and at 00A0h for CNTRL-2.

An I/O write to the CNTRL-1 or CNTRL-2 base address with data bit 4 equal to 1 is interpreted as ICW1. For MPIIX-based ISA systems, three I/O writes to "base address + 1" (021h for CNTRL-1 and 0A0h for CNTRL-2) must follow the ICW1. The first write to "base address + 1" (021h/0A0h) performs ICW2, the second write performs ICW3, and the third write performs ICW4.

**2. Operation Command Words (OCWs):** These are the command words that dynamically reprogram the interrupt controller to operate in various interrupt modes. Any interrupt lines can be masked by writing an OCW1. A 1 written in any bit of this command word masks incoming interrupt requests on the corresponding IRQx line. OCW2 is used to control the rotation of interrupt priorities when operating in the rotating priority mode and to control the End of Interrupt (EOI) function of the controller. OCW3 set up reads of the ISR and IRR, enable/disables the Special Mask Mode (SMM), and sets up the interrupt controller in polled interrupt mode. The OCWs can be written to the Interrupt Controller any time after initialization.

#### 4.7.1.1. Edge and Level Triggered Mode

In ISA systems this mode is programmed using bit 3 in ICW1. With MPIIX this bit is disabled and a new register for edge and level triggered mode selection, per interrupt input, is included. This is the Edge/Level control Registers ELCR1 and ELCR2. The default programming is equivalent to programming the LTIM bit (ICW1 bit 3) to a 0 (all interrupts selected for edge triggered mode). Note, that IRQ0, 1, 2, 8#, and 13 can not be programmed for level sensitive mode and can not be modified by software.

If an ELCR bit = 0, an interrupt request will be recognized by a low to high transition on the corresponding IRQx input. The IRQ input can remain high without generating another interrupt. If an ELCR bit = 1, an interrupt request will be recognized by a low level on the corresponding IRQ input and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued to prevent a second interrupt from occurring.

In both the edge and level triggered modes, the IRQ inputs must remain active until after the falling edge of the first INTA#. If the IRQ input goes inactive before this time, a default IRQ7 will occur when the CPU acknowledges the interrupt. This can be a useful safeguard for detecting interrupts caused by spurious noise glitches on the IRQ inputs. To implement this feature, the IRQ7 routine is used for "clean up" simply executing a return instruction, thus ignoring the interrupt. If IRQ7 is needed for other purposes, a default IRQ7 can still be detected by reading the ISR. A normal IRQ7 interrupt will set the corresponding ISR bit; a default IRQ7 will not set this bit. If a default IRQ7 routine occurs during a normal IRQ7 routine, however, the ISR will remain set. In this case, it is necessary to keep track of whether or not the IRQ7 routine was previously entered. If another IRQ7 occurs, it is a default.

### 4.7.2. INTERRUPT STEERING

The MPIIX can be programmed to allow two PCI programmable interrupts (PIRQ[A,B]) to be internally routed to one of 11 interrupts (IRQ15,14,12:9,7:3]. PCICLK is used to synchronize the PIRQx# inputs. The PIRQx# lines are run through an internal multiplexer that assigns, or routes, an individual PIRQx# line to any one of 11 IRQ inputs. The assignment is programmable through the PIRQx Route Control registers. One or more PIRQx# lines can be routed to the same IRQx input. If interrupt steering is not required, the Route Registers can be programmed to disable steering.

The PIRQx# lines are defined as active low, level sensitive to allow multiple interrupts on a PCI Board to share a single line across the connector. When a PIRQx# is routed to specified IRQ line, the software must change the IRQ's corresponding ELCR bit to level sensitive mode. Note, that this means that the selected IRQ can no longer be used by an another device, unless that device can respond as an active low level sensitive interrupt.

The MPIIX also supports a programmable interrupt (MIRQ). MIRQ is intended for use with motherboard devices and can be routed to the any of the same 11 interrupts that the PIRQx# lines can be routed to using the MIRQ Register. The routing accomplished in the same manner as for the PIRQx# inputs, except it is assumed that the interrupts are active high. If interrupt steering is not required, the MIRQ register can be programmed to disable routing.

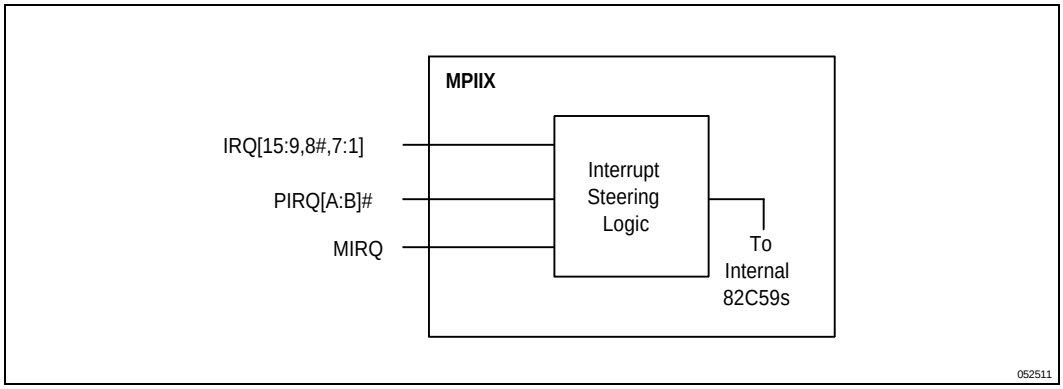



**Figure 11. Interrupt Steering Logic**

### 4.7.3. MOUSE FUNCTION

When the mouse interrupt function is enabled (via the FDC Enable Register), the mouse interrupt function is provided on the IRQ12/M input signal. In this mode, a mouse interrupt generates an interrupt through IRQ12 to the Host CPU. The MPIIX informs the CPU of this interrupt via a INTR. A read of 60h releases IRQ12. If the IRQ12 function is enabled (mouse function disabled), a read of address 60h has no effect on IRQ12/M. Reads and writes to this register flow through to the Extended I/O Bus. For additional information, see the IRQ12/M description in the Signal Description.

### 4.7.4. COPROCESSOR ERROR FUNCTION

This function provides coprocessor error support for the CPU. This function is enabled via the FDC Enable Register. FERR# is tied directly to the coprocessor error signal of the CPU. If FERR# is driven active to the MPIIX, an internal IRQ13 is generated and an the INTR output from the MPIIX is generated. When a write to I/O location F0h is detected, the MPIIX negates IRQ13 (internal to the MPIIX) and asserts IGNNE#. IGNNE# remains asserted until FERR# is negated. Note, that IGNNE# is not driven active unless FERR# is active.

### 4.7.5. NMI SUPPORT

See Register Description.

## 4.8. Power Management Support

The Intel 430MX PCIset power management provides flexible mechanisms to help the operating system and system software manage the use of system resources for the lowest possible power consumption while providing the best performance to the user.

MPIIX uses several mechanisms to help the power management software initiate and manage the transitions between the power managed states. These include System-wide and Local Peripheral Event Monitors to identify idle and wake-up conditions, Intel's System Management Interrupt (SMI#) support, Advanced Power Management (APM) interface, and Pentium® Processor STPCLK# Clock Control, and Suspend/Resume Hardware. MPIIX provides the following 3 basic areas of power management:

- **CPU Standby.** When the operating system, application program, or system software is not doing useful work, the CPU complex (CPU, DRAM, L2 Cache) does not need to be executing cycles and can therfore be placed in a CPU Standby mode.
- **Local Standby.** When a local peripheral, such as IDE hard disk or COM port, has not been used for a specified amount of time, that peripheral can be placed in a Local Standby mode.
- **System Suspend.** When the entire system has been idle for a specified amount of time or a critical system event occurs, such as a battery low condition, the system will be put in it's lowest power state.

**Power Management Feature Summary**

- **Power Management Initiation (SMI Generation)**
  - — Global Standby Timer to Identify the System Idle Condition
  - — Software SMI#, External H/W SMI# (EXTSMI#)
  - — APM Software Initiated SMI#
  - — SMI# Generation Delay Timers
  - — SMI# Generation Status
  - — APM CallBack Feature
- **Power Management for CPU Complex: CPU, DRAM, L2 Cache ( CPU Standby)**
  - — Flexible STPCLK# Mechanism for CPU Clock Control
    - With CPU CLK input running
    - With CPU CLK input Stopped
  - — Hardware Event Control for Wake-Up from STPCLK# (Stop Break Events)
  - — STPCLK# Duty Cycle Control for Low Frequency Emulation
  - — PCI Clock Control (CLKRUN#)
  - — APM Initiated Stop Clock Control
- **Power Management of Local Peripherals (Local Standby)**
  - — Timers to Identify the Peripheral Idle Conditions
  - — Traps for Access to Powered-Down Peripherals
  - — SMOUT[5:0] Programmable Outputs for Power Plane Control
  - — Leakage Control for Powered-Down Peripherals
  - — SMI# Sequencing to CPU for I/O Cycle Restart
- **Power Management for System Suspend**
  - — Suspend/Resume Button Input (SRBTN#)
  - — BATLOW# Indication Pin
  - — Shadow Registers for Standard AT Write-Only Registers
  - — DRAM Self-Refresh During Suspend
  - — "Resume Well" To Monitor Wake-up Events During 0V Suspend
  - — Power-Down Leakage Control
  - — Resume Power and Reset Sequencing

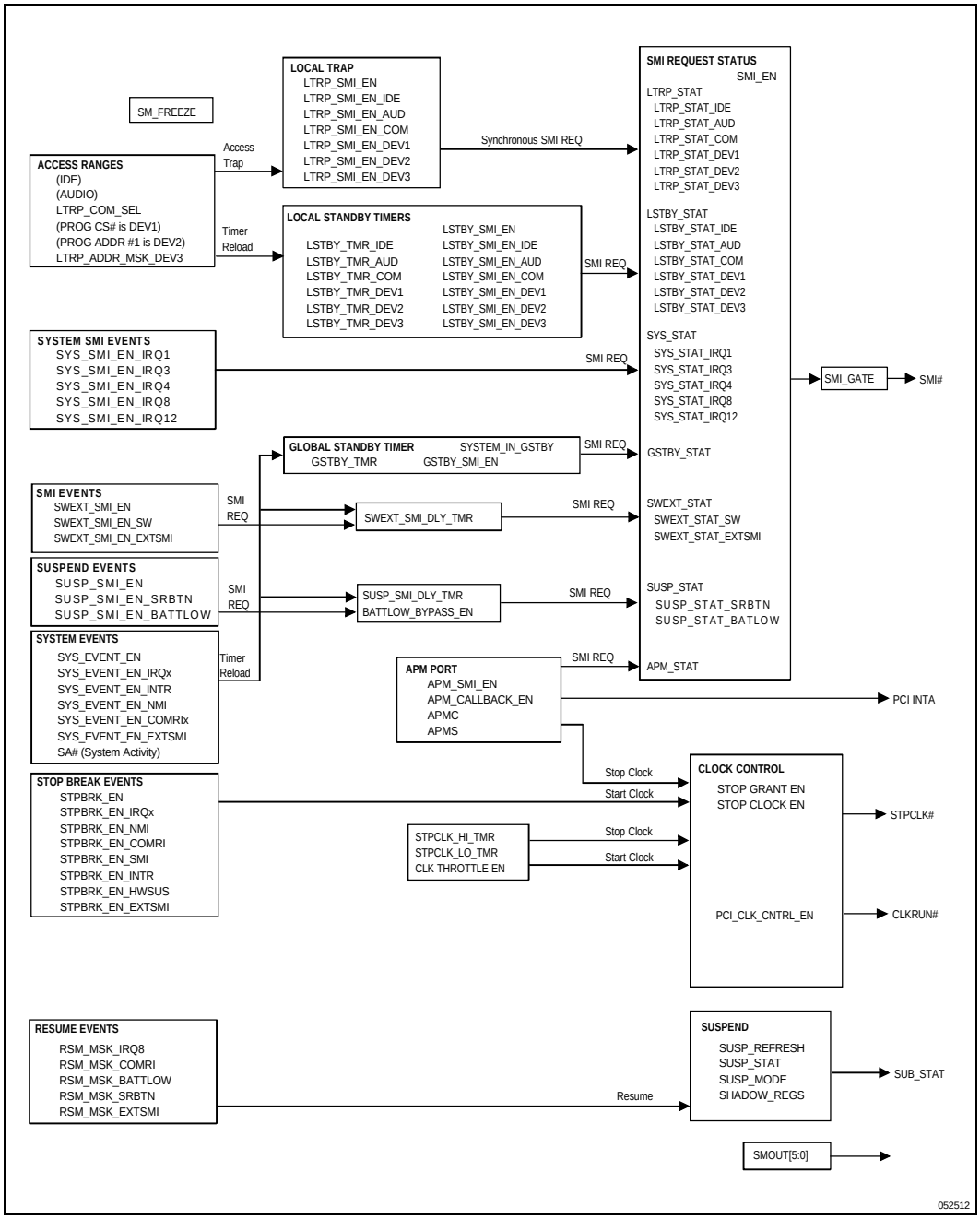



**Figure 12. Power Management Overview**

#### 4.8.1. SMI GENERATION

SMI# Generation logic controls the enabling of the SMI# sources, the timing and assertion of the SMI# signal to the CPU, and the recording of what event triggered SMI#.

- **SMI# Sources:** SMI# is generated periodically for system polling or specifically in response to a change in the system's power management requirements. These changes can be signaled by the Local Standby hardware, Global Standby Idle Timer expiration, specific software SMI# requests such as writes to the Advanced Power Management port (APMC) or the software SMI request, the external SMI# signal (EXTSMI#), specific hardware SMI# requests such as the suspend/resume button (SRBTN#), or the Battery Low signal (BATTLOW#).

  Note: Local Standby is discussed in more detail in a separate section. SRBTN# and BATLOW# are discussed in the Suspend section. All other SMI sources will be discussed in this section.
- **SMI# Enables:** All of the SMI# sources have individual enables. Some sources have group enables that minimize the time necessary to block SMI# during certain power management procedures. The global SMI# Enable bit (SMI_EN) in the SYSMGNTC Register prevents any SMI# source from setting it's corresponding SMI# status bit.
- **SMI# Request Status:** The SMI# sources have a corresponding request status bit structure which allows the SMI handler to quickly vector to the appropriate subroutine.
- **SMI# Signal Generation:** SMI# will remain asserted as long as the SMI_GATE is set to 1 (SYSMGNTC Register). SMI# is negated when the SMI_GATE bit is set to 0. SMI# is asserted again when SMI_GATE is set to 1, if any SMIs are pending. Access to a powered-down peripheral requires a special sequence ("synchronous SMI#") so that the CPU can restart the cycle to that device after it is returned to full power.

##### 4.8.1.1. SMI Enables

MPIIX has one global SMI# enable that, when disabled, blocks all SMI# sources and when enabled allows individual enable control. Each SMI# source has its own individual enable bit while some groups of SMI# sources have group enables. When a 1 is written to the enable, the source or group is enabled. When a 0 is written to the enable, the source or group is disabled.

If the SMI# source is associated with a timer, setting the SMI# enable bit will generally initiate the timer count-down. If the SMI# source is associated with an access trap, setting the enable bit will enable the access trap. The specific mechanisms are described in more detail in the specific SMI# source section.

##### 4.8.1.2. SMI Request Status

The request status bits correspond directly to the SMI event that needs servicing. When an SMI event occurs, the hardware automatically sets the corresponding request status bit(s) to a 1 for the event that caused the SMI. The status bits are cleared by writing 0 to them. Only the hardware can set status bits to 1. In the event that the hardware is trying to set the bit to a 1 at the same time that it is being cleared, the hardware set to 1 will dominate. The SMI handler will query the status bits to see what caused the SMI and then branch to the appropriate routine. As the individual routines complete they reset the appropriate status bit by writing a 0 to the corresponding bit.

The first column of Table 9 lists the bits that are used to enable the SMI# sources. The specific function of each of the enables is described in the corresponding section for that SMI# source. The second column lists the status bit for each of the SMI# sources.

**Table 9. SMI# Enables and Status**

| Enable Bit | Status Bit | SMI# Source |
|---|---|---|
| LSTBY_SMI_EN | LSTBY_STAT | Any Local Standby Idle timer. |
| LSTBY_SMI_EN_IDE | LSTBY_STAT_IDE | IDE Idle Timer Expired. |
| LSTBY_SMI_EN_AUD | LSTBY_STAT_AUD | Audio Idle Timer Expired. |
| LSTBY_SMI_EN_COM | LSTBY_STAT_COM | COM Port Idle Timer Expired. |
| LSTBY_SMI_EN_DEV1 | LSTBY_STAT_DEV1 | Programmable Device 1 (PCS#) Idle Timer Expired. |
| LSTBY_SMI_EN_DEV2 | LSTBY_STAT_DEV2 | Programmable Device 2 (PAC1) Idle Timer Expired. |
| LSTBY_SMI_EN_DEV3 | LSTBY_STAT_DEV3 | Programmable Device 3 Idle Timer Expired. |
| | | |
| LTRP_SMI_EN | LTRP_STAT | Any Local Standby Access Trap. |
| LTRP_SMI_EN_IDE | LTRP_STAT_IDE | Access to IDE device. |
| LTRP_SMI_EN_AUD | LTRP_STAT_AUD | Access to Audio device. |
| LTRP_SMI_EN_COM | LTRP_STAT_COM | Access to COM port. |
| LTRP_SMI_EN_DEV1 | LTRP_STAT_DEV1 | Access to programmable device 1 (PCS#). |
| LTRP_SMI_EN_DEV2 | LTRP_STAT_DEV2 | Access to programmable device 2 (PAC1). |
| LTRP_SMI_EN_DEV3 | LTRP_STAT_DEV3 | Access to programmable device 3. |
| | | |
| SYS_SMI_EN | SYS_STAT | Any System SMI Event. |
| SYS_SMI_EN_IRQ1 | SYS_STAT_IRQ1 | Timer Tick. |
| SYS_SMI_EN_IRQ3 | SYS_STAT_IRQ3 | COM. |
| SYS_SMI_EN_IRQ4 | SYS_STAT_IRQ4 | COM. |
| SYS_SMI_EN_IRQ8 | SYS_STAT_IRQ8 | RTC Alarm. |
| SYS_SMI_EN_IRQ12 | SYS_STAT_IRQ12 | Mouse. |
| | | |
| SWEXT_SMI_EN | SWEXT_STAT | Software SMI mechnism or EXTSMI# pin. |
| SWEXT_SMI_EN_SW | SWEXT_STAT_SW | Software SMI mechanism. |
| SWEXT_SMI_EN_EXTSMI | SWEXT_STAT_EXTSMI | External SMI pin (EXTSMI#). |
| | | |
| SUSP_SMI_EN | SUSP_STAT | Suspend Resume Button or Battery Low Pin. |
| SUSP_SMI_EN_SRBTN | SUSP_STAT_SRBTN | Suspend Resume Button (pin). |
| SUSP_SMI_EN_BATLOW | SUSP_STAT_BATLOW | Battery Low Pin. |
| | | |
| APM_SMI_EN | APM_STAT | Write to APM Control Port (APMC). |
| | | |
| GSTBY_SMI_EN | GSTBY_STAT | Global Standby Timer Expired. |

#### 4.8.1.3. SMI# Signal Generation

When the SMI_GATE bit is set to a 1 (SYSMGNTC Register), SMI# is asserted when the hardware, software, or external SMI is asserted. Clearing the SMI_GATE bit causes SMI# to be negated. SMI# is re-asserted when the SMI_GATE bit is set to a 1, if there is a pending SMI. If simultaneous active set and reset conditions occur the SMI_GATE reset function is dominant.

The Local Trap source (see Local Standby section) requires that the SMI# signal is asserted at least 3 CPU CLKs prior to asserting the ready signal (RDY#, BRDY#) that completes the I/O cycle that generated the trap. Since the CPU's ready signal is asserted by the MTSC, and the SMI# signal is asserted by MPIIX, this "synchronous SMI#" timing is guaranteed by the timing of the MPIIX PCI ready generation and the propagation of the PCI ready through the MTSC.

#### 4.8.1.4. SMI SOURCES

**Global Standby Timer**

The Global Standby Timer is used to identify when the system is idle. Power management software loads this timer with a 8-bit count, then starts the timer by writing a 1 to the GSTBY_SMI_EN bit. The counter is decremented by an 8 second clock to provide a maximum timeout of 34 minutes. When the count expires, an SMI# request is generated to the SMI logic and the GSTBY_STAT bit is set.

This Global Standby Timer is reloaded with the initial count by the following events:

- Setting the GSTBY_SMI_EN bit (GSMIE Register).
- Globall Standby Timer expires.
- System Events listed in Table 10. These system events can be individually enabled to reload or not reset the global standby timer. The global enable (SYS_EVENT_EN) can be used to block all events from reseting the global standby timer.

The Global Standby Timer countdown is stopped when SM_FREEZE=1 in the SYSMGNTC Register.

If the power management software determines that the system can be placed in a "global standby" state, MPIIX provides a register bit that can be used to indicate, to a future SMI# handler call, that the system is in a global standby state. (There is no specific global standby state defined by the Intel 430MX PCIset hardware. This is defined by the system designer.)

**Table 10. System Events that Reload Global Timer and SMI# Delay Timers**

| Enable Bit | System Event |
|---|---|
| SYS_EVENT_EN | Global enable for all System Events. Setting to 0 prevents all enabled System Events from reseting the Global Standby Timer. |
| SYS_EVENT_EN_IRQx | IRQ1, IRQ3–12,14,15. |
| SYS_EVENT_EN_INTR | Enable INTR to reload timers. |
| SYS_EVENT_EN_COMRI | Enable the COM Ring Indicate to reload timers. |
| SYS_EVENT_EN_NMI | Enable the NMI signal to reload timers. |
| SYS_EVENT_EN_HWSUS | Enable the BATLOW# and SRBTN# signals to reload the timers. |
| SYS_EVENT_EN_SMI | Enable the SMI# signal to reload the timers. |
| SYS_EVENT_EN_EXTSMI | Enable the EXTSMI# to reload the timers. |
| SA# (SYSTEM ACTIVITY) | The SA# (System Activity) signal on MPIIX will always reload the timers. |

**External Hardware SMI (EXTSMI#) and Software SMI#**

Both of these SMI# sources can be disabled (by the SWEXT_SMI_EN bit to 0 in the GSMIE Register). When SWEXT_SMI_EN=1, the individual SMI enables determine if the source is enabled.

This hardware EXTSMI# signal and the software SMI# might require a delay to allow the system to settle prior to asserting the SMI# signal to the CPU. Both sources share an SMI# delay timer (Software/EXTSMI# SMI Delay Timer) that allows the system to finish its current bus master or docking station bridge actvity before the SMI# is generated.

**EXTSMI#**

The external SMI input signal (EXTSMI#) permits hardware to generate an SMI# to the CPU. Power management software can enable this SMI# source (by setting the SWEXT_SMI_EN_EXTSMI bit to a 1 in the SYSSMIE Register). When an EXTSMI# input is asserted, the SMI# request is passed to the Software/EXTSMI# SMI Delay Timer and the timer begins to count down. The EXTSMI# signal is level triggered and should be asserted for a minimum of 32 usec. The EXTSMI# signal is typically asserted until it is cleared by the SMI# interrupt handler routine.

**SW SMI#**

Software can generate an SMI# to invoke the SMI# handler by setting the SWEXT_SMI_EN_SW bit to a 1 in the GSMIE Register. This SMI# request is passed to the Software/EXTSMI# SMI Delay Timer and the timer begins to count down.

**SWEXT SMI# Delay Timer**

The Software/EXTSMI# SMI Delay Timer is loaded by software with an 8-bit count for a minimum delay of 1 ms to a maximum delay of 255 ms. This delay timer begins counting down when the software SMI# enable bit is set (SWEXT_SMI_EN) or when the EXTSMI# input is asserted by the system (and the SWEXT_SMI_EN_EXTSMI bit was previously enabled).

The Software/EXTSMI# SMI Delay Timer is reloaded by the system events that are enabled. (These are the same system events that reload the Global Standby Timer). When the timer expires it will generate an SMI#, if enabled. When the SMI# request is generated, the MPIIX sets the SMI# group request status bit (SWEXT_STAT in the GSMIS Register). The MPIIX also sets the individual status bit (SYSSMIS Register) for the source that caused the SMI# or sets the status bit for both sources, if both become active while the SMI# delay timer is counting down.

**APM Ports**

The APM ports consist of two 8-bit ports—A status port (APMS Register) and a control port (APMC Register). These read/write registers are used to transfer information between the OS and the SMI handler. The APMS Register resides at system I/O address 0B3h and the APMC Register at 0B2h. I/O writes to these registers store data in them. I/O reads return data from these registers.

The MPIIX positively decodes PCI accesses to B2h and B3h. Read data is returned from the internal MPIIX register. Extended I/O masters can not access B2h and B3h.

I/O writes to the APMC Register generates an SMI if the APM_SMI_EN bit is set to 1 (see SYSSMIE Register). MPIIX also supports the APM CallBack feature where the SMI# is routed to one of the PCI interrupts (PIRQA), instead of the SMI# interrupt. This PIRQA interrupt can then be routed to any of the internal IRQs. The APM CallBack feature is enabled when the APM_CALLBACK_EN bit is set to 1 (MISCSMIE Register) and the APM_SMI_EN bit is set to 1 (SYSSMIE Register) and the AMP_SMI_EN bit is set to 1. When either the APM SMI# or the APM callBack to PIRQA# occurs, the APM_STAT bit in the GSMIS Register is set to 1. The interrupt handler should clear the APM_STAT bit before returning.

I/O reads to the APMC Register additionally generate STPCLK#, if enabled in the CLKC Register (See Software Control of STPCLK# section).

### 4.8.2. CPU POWER MANAGEMENT (CPU, DRAM, L2 CACHE, DATAPATH)

**OS Idle Condition.** When the OS enters an idle condition, it is waiting for some user input or hardware event to continue useful work. Since the OS does not require the CPU to execute software, the CPU complex including the DRAM, L2 cache and datapath, does not need to be running.

The system can indicate an idle condition through hardware (idle timers) or software (O/S, APM) and generate an SMI# to invoke the power management BIOS. Power management BIOS determines what level of clock control is required and instructs MPIIX to execute that mode.

**Lower Frequency Emulation (Clock Throttling).** When the power management software determines that the system does not require full frequency operation but cannot stop the clock for an indefinite amount of time, the power management can transition between ON and CPU Standby at a predetermined duty cycle. For example, at a 50% duty cycle the effective frequency is 1/2 of the CPU CLK frequency.

The Intel 430MX PCIset has 3 independent Clock Control mechanisms: Stop Clock (either CPU Stop Grant State or CPU Stop Clock State), Clock Throttling, and Auto Clock Throttle.

**PCI Clock Control.** MPIIX uses the CLKRUN# protocol to provide the capability to stop the PCI clock when there is no PCI bus activity.

MPIIX generates and controls both the CPU Host Clock (HCLKO) and the PCI clock (PCICLKO) to the system as illustrated in Figure 13.

#### 4.8.2.1. Stop Clock

The processor can be put in a low power state by externally asserting the STPCLK#. STPCLK# is an interrupt to the CPU; however, there is not an interrupt acknowledge cycle generated. Once the STPCLK# interrupt is executed, the processor enters the Stop Grant state. In the Stop Grant state the internal clocks are disabled and instruction execution is stopped. To exit the Stop Grant state the STPCLK# signal is negated. The CPU power consumption can be further reduced by stopping the host CLK input to the CPU while in the Stop Grant state, causing the CPU to enter the Stop Clock state. The Stop Clock state requires a warm-up delay when re-starting the CLK input to the CPU.

MPIIX waits for the PCICLK to stop before stopping the Host Clock (HCLKO from MPIIX). If the PCI Clock Control is disabled in the Clock Control Register (PCI_CLK_CTRL_EN bit), the Host Clock cannot be stopped. In this case the system can enter the Stop Grant state and will not enter the Stop Clock State.

- **Entering CPU Stop Grant State**
  - MPIIX asserts STPCLK#.
  - CPU accepts STPCLK# interrupt, flushes buffers, sends the Stop Grant bus cycle.
  - MTSC host-to-PCI bridge forwards Stop Grant bus cycle to PCI bus and does PCI master abort.
  - MTSC completes the Stop Grant bus cycle by returning a RDY# (BRDY#) to the CPU
  - CPU gates the internal clocks to the CPU core and enters the Stop Grant state.

- **Leaving the CPU Stop Grant State**
  - MPIIX negates the STPCLK# input.
  - CPU returns to the On state and resumes execution.

- **Entering CPU Stop Clock State**
  - MPIIX asserts STPCLK#.
  - CPU accepts STPCLK# interrupt, flushes buffers, sends the STOP GRANT bus cycle.
  - MTSC host-to-PCI bridge forwards Stop Grant bus cycle to PCI bus and does PCI Master Abort.
  - MTSC completes the Stop Grant bus cycle by returning a RDY# (BRDY#) to the CPU
  - CPU gates the internal clocks to the CPU core and enters the Stop Grant state.
  - MPIIX stops the CLK input to the CPU only AFTER the RDY# has been returned to the CPU.

- **Leaving the CPU Stop Clock State**
  - MPIIX starts the CLK input to the CPU and waits for the CLK Start Latency timer to expire (about 1 ms).
  - MPIIX negates STPCLK#.
  - CPU returns to the On state and resumes normal execution.

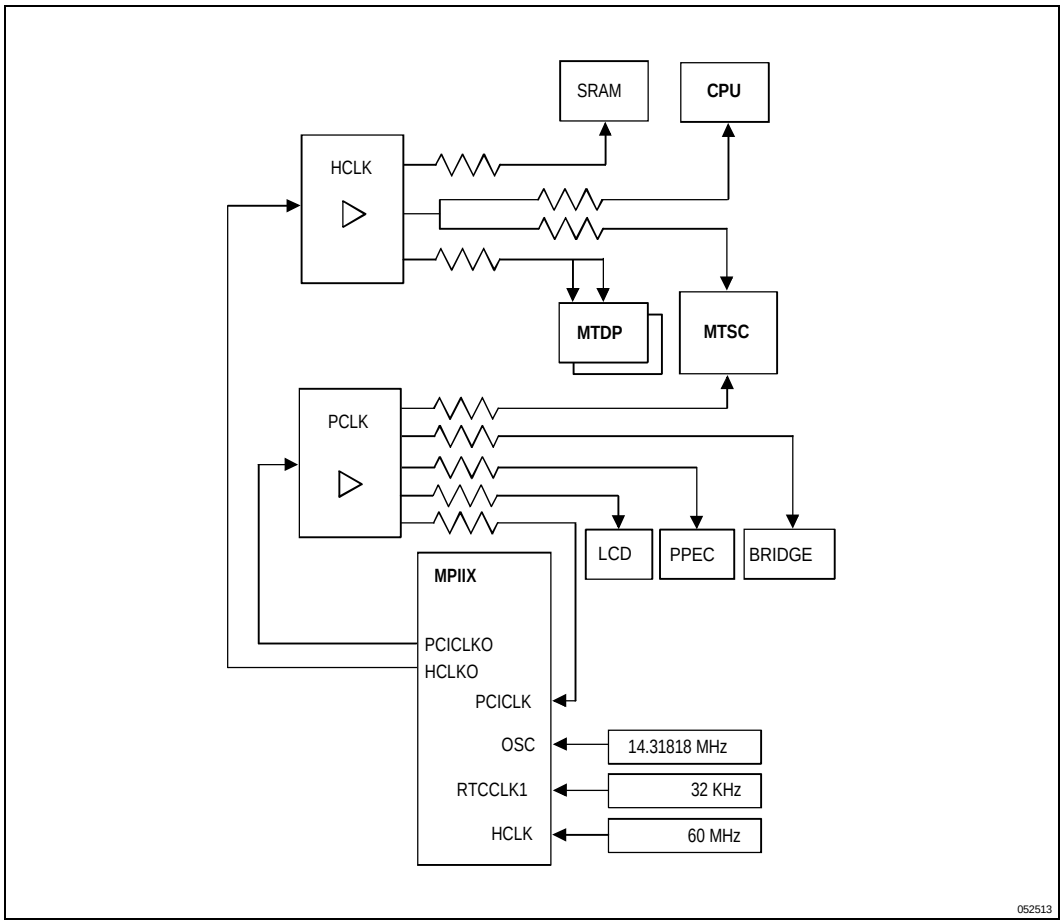



**Figure 13. Intel 430MX PCIset Clock Distribution**

#### 4.8.2.2. Software Control of STPCLK#

The STPCLK# process is initiated by reading from the APMC Register. The MPIIX places the CPU into a Stop Grant state (external CLK still applied to CPU) or places the CPU in a lower power Stop Clock state (external CLK stopped to the CPU) depending on the STPCLK_MODE bit in the CLKC Register.

The system is brought out of the STPCLK# state when any enabled Stop Break Event occurs. Each of the Stop Break Events can be individually enabled by writing a 1 to its enable bit in the Stop Break Enable Register and all Stop Break Events can be disabled as a group by the STPBRK_EN bit to 0 in the STPBRKE2 Register. Stop Break Events include IRQ[15:3,1], NMI, COMRI#, SMI#, INTR, SRBTN#, BATLOW#, and EXTSMI#. Stop break events are not recognized until after the stop grant bus cycle has completed.

#### 4.8.2.3. Emulating Clock Division (Clock Throttling)

When emulating clock division, the processor is running at full frequency for a pre-defined time period and then is stopped for a pre-defined time period. The Run/Stop time interval ratio emulates the clock division effect from a power/performance point of view. The clock division emulation is more effective than physically dividing the processor frequency since upon a system Break Event the processor clock returns to full frequency. Also there is no recovery time latency to start the clock. The clock division emulation is described in Figure 4.12. It works in conjunction with the software driven Stop Clock feature.

**Functional Description**

Two programmable time intervals are provided to throttle the clock. The STPCLKHT Register defines the time that the STPCLK# signal is negated, and the STPCLKLT Register defines the time that STPCLK# is asserted. A single timer is loaded to count both intervals. To enable the CPU clock throttling the CLK_THROTTLE_EN bit must be set to 1.

When enabled, the STPCLK# Timer is automatically loaded as follows:

- When STPCLK# is negated, the STPCLK# Timer is loaded from the STPCLKHT Register and the timer starts counting down. When the timer reaches 00h, STPCLK# is asserted.
- When STPCLK# is asserted, the STPCLK# Timer is loaded from the STPCLKLT Register. When the timer reaches 00h, STPCLK# is negated.
- While STPCLK# is negated, the STPCLK# Timer is loaded from the STPCLKHT Register when a Break Event occurs. This prevents the CPU from entering a lower performance state if the system is active. Break Events should be disabled if a constant lower frequency emulation is desired. Stop break events are not recognized until after the stop grant bus cycle has completed.

The 8-bit STPCLK# Timer is clocked by a 32 usec clock. The STPCLKHT and STPCLKLT Registers allow programming of the timing intervals for the assert/negate states of STPCLK#. The actual assertion and negation time for STPCLK# is 2 counts greater that the programmed count. This allows a programmable interval from approximately 96 μsec to 8 msec for both STPCLK# asserted and STPCLK# negated periods. The actual time depends on the frequency of RTCCLK.

#### 4.8.2.4. STPCLK Control State Machine

Power management software can implement 3 possible state machines as illustrated in the Figure 4.12.

1. The MPIIX Clock Throttling hardware can automatically assert and negate STPCLK# based on the desired duty cycle set up in the STPCLKLT and the STPCLKHT Registers when the CLK_THROTTLE_EN bit is set (CLKC Register). Once this process (inner ring) is started, no further setup is required to continue the low frequency operation. If system activity is detected (Break Events), MPIIX negates STPCLK# immediately.
2. The first software controlled STPCLK# sequence (middle ring) is initiated by a read of APMC and enters the Stop Grant state if the STPCLK_MODE bits are set to enable the Stop Grant state. The system remains in the Stop Grant state until a Break Event occurs. Then the MPIIX negates the STPCLK# signal to the CPU.
3. The second software controlled STPCLK# sequence (outside ring) is similar to the previous sequence. However, since the STPCLK_MODE bits are set to enable the Stop Clock state, MPIIX brings the CPU down to the Stop Clock state by stopping the external clock (HCLK) to the CPU complex. The CPU remains in this state until a Break Event occurs. Then the MPIIX must start the HCLK again and wait for the internal CPU clock to start up before negating the STPCLK# signal to the CPU.

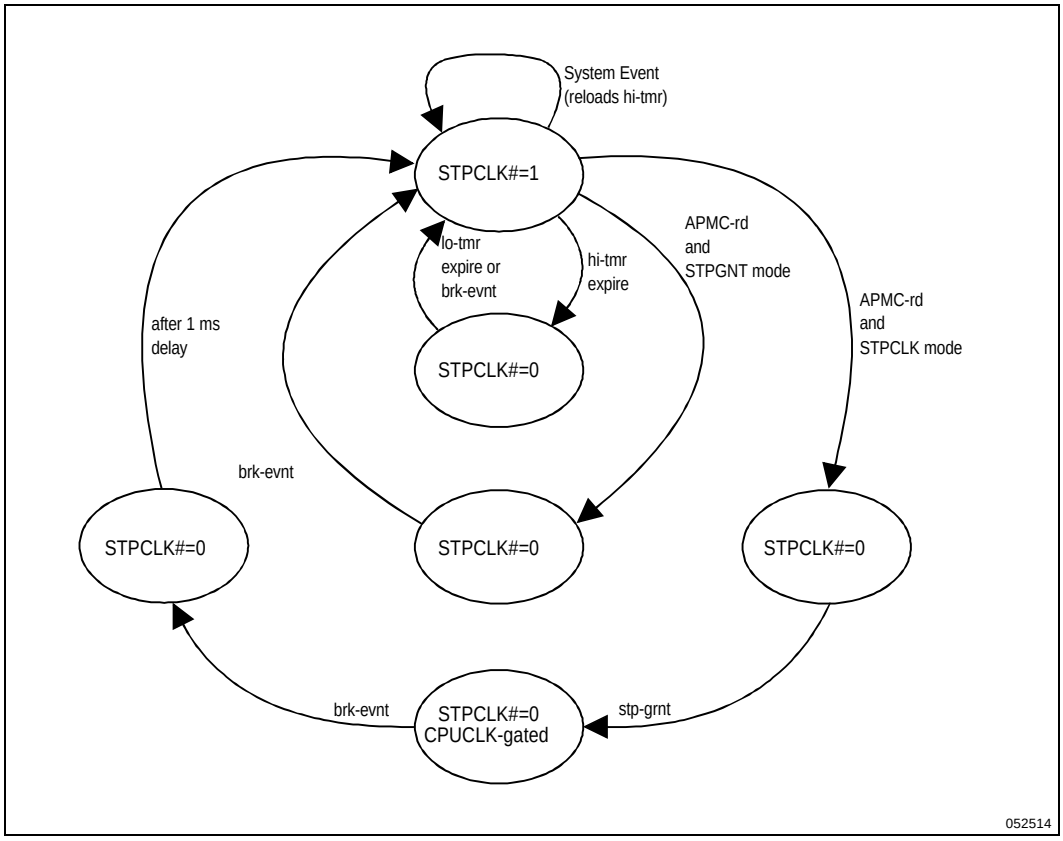



**Figure 14. STPCLK Control State Machine**

#### 4.8.2.5. Auto Clock Throttle (ACT) Feature

The Auto Clock Throttle mode is similar to the Clock Throttle mechanism but is designed to allow the use of "clock throttle" and, at the same time, handle break events that require more time to complete than is allowed by the STPCLK High Timer (programmed via the STPCLKHT register). The Auto Clock Throttle mechanism provides 3 groups of events (Table 11) that can "break" out of the Stop Grant state: the STPCLKHT/STPCLKLT timers, the Burst Clock Events (BSTCLKEE[6:0] register), or the Clock Throttle Break Events (CLKTHLBRKE register). These can be considered "Short", "Medium", and "Long" bursts of time. When a break event occurs or the STPCLK Low timer expires, the MPIIX negates STPCLK# to the CPU for the amount of time assigned to that break event.

MPIIX negates the STPCLK# signal as soon as the STPCLK Low Timer has expired. Then MPIIX transitions between the STPCLK High Timer state and the STPCLK Low timer state, according to the values programmed into the STPCLKHT and STPCLKLT Registers until a break event occurs.

**Table 11. Break Even Groups For Auto Clock Throttle**

| Break Event Group | Timer | Resolution | Count | Min* | Max |
|---|---|---|---|---|---|
| CLKTHL_BRK Event | Clock Throttle Standby Timer | 4ms | 2 - 256 | 8 ms | 1 sec |
| | | (or 32ms) | (2 - 256) | (64 ms) | (8 sec) |
| BSTCLK Event | Burst CLock Timer | 32us | 2 - 256 | 64 us | 8 ms |
| (throttle ratio timers) | STPCLK Low Timer | 32us | 2 - 256 | 64 us | 8 ms |
| | STPCLK High Timer | 32us | 2 - 256 | 64 us | 8 ms |

(*Minimum count is 2 or greater since actual value is ± 1 resolution.)

The Auto Clock Throttle Mode is enabled by setting ACT_MODE_EN bit in the Clock Control Register (offset D4h). The MPIIX asserts the STPCLK# signal AFTER the Clock Throttle Standby timer expires. This is the longest delay of the 3 timers groups. While the STPCLK# signal is asserted the CPU is in the Stop Grant Mode since MPIIX continues to run the HCLKO signal. The break events for each group are listed in the Table 12.

**Table 12. Auto Clock Throttle Break Events**

| Stop Break Events STPBRKEx Register | Burst Clock Events BSTCLKEE Register | Clock THRTL Break Events CLKTHLBRKEE Register |
|---|---|---|
| IRQs | IRQs | IRQs |
| INTR | | INTR |
| NMI | | NMI |
| SMI# | SMI# | SMI# |
| EXTSMI# | EXTSMI# | EXTSMI# |
| COMRI# | COMRI# | COMRI# |
| BATLOW# | | BATLOW# |
| SRBTN# | | SRBTN# |
| (see note:) | PHLDA#<br>EXTEVNT#<br>Fixed Peripheral Decode | PHLDA#<br>EXTEVNT#<br>Fixed Peripheral Decode |
| | Programmable I/O Decode | Programmable I/O Decode |
| | Programmable Memory Decode | Programmable Memory Decode |

Note: These "Stop Break" events only apply to the Stop Clock mode and the Clock Throttle mode

**Table 13. Fixed Peripheral Decode**

| Name | I/O Address |
|---|---|
| Keyboard Ports | 60h, 64h |
| IDE-Primary<br>IDE-Secondary | 01F0h-01F7h,03F6h<br>0170h-0177h,0376h |
| FDC-Primary<br>FDC-Secondary | 03F0h-03F5h, 03F7h<br>0370h-0375h, 0377h |
| Serial Port 1 (COM1)<br>Serial Port 2 (COM2)<br>Serial Port-3 (COM3)<br>Serial Port-4 (COM4) | 03F8h-03FFh,<br>02F8h-02FFh,<br>03E8h-03EFh,<br>02E8h-02EFh |
| Parallel Port 1<br>Parallel Port 2<br>Parallel Port 3 | 03BCh-03BFh(07BCh-07BEh)<br>0378h-037Fh(0778h-077Ah)<br>0278h-027Fh(0678h-067Ah) |
| Audio_A<br>Audio_B<br>Audio_C<br>Audio_D<br>Audio_E | 0220h-022Fh<br>0230h-023Fh<br>0240h-024Fh<br>0250h-025Fh<br>0388h-038Bh |

**Table 14. Programmable I/O and Memory Decodes**

| Decode Register | Programmable Register Offset | Mask Register Offset |
|---|---|---|
| **I/O Decode Registers** | | |
| PCSC Register | 92h (16 bits) | 9Ah (bits 3–0) |
| PAC1 | 94h (16 bits) | 9Ah (bits 7–4) |
| PAC2 | 96h (16 bits) | 9Bh (bits 3–0) |
| PAC3 | 98h (16 bits) | 9Bh (bits 7–4) |
| PAC4 | A0h (16 bits) | A4h (bits 3–0) |
| PAC5 | A2h (16 bits) | A4h (bits 7–4) |
| **Memory Decode Register** | | |
| PMAC0 | 8A (16 bits) | 8Eh |
| PMAC1 | 8C (16 bits) | 8Fh |

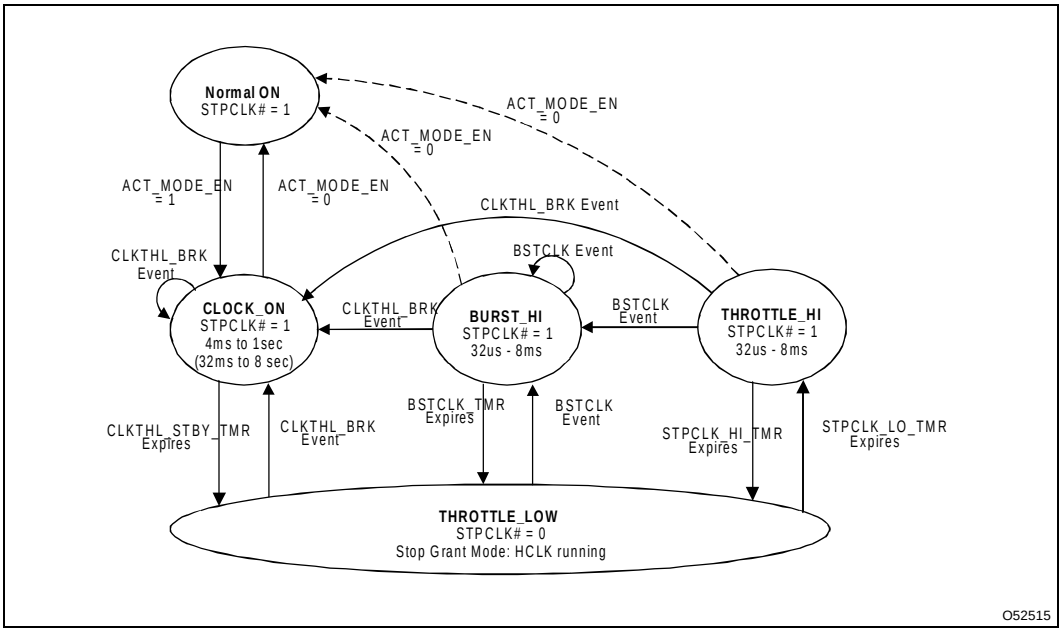



**Figure 15. Auto Clock Throttle State Diagram**

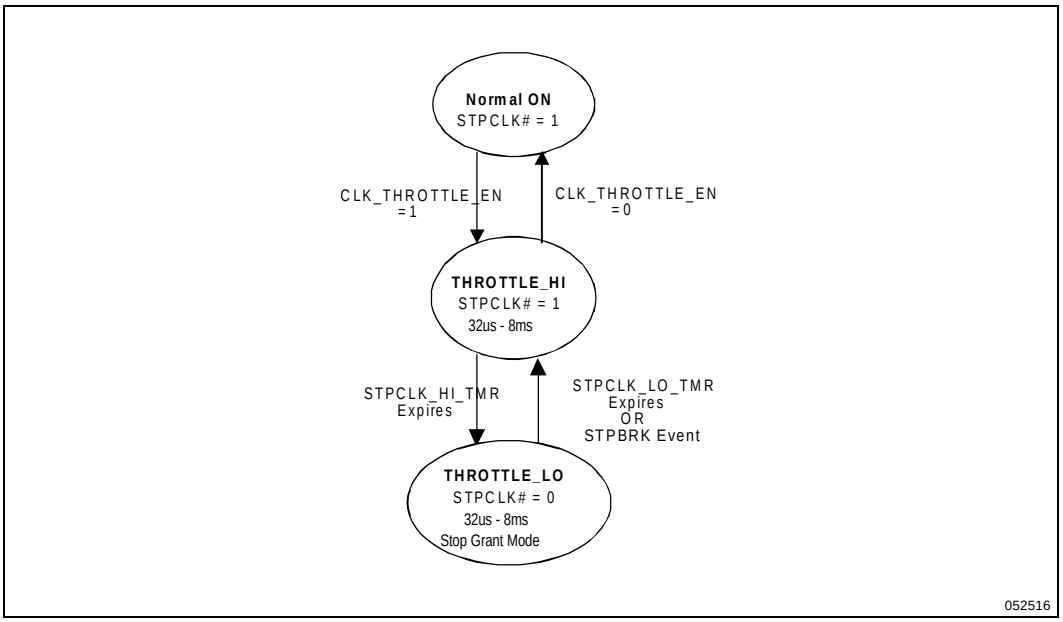



**Figure 16. Clock Throttle State Diagram**

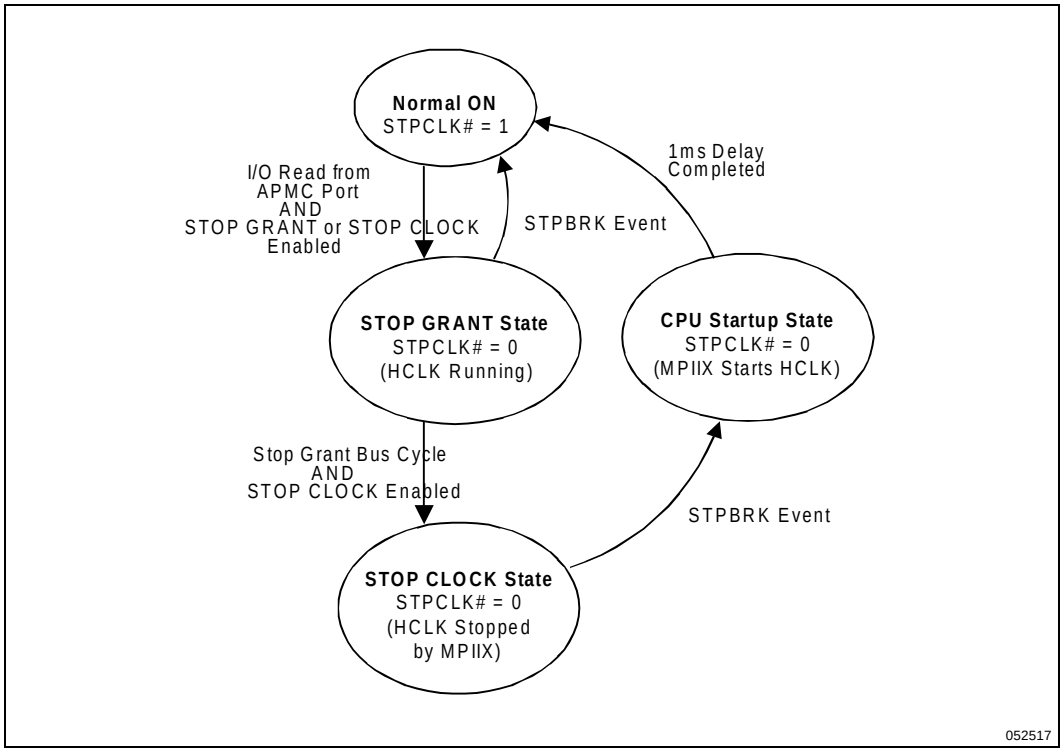



**Figure 17. Stop Clock State Diagram**

#### 4.8.3. LOCAL STANDBY (PERIPHERAL MANAGEMENT)

The system management places peripherals in a low power state when they have been inactive for long periods of time. The MPIIX Local Standby hardware enables the system to identify idle devices (Idle Timers, SMI# generation, Idle Status), put them into a low power "standby" state (SMOUTs, Leakage Control), and trap accesses to powered-down peripherals (Trap Ranges, Synchronous SMI# Generation, Trap Status). MPIIX provides resources to manage 6 local devices, IDE, Audio, COM port, and 3 user programmable.

##### 4.8.3.1. Local Standby Sequence

**Setup:** The system's power management setup initializes the Local Trap Access I/O address ranges and the Idle Timer counter for each peripheral device. Then the setup sets the global SMI enable for the Local Standby Idle Timers and the global SMI enable for the Local Trap Access. (The global SM_FREEZE bit that enables all timers must be cleared to allow idle timers to count down.)

**On-to-Off Transition**: When power management software enables the idle timer for that device, the Idle Timer begins to count down. Any access to a peripheral device's I/O address reloads that device's Idle Timer. When the Idle timer expires, the Local Standby SMI Request status bits are set and SMI# is generated. (Both the global local standby request status and the specific local standby request status for that device are set.) The software places the peripheral device into a low power state, disables the Idle Timer hardware, and enables the Local Trap hardware.

**Off-to-On Transition**: Power management software enables the Local Trap hardware by setting the Local Trap SMI enable. When the system requires an I/O access to that device range, the access is trapped, an SMI# is generated, and the corresponding Local Trap SMI request indication bit are set. (Both the global Local Trap SMI Request status and the specific Local Trap SMI Request status bits are set.) The software then places the peripheral device in an on state, clears the Local Trap SMI status bits, then enables the Local Standby Idle Timer hardware.

#### 4.8.3.2. Access Ranges

The IDE and Audio ranges are selected when the devices are configured, so no further action is required to setup these ranges prior to enabling the access monitoring or trapping. The COM port and programmable trap ranges must be setup prior to enabling the access monitoring or trapping.

- IDE—I/O address range trap, for either the Primary or Secondary IDE ranges, whichever is enabled.
  - — 1F0h to 1F7h, 3F4h to 3F7h I/O reads and writes are trapped if the primary interface is enabled.
  - — 170h to 177h, 374h to 377h I/O reads and writes are trapped if the secondary interface is enabled.
- Audio and FM Synthesis—I/O address traps.
  - — 0201h, 02x0h–02xFh, 388h–38Bh I/O reads and writes where x= 2, 3, 4, or 5.
  - — 2xAh, 2xEh – I/O reads where x= 2, 3, 4, or 5.
  - — MDAK1 or MDAK2 can be enabled to cause the Audio Local Standby timer to be reloaded.
- COM ports—One or all of the following I/O address options are selected by programming the LTMDEV3 Register. When an access occurs to an enabled range, the Local Standby COM Timer is re-loaded.
  - — 3F8–3FFh.
  - — 2F8–2FFh.
  - — 2E8–2EFh.
  - — 3E8–3EFh.
- Three Programmable I/O Address traps for CPU driven cycles on the PCI Bus.
  - — 16-bit I/O port base register, for PCI I/O address bits [15:0]. (TRP_ADR_xxx).
    - DEV1 uses Extended I/O Programmable Chip Select (PCS#) Range.
    - DEV2 uses Extended I/O decode Programmable Address Range #1.
    - DEV3 uses a independent Local Trap Address Range.
  - — 4-bit I/O port mask register, for PCI I/O address bits [3:0]. (TRP_MSK_xxx).
  - — The PCI trap logic compares the 16 least significant bits, while checking that PCI I/O address bits [31:16] are all 0. This provides address trapping of the PCI low 64 Kbytes, while not aliasing in address ranges above 64 Kbytes.

#### 4.8.3.3. Idle Timers

MPIIX provides 6 idle timers (three timers for the programmable device access ranges, one timer per IDE, COM and Audio/FM). When an idle timer is enabled, it will count down until its corresponding access monitor detects device activity. At that time, the timer is reloaded with the initial count. If the timer expires, an SMI# is generated.

**Enable:** The Local Standby Idle Timers are globally enabled by setting the LSTBY_SMI_EN bit in the GSMIE Register. When this bit is set to 1, the enabling of the individual timer is controlled by the LSTBY_SMI_EN_xxx for that specific device.

**Count:** All timers use an 8 second internal clock period and can be programmed with an 8-bit value (up to 255, 00h is not legal) for an idle time-out range of 8 seconds to 34 minutes. This timer is reloaded with the initial count value when there is an access to the device address or when the LSTBY_SMI_EN_xxx enable bit is set for that device.

**Freeze:** The local idle timers will be frozen (i.e. enabled timers will stop the count down, while maintaining the same values) when the SM_FREEZE bit is set. The SM_FREEZE bit is used by power management software when a long service routine is started. The same software routine clears the SM_FREEZE bit before returning to normal operation.

**SMI Status:** Local Idle SMI status is recorded in the GSMIS Register in the LSTBY_STAT bit when any local standby timer generates an SMI#. Individual SMI status is recorded in the LSTBY_STAT_xxx bit for that particular device.

#### 4.8.3.4. Access Traps

Before power management software leaves the SMI# handler that places a device into a standby state, it must enable the access trap for that device. Any access to an enabled trap range causes a synchronous SMI# to be generated, so that power management software can return the device to an On state. Before leaving the power-up routine, power management should enable the idle timer hardware to begin the cycle again.

**Synchronous SMI#:** The I/O trap SMI# is synchronous to the completion of the I/O instruction in the CPU. The I/O instruction is completed when the Ready (RDY#, BRDY#) is returned to the CPU. MPIIX coordinates the assertion of SMI# to the CPU with the generation of Ready to the CPU by the Mobile System Controller (MTSC) such that the SMI# is generated at least 3 HCLKs before Ready is generated.

**Enable:** The Local Traps are globally enabled by setting the LTRP_SMI_EN bit in the GSMIE Register. When this bit is set to 1, the enabling of the individual traps is controlled by the LTRP_SMI_EN_xxx for that specific device.

**SMI Status:** Local Trap SMI status is recorded in the GSMIS Register in the LTRP_STAT bit when any local trap generates an SMI#. Individual SMI status is recorded in the LTRP_STAT_xxx bit for that particular device.

#### 4.8.3.5. SMOUT Programmable Outputs

Six output signals (SMOUT[5:0]) can be individually programmed to a 0 or 1 via the SMOUTC Register. These signals provide the SL flexibility to control system power planes and isolation buffer output enables. It is assumed that the above devices will be power managed in a centralized manner, while other devices will be managed by the Keyboard (example: video), or self controlled (example: Super I/O chips).

### 4.8.4. SUSPEND

The Intel 430MX PCIset provides hardware to support two types of suspend—Suspend-to-DRAM and Suspend-to-Disk. The different suspend modes differ in the power saving and resume sequence latencies. The following features are common to both suspend modes:

- Suspend is initiated by power management software in response to one of the special suspend SMI# events, Suspend Resume Button (SRBTN#) and Battery Low (BATLOW#), or any other SMI#, depending on the power management strategy.
- SMI# delay timers allow system activity to complete before starting the Suspend sequence.
- During the Suspend mode, only the 32 KHz RTC clock is active. All other clocks are stopped.
- MPIIX provides low power Resume logic that is clocked by the 32 KHz RTC clock. This allows the system to use minimum power while monitoring Resume Events (COMRI#, IRQ8#, SRBTN#, and EXTSMI#). The

BATLOW# signal prevents a resume in the event of a low battery. A RSMRST# signal is also provided to ensure that the system is completely reset if the system state is corrupted when the battery backup power is lost.

- The SUSTAT# output signal provides an indication to the system components and power supply that they should enter their corresponding suspend mode.
- AT Write-only registers are Shadowed so that the system state can be completely read and saved.
- If DRAM remains active, the suspend refresh is activated.

**Suspend-to-DRAM**

This mode eliminates the leakage current found in the system by removing power from all components. However, there is a longer Resume latency, since it must reset and initialize the system. In this mode, power is removed from the all system components except the DRAM, Real Time Clock, and the Suspend Refresh logic portion of the MTSC and the MPIIX. MTSC has a separate VCC input for this logic. The SUSTAT# signal is asserted to inform the system that it can switch off power supplies.

**Suspend-to-Disk**

This mode eliminates the power consumption of the DRAM Refresh and has a longer Resume latency than Suspend-to-DRAM, since it must restore the system state from disk. Suspend-to-Disk is similar to Suspend-to-DRAM except that the DRAM state must be saved and restored. Only the Resume logic and RTC is active during this mode.

#### 4.8.4.1. Suspend mode selects

For the correct resume sequence to take place, the power management software enables the appropriate suspend mode in the SUSRSMC2 Register. These bits are powered by the ”Resume Well”.

#### 4.8.4.2. Suspend SMI# Requests (SRBTN# and BATLOW#)

There are two special purpose hardware SMI# sources (SRBTN# and BATLOW# input signals), that are generally used to indicate that a suspend is requested. Both of these SMI# sources can be disabled by writing a 0 to the SUSP_SMI_EN bit. When the SUSP_SMI_EN bit is written to a 1, the individual SMI enables determine if the source is enabled. Both of these sources share the Suspend SMI Delay Timer that provides time for the system to complete current bus master or docking bridge activity.

**SRBTN#.** The Suspend/Resume Button input generates an SMI# request to the Suspend SMI Delay Timer, if the SUSP_SMI_EN_SRBTN bit is set to 1 in the in the MISCSMIE Register. When the request is generated, the delay timer begins to count down. The SRBTN# and EXTSMI# inputs are monitored during a Suspend and can be used to bring the system out of suspend. (Note: SMI# is not generated by the SRBTN# or EXTSMI# while in suspend. It is the responsibility of the power management resume routine to generate a software SMI to complete the resume.)

**BATLOW#.** During normal operating mode the Battery Low signal is used to generate a suspend request SMI#. and when the system is in suspend, this signal is used to prevent the system from resuming. The Battery Low input signal will generate an SMI# request to the Suspend SMI Delay Timer, if the SUSP_SMI_EN_BATLOW bit is set to 1 in the MISCSMIE Register. When the BATLOW# signal is asserted, the delay timer begins to count down. The Suspend SMI Delay Timer will stop counting if BATLOW# negates prior to timer reaching zero. It will resume counting when BATLOW# reasserts or SRBTN# asserts. MPIIX also provides the option to bypass the SMI# delay timer for a critically low battery condition. BATLOW# assertion generates an immediate SMI# when the BATLOW_BYPASS_EN bit is set to 1 in the SUSRSMC1 Register. The BATLOW# input is monitored during a Suspend and can be used to prevent the system from resuming. When the SUSP_SMI_EN_BATLOW bit is set to a 1, the Battery Low signal prevents MPIIX from initiating a resume.

**Suspend DRAM Refresh**

In the Suspend to DRAM mode, MTSC uses the 32 KHz RTC clock to trigger the refresh on the falling and rising edges. The refresh mode selection is described in the MTSC data sheet. Since the HCLK is stopped, the MTSC uses a ring oscillator to generate the RAS pulses. MTSC disables the ring oscillator if the suspend refresh mode is set for DRAMs that support Self-Refresh. The suspend refresh is enabled by writing to the SUS_REF bit in the SUSRSMC1 Register. When this bit is set the SUS_STAT bit is also set, asserting the SUSTAT# signal. The MTSC suspend refresh logic shadows the SUS_REF bit. When the SUS_REF bit is set, the MTSC enables the suspend refresh.

#### 4.8.4.3. Suspend Status (SUSTAT#) Signal and Register

Power management software will set the SUS_STAT bit in the SUSRSMC1 Register at the very last stage of the suspend transition to activate the hardware suspend sequence and resume logic. The SUS_STAT bit can be directly set by the handler for Suspend-to-Disk or it can be automatically set when the handler enables suspend-refresh for Suspend-to-DRAM.

When Suspend-to-DRAM or Suspend-to-Disk is enabled and the SUS_STAT bit is set, the SUSTAT# output signal is driven by the MPIIX. This signal can be used by system components that transition to low power mode during suspend. In the Suspend-to-Disk mode, the RTC battery voltage is used as the power source for the logic that drives the SUSTAT# output.

When a Resume Event triggers a system resume, MPIIX negates SUSTAT#. The inactive edge of SUSTAT# can be used to apply power to the powered down components. The Resume logic initiates a reset sequence, with some delay, following the SUSTAT# negation.

**Resume Event Logic**

The resume logic is triggered to exit suspend by the following events:

- RTC alarm (IRQ8#).
- UART ring indication (COMRI#).
- External SMI (EXTSMI#).
- Suspend Resume button press (SRBTN#).

These signals, as well as other resume logic, are in a “Resume Well” that is powered by the RTC battery or the DRAM power supply that remains powered during Suspend-to-DRAM. For the SRBTN# to be recognized as a resume event, the system must be in a suspend mode (the SUS_STAT bit is set).

Some of the resume events (COMRI#, IRQ8#, BATLOW#, and EXTSMI#) can be masked such that the system will not resume as a result of active - masked event. These masks are set in the SUSRSMC[1,2] Registers. Battery Low indication (BATLOW#) can be masked such that it does not prevent a resume, when both BATLOW# and a resume event are active. The MPIIX privides a Resume Reset input (RSMRST#) to reset the entire system (including Resume logic and the RTC content) when the RTC power can not sustain a valid suspend mode.

The IRQ8# input has an internal 8 KΩ pull-up resister that must always be enabled to maintain a valid logic level on this input. When the IRQ8# input is masked as a resume event, the interrupt must be disabled at the RTC to prevent a DC path across the MPIIX internal pull-up resistor.

The COMRI# input has an internal 50 KΩ pull-up resistor that maintains a valid logic level on this input when it is not connected to any device. If the COMRI# input is left un-connected, the COMRI# input should NOT be masked as a resume event. When the COMRI# input is masked as a resume event, the COMRI# input pull-up resistor is disabled while in suspend mode. If the device that generates COMRI# is powered during suspend and the COMRI# input is masked as a resume event, the internal pull-up is disabled and there will not be a DC path. If the device that generates COMRI# is powered down during suspend, the COMRI# input should be masked as a resume event. This disables the MPIIX internal pull-up resistor and prevents a DC path between VCC and the COMRI# device's power plane (the COMRI# signal to float to ground in this case). The COMRI# input is masked as a resume event in the default state after PCIRST#.

The EXTSMI# signal has an internal 8 KΩ pull-up resistor. When EXTSMI# is masked as a resume event, the pull-up resistor is disabled while the system is in suspend mode. If the EXTSMI# input is left un-connected, the EXTSMI# input must NOT be masked as a resume event. This prevents the EXTSMI# input from floating since the internal pull-up resistor is enabled. After PCIRST# the EXTSMI# input defaults to NOT masked as a resume event.

Power management software can pole the real-time clock, COMRI# device, and EXTSMI# devices to determine the source of the resume event. If none of these devices caused the resume event, the SRBTN# was the default cause of the resume.

If the RTC alarm (attached to IRQ8#) is the resume event, it must be cleared by the resume SMI handler. The 8259-compatible programmable interrupt controller is not powered during suspend and an interrupt will not be generated. A read to the RTC Status Register (I/O address 0Ch) clears the alarm status bit and causes IRQ8# to be negated.

Battery Low indication (BATLOW# signal) can be masked such that it will not prevent a resume when both BATLOW# and a resume event are active. The MPIIX privides a Resume RESET input (RSMRST# signal) to reset the entire system (including Resume logic and the RTC content) when the RTC power can not sustain a "valid" suspend mode.

#### 4.8.4.4. POWER PLANE CONTROL

The SUSTAT# signal indicates that the system logic has entered a suspend mode and that the appropriate power planes can be turned off. For suspend-to-DRAM, the MPIIX (and MTSC component) requires power to the "Resume Well". DRAM (and Video RAM) also requires power during this suspend state. Additional logic is required to gate the SUSTAT# signal to the power planes that supply the power to the "Resume Well" and DRAM. This gate should be controlable by power management software through the keyboard controller or other programmable logic.

**Timing Requirements**

- **PWROK.** All power supplies must be stable for at least 1ms before PWROK is asserted to the MPIIX/MTSC.
- **PWRSD.** The **VDDM DRAM** and **VDDR MTSC Resume Well** power supplies must be stable for at least 1 ms before PWRSD is asserted to the MTSC.
- **CPU and PCI Reset.** The Power Supply and Clocks (HCLK, PCICLK) must be stable for 1ms before CPU and PCI Resets are de-asserted. MPIIX drives the PCIRST# signal for at least 1 ms after the PWROK signal is asserted. MPIIX drives the CPURST signal for at least 2 ms after the PWROK signal is asserted.

**Notes:**

1. The SUSTAT# signal from the MPIIX is used to switch off the power supplies when entering suspend mode. Systems that require both Suspend-to-DRAM and Suspend-to-DISK must distinguish between the two different suspend modes to turn off the appropriate power supplies when SUSTAT# asserted.

2. The voltage on the RSMRST# (MPIIX VDDR Resume Well) and MTSC VDDR Resume Well power supplies are at 5V in normal mode. However, they can drop to a 3.3V level during Suspend to DRAM. (During Suspend-to-DRAM, the 5V supply might be turned off if 3.3V DRAM is used. )
3. The voltage on the RSMRST# and MPIIX VDDR Resume Well power supplies are normally at 5V. However, they can drop to the voltage level of the Real-Time Clock power supply during Suspend-to-DISK.

#### 4.8.4.5. SHADOW REGISTERS

MPIIX includes a set of shadow registers for the standard AT write-only registers. In the transition to Suspend mode, the contents of these registers are saved so the system state can be restored, when resumed. The shadowed registers can be read back through the SHDW Register.

The SHDW Register contains a counter that points to a shadow register. The counter is initialized upon writing to the SHDW Register. When the SHDW Register is read, it returns the data from the shadow register pointed by the counter. The counter increments the count every time the software reads the register.

### 4.8.5. SUMMARY OF TIMER RANGES

**Table 15. Timer Resolutions and Maximum Counts**

| Timer | Count | Resolution | Maximum |
|---|---|---|---|
| STPCLKHT | 255 | 34 us | 8 ms |
| STPCLKLT | 255 | 34 us | 8 ms |
| CLK_START_DLY | 4 | 500 us | 2 ms |
| LSTBY_TMR_xxx | 255 | 8s | 34 min |
| GSTBY_TMR | 255 | 8s | 34 min |
| SWEXT_SMI_DLY_TMR | 255 | 1 ms | 255 ms |
| SUSP_SMI_DLY_TMR | 255 | 128 ms | 32s |

## 4.9. Reset Support

The MPIIX integrates the system reset logic for the system and generates CPURST, PCIRST#, and RSTDRV during power up (PWROK) and when a hard reset is initiated through the RC register. CPURST is asserted for 2 mS after the assertion of PWROK and PCIRST# is asserted for 1 ms after the assertion of PWROK.

The following MPIIX signals interface directly to the processor:

- CPURST
- INIT
- INTR
- NMI
- IGNNE#
- SMI#
- STPCLK#

These signals are open drain so that external logic is not required for interface with the processors based on 3.3V technology which do no support 5V tolerant input buffers. During power-up these signals are driven low to prevent problems associated with 5V/3.3V power sequencing.

Some PCI devices may drive 3.3 V friendly signals directly to 3.3 V devices that are not 5 V tolerant. If such signals are powered from the 5 V supply, they must be driven low when PCIRST# is asserted. Some of these signals may need to be driven high before CPURST is negated. PCIRST# is negated 1 to 2 ms before CPURST to allow time for this to occur.

# 5.0. PINOUT AND PACKAGE INFORMATION

## 5.1. Pinout Information

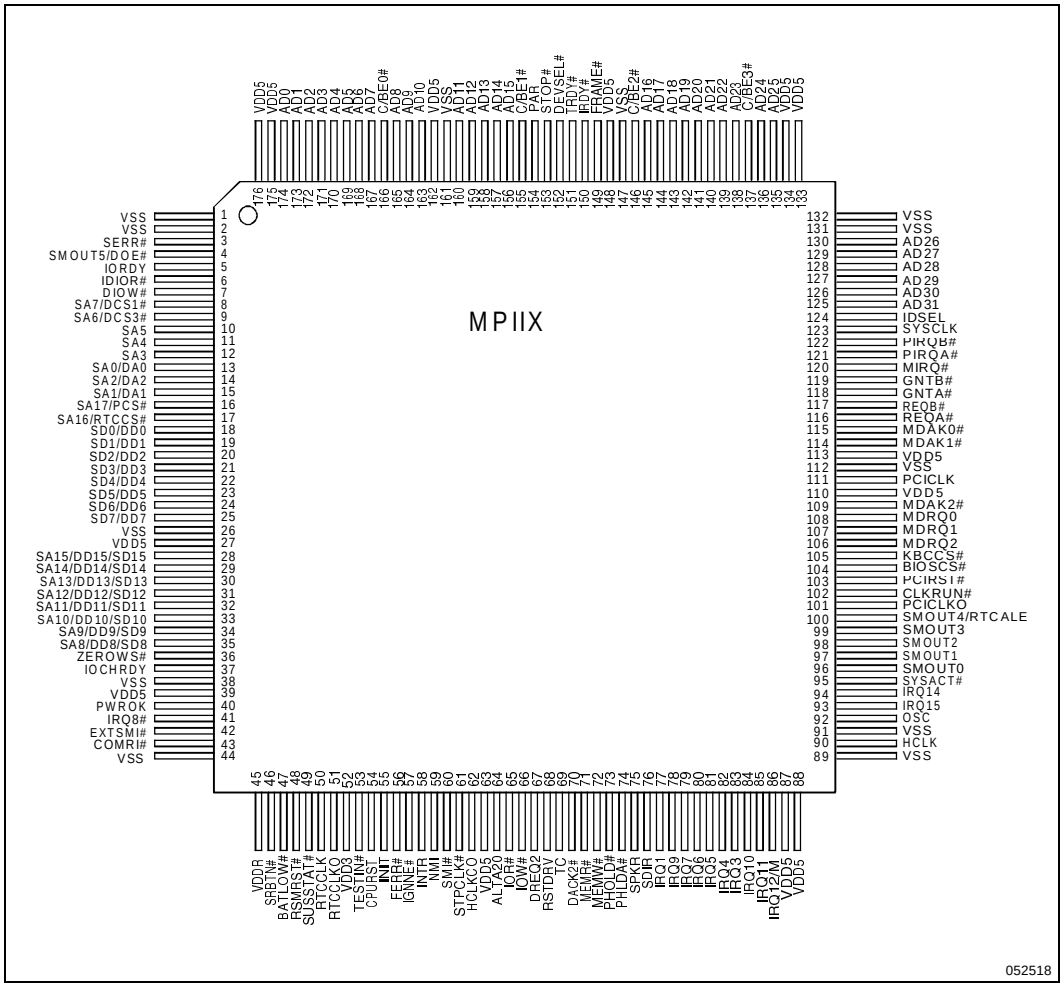



**Figure 18. MPIIX Pinout Diagram**

**Table 16. Alphabetical Pin Assignment**

| Name | Pin # | Type |
|---|---|---|
| AD0 | 174 | I/O |
| AD1 | 173 | I/O |
| AD2 | 172 | I/O |
| AD3 | 171 | I/O |
| AD4 | 170 | I/O |
| AD5 | 169 | I/O |
| AD6 | 168 | I/O |
| AD7 | 167 | I/O |
| AD8 | 165 | I/O |
| AD9 | 164 | I/O |
| AD10 | 163 | I/O |
| AD11 | 160 | I/O |
| AD12 | 159 | I/O |
| AD13 | 158 | I/O |
| AD14 | 157 | I/O |
| AD15 | 156 | I/O |
| AD16 | 145 | I/O |
| AD17 | 144 | I/O |
| AD18 | 143 | I/O |
| AD19 | 142 | I/O |
| AD20 | 141 | I/O |
| AD21 | 140 | I/O |
| AD22 | 139 | I/O |
| AD23 | 138 | I/O |
| AD24 | 136 | I/O |
| AD25 | 135 | I/O |
| AD26 | 130 | I/O |
| AD27 | 129 | I/O |
| AD28 | 128 | I/O |
| AD29 | 127 | I/O |
| AD30 | 126 | I/O |
| AD31 | 125 | I/O |
| ALTA20 | 64 | O |
| BATLOW# | 47 | I |

| Name | Pin # | Type |
|---|---|---|
| BIOSCS# | 104 | O |
| C/BE0# | 166 | I/O |
| C/BE1# | 155 | I/O |
| C/BE2# | 146 | I/O |
| C/BE3# | 137 | I/O |
| CLKRUN# | 102 | I/O |
| COMRI# | 43 | I |
| CPURST | 54 | OD |
| DACK2# | 70 | O |
| DEVSEL# | 152 | I/O |
| DIOR# | 6 | O |
| DIOW# | 7 | O |
| DREQ2 | 67 | I |
| EXTSMI# | 42 | I |
| FERR# | 56 | I |
| FRAME# | 149 | I/O |
| GNTA# | 118 | O |
| GNTB# | 119 | O |
| HCLK | 90 | I |
| HCLKCO | 62 | O |
| IDSEL | 124 | I |
| IGNNE# | 57 | OD |
| INIT | 55 | O |
| INTR | 58 | O |
| IOCHRDY | 37 | I |
| IOR# | 65 | O |
| IORDY | 5 | I |
| IOW# | 66 | O |
| IRDY# | 150 | I/O |
| IRQ1 | 77 | I |
| IRQ10 | 84 | I |
| IRQ11 | 85 | I |
| IRQ12/M | 86 | I |
| IRQ14 | 94 | I |

| Name | Pin # | Type |
|---|---|---|
| IRQ15 | 93 | I |
| IRQ3 | 83 | I |
| IRQ4 | 82 | I |
| IRQ5 | 81 | I |
| IRQ6 | 80 | I |
| IRQ7 | 79 | I |
| IRQ8# | 41 | I |
| IRQ9 | 78 | I |
| KBCCS# | 105 | O |
| MDAK0# | 115 | O |
| MDAK1# | 114 | O |
| MDAK2# | 109 | O |
| MDRQ0 | 108 | I |
| MDRQ1 | 107 | I |
| MDRQ2 | 106 | I |
| MEMR# | 71 | O |
| MEMW# | 72 | O |
| MIRQ# | 120 | I |
| NMI | 59 | O |
| OSC | 92 | I |
| PAR | 154 | O |
| PCICLK | 111 | I |
| PCICLKO | 101 | O |
| PCIRST# | 103 | O |
| PHLDA# | 74 | I |
| PHOLD# | 73 | O |
| PIRQA# | 121 | I |
| PIRQB# | 122 | I |
| PWROK | 40 | I |
| REQA# | 116 | I |
| REQB# | 117 | I |
| RSMRST# | 48 | I |
| RSTDRV | 68 | O |
| RTCCLK | 50 | I |
| RTCCLKO | 51 | O |
| SA0/DA0 | 13 | O |

| Name | Pin # | Type |
|---|---|---|
| SA1/DA1 | 15 | O |
| SA2/DA2 | 14 | O |
| SA3 | 12 | O |
| SA4 | 11 | O |
| SA5 | 10 | O |
| SA6/DCS3# | 9 | O |
| SA7/DCS1# | 8 | O |
| SA8/DD8/ SD8 | 35 | I/O |
| SA9/DD9/ SD9 | 34 | I/O |
| SA10/DD10/ SD10 | 33 | I/O |
| SA11/DD11/ SD11 | 32 | I/O |
| SA12/DD12/ SD12 | 31 | I/O |
| SA13/DD13/ SD13 | 30 | I/O |
| SA14/DD14/ SD14 | 29 | I/O |
| SA15/DD15/ SD15 | 28 | I/O |
| SA16/ RTCCS# | 17 | O |
| SA17/PCS# | 16 | O |
| SD0/DD0 | 18 | I/O |
| SD1/DD1 | 19 | I/O |
| SD2/DD2 | 20 | I/O |
| SD3/DD3 | 21 | I/O |
| SD4/DD4 | 22 | I/O |
| SD5/DD5 | 23 | I/O |
| SD6/DD6 | 24 | I/O |
| SD7/DD7 | 25 | I/O |
| SDIR | 76 | O |
| SERR# | 3 | I |
| SMI# | 60 | O |
| SMOUT0 | 96 | O |
| SMOUT1 | 97 | O |

| Name | Pin # | Type |
|---|---|---|
| SMOUT2 | 98 | O |
| SMOUT3 | 99 | O |
| SMOUT4/ RTCALE | 100 | O |
| SMOUT5/ DOE# | 4 | O |
| SPKR | 75 | O |
| SRBTN# | 46 | I |
| STOP# | 153 | I/O |
| STPCLK# | 61 | O |
| SUSSTAT# | 49 | O |
| SYSACT# | 95 | I |
| SYSCLK | 123 | O |
| TC | 69 | O |
| TESTIN# | 53 | I |
| TRDY# | 151 | I/O |
| ZEROWS# | 36 | I |
| VDD3 | 52 | V |
| VDD5 | 39 | V |
| VDD5 | 87 | V |
| VDD5 | 110 | V |
| VDD5 | 134 | V |
| VDD5 | 175 | V |
| VDD5 | 27 | V |

| Name | Pin # | Type |
|---|---|---|
| VDD5 | 63 | V |
| VDD5 | 88 | V |
| VDD5 | 113 | V |
| VDD5 | 133 | V |
| VDD5 | 148 | V |
| VDD5 | 162 | V |
| VDD5 | 176 | V |
| VDDR | 45 | V |
| VSS | 2 | V |
| VSS | 38 | V |
| VSS | 91 | V |
| VSS | 131 | V |
| VSS | 1 | V |
| VSS | 26 | V |
| VSS | 89 | V |
| VSS | 112 | V |
| VSS | 132 | V |
| VSS | 147 | V |
| VSS | 161 | V |
| VSS | 44 | V |

## 5.2. Package Information

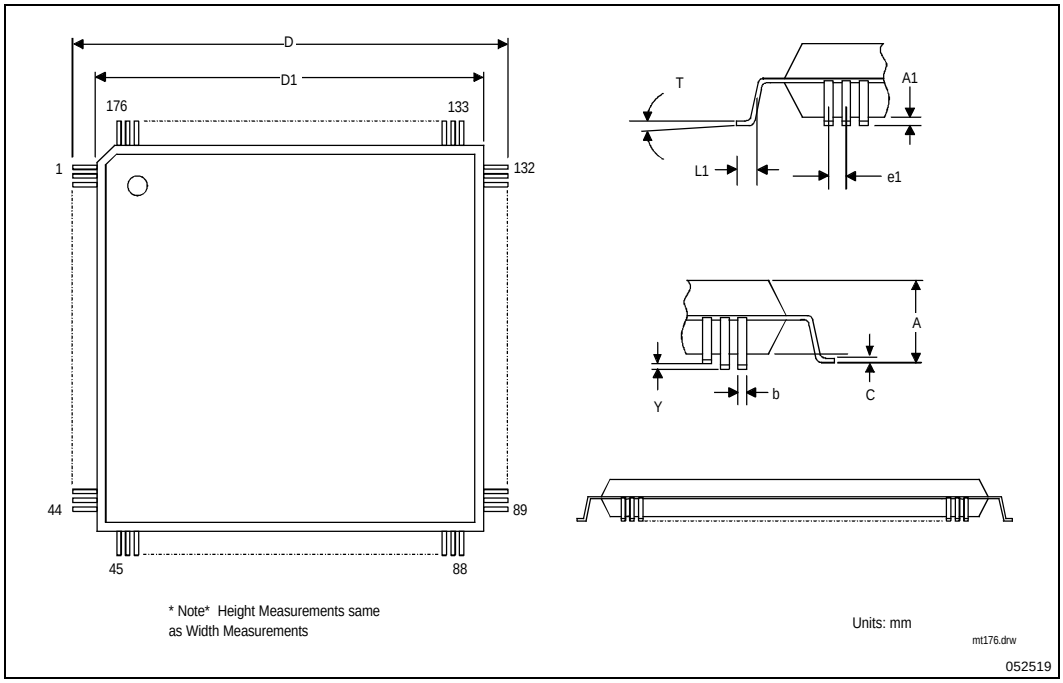



**Figure 19. MPIIX Physical Dimensions (176-lead TQFP)**

**Table 17. MPIIX Physical Dimensions (176-lead TQFP)**

| Symbol | Dimension in Milimeters | | |
|---|---|---|---|
| | Minimum | Nominal | Maximum |
| A | | | 1.7 |
| A1 | 0.0 | 0.1 | 0.2 |
| b | 0.13 | 0.22 | 0.28 |
| C | 0.105 | 0.125 | 0.175 |
| D | 25.8 | 26.0 | 26.2 |
| D1 | 23.8 | 24.0 | 24.2 |
| e1 | | 0.05 | |
| L1 | 0.3 | 0.5 | 0.7 |
| Y | | | 0.1 |
| T | 0.0 | | 10.0 |

## 6.0. TESTABILITY

**General Test Mode Description**

The test modes are decoded from the IRQ[7:5] inputs when TESTIN# is low. Test modes (Table 18) are latched by a positive assertion of PWROK.

**Table 18. Test Modes**

| Test Mode | IRQ7 | IRQ6 | IRQ5 | TESTIN# |
|---|---|---|---|---|
| NAND Tree | 0 | x | x | 0 |
| NAND Tree | x | x | 0 | 0 |
| Tri-state All Outputs | 1 | 0 | 1 | 0 |

**Tri-State Mode Description**

When in the tri-state test mode all outputs and bi-directional pins are tri-stated, including the NAND Tree final output IORDY.

**NAND Tree Test Mode Description**

Tri-states all outputs and bi-directional buffers except for IORDY and SMOUT0. Every output buffer except for IORDY and SMOUT0 is configured as an input in NAND Tree mode and included in the NAND chain. The first input of the NAND chain is DIOR#, and the NAND chain is routed counter-clockwise around the chip (e.g. DIOR#, DIOW#, SA7/DCS1#, ...). The last cell in the chain is SMOUT5 and IORDY is the final output. PCICLK, HCLK, RTCCLK, IORDY, PWROK and TESTIN# are the only input pins not included in the NAND chain. Note in the table above there are two possible ways to select NAND Tree test mode.

**NAND Tree Test Mode Operation**

To perform a NAND Tree test, all pins included in the NAND Tree should be driven high, except for the following pins, which use inverting Schmitt trigger inputs and should be driven low:

| Pin # | Pin Name |
|---|---|
| 77 | IRQ1 |
| 41 | IRQ8# |
| 78 | IRQ9 |
| 36 | ZEROWS# |
| 79 | IRQ7 |
| 80 | IRQ6 |
| 81 | IRQ5 |

| Pin # | Pin Name |
|---|---|
| 82 | IRQ4 |
| 83 | IRQ3 |
| 84 | IRQ10 |
| 85 | IRQ11 |
| 86 | IRQ12 |
| 93 | IRQ15 |
| 94 | IRQ14 |

Beginning with DIOR# and working counter-clockwise around the chip, each pin can be toggled and a resulting toggle observed on IORDY. Once a pin is toggled it must remain in the new state for the remainder of the NAND Tree test.

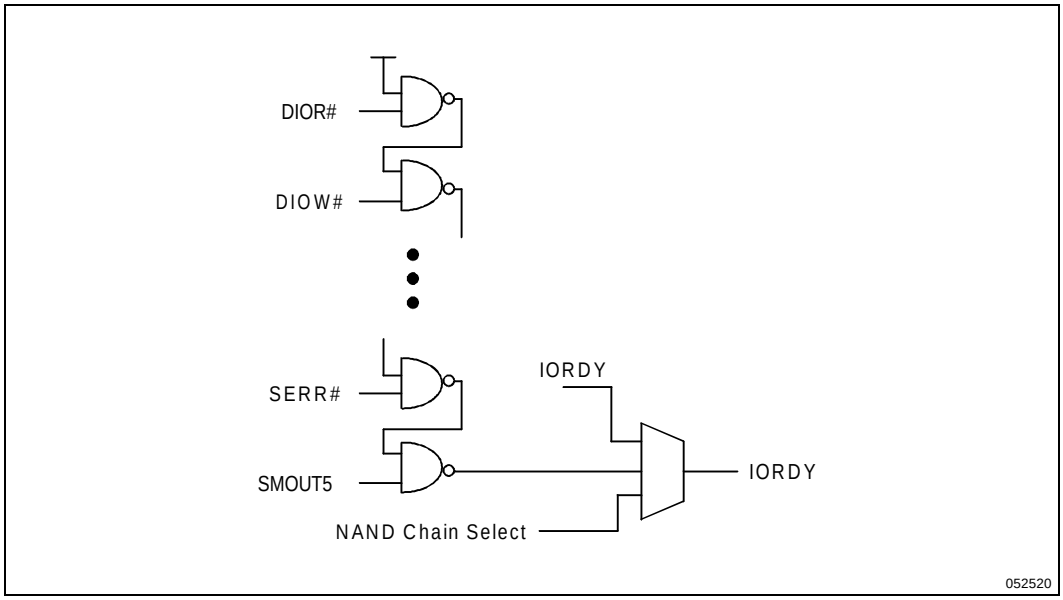



**Figure 20. NAND Tree Chain**

**NAND Tree Timing Requirements**

Allow 500 ns for the input signals to propagate to the NAND Tree outputs (input-to-output propagation delay specification).

**Table 19. NAND Tree**

| Pin # | Pin Name | Notes |
|---|---|---|
| 53 | TESTIN# | TESTIN# should be driven low for the duration of the NAND Tree test. |
| 40 | PWROK | A positive assertion of PWROK is required for selection of NAND Tree test mode. |
| 79 | IRQ7 | Test mode select signal. |
| 80 | IRQ6 | Test mode select signal. |
| 81 | IRQ5 | Test mode select signal. |
| 6 | DIOR# | First signal in the NAND Tree chain. |
| 7 | DIOW# | |

| Pin # | Pin Name | Notes |
|---|---|---|
| 8 | SA7/DCS1# | |
| 9 | SA6/DCS3# | |
| 10 | SA5 | |
| 11 | SA4 | |
| 12 | SA3 | |
| 13 | SA0/DA0 | |
| 14 | SA2/DA2 | |
| 15 | SA1/DA1 | |
| 16 | SA17/PCS# | |
| 17 | SA16/ RTCCS# | |
| 18 | SD0/DD0 | |
| 19 | SD1/DD1 | |
| 20 | SD2/DD2 | |

| Pin # | Pin Name | Notes |
|---|---|---|
| 21 | SD3/DD3 | |
| 22 | SD4/DD4 | |
| 23 | SD5/DD5 | |
| 24 | SD6/DD6 | |
| 25 | SD7/DD7 | |
| 28 | SA15/DD15/ SD15 | |
| 29 | SA14/DD14/ SD14 | |
| 30 | SA13/DD13/ SD13 | |
| 31 | SA12/DD12/ SD12 | |
| 32 | SA11/DD11/ SD11 | |
| 33 | SA10/DD10/ SD10 | |
| 34 | SA9/DD9/ SD9 | |
| 35 | SA8/DD8/ SD8 | |
| 36 | ZEROWS# | Inverted input signal. |
| 37 | IOCHRDY | |
| 41 | IRQ8# | Inverted input signal. |
| 42 | EXTSMI# | |
| 43 | COMRI# | |
| 46 | SRBTN# | |
| 47 | BATLOW# | |
| 48 | RSMRST# | |
| 49 | SUSSTAT# | |
| 51 | RTCCLKO | |
| 54 | CPURST# | |
| 55 | INIT | |
| 56 | FERR# | |
| 57 | IGNNE# | |
| 58 | INTR | |
| 59 | NMI | |
| 60 | SMI# | |
| 61 | STPCLK# | |

| Pin # | Pin Name | Notes |
|---|---|---|
| 62 | HCLKCO | |
| 64 | ALTA20 | |
| 65 | IOR# | |
| 66 | IOW# | |
| 67 | DREQ2 | |
| 68 | RSTDRV | |
| 69 | TC | |
| 70 | DACK2# | |
| 71 | MEMR# | |
| 72 | MEMW# | |
| 73 | PHOLD# | |
| 74 | PHLDA# | |
| 75 | SPKR | |
| 76 | SDIR | |
| 77 | IRQ1 | Inverted input signal. |
| 78 | IRQ9 | Inverted input signal. |
| 79 | IRQ7 | Inverted input signal. |
| 80 | IRQ6 | Inverted input signal. |
| 81 | IRQ5 | Inverted input signal. |
| 82 | IRQ4 | Inverted input signal. |
| 83 | IRQ3 | Inverted input signal. |
| 84 | IRQ10 | Inverted input signal. |
| 85 | IRQ11 | Inverted input signal. |
| 86 | IRQ12/M | Inverted input signal. |
| 92 | OSC | |
| 93 | IRQ15 | Inverted input signal. |
| 94 | IRQ14 | Inverted input signal. |
| 95 | SYSACT# | |
| 97 | SMOUT1 | |
| 98 | SMOUT2 | |
| 99 | SMOUT3 | |
| 100 | SMOUT4/ RTCALE | |
| 101 | PCICLKO | |
| 102 | CLKRUN# | |
| 103 | PCIRST# | |
| 104 | BIOSCS# | |

| Pin # | Pin Name | Notes |
|---|---|---|
| 105 | KBCCS# | |
| 106 | MDRQ2 | |
| 107 | MDRQ1 | |
| 108 | MDRQ0 | |
| 109 | MDAK2# | |
| 114 | MDAK1# | |
| 115 | MDAK0# | |
| 116 | REQA# | |
| 117 | REQB# | |
| 118 | GNTA# | |
| 119 | GNTB# | |
| 120 | MIRQ# | |
| 121 | PIRQA# | |
| 122 | PIRQB# | |
| 123 | SYSCLK | |
| 124 | IDSEL | |
| 125 | AD31 | |
| 126 | AD30 | |
| 127 | AD29 | |
| 128 | AD28 | |
| 129 | AD27 | |
| 130 | AD26 | |
| 135 | AD25 | |
| 136 | AD24 | |
| 137 | C/BE3# | |
| 138 | AD23 | |
| 139 | AD22 | |
| 140 | AD21 | |
| 141 | AD20 | |
| 142 | AD19 | |
| 143 | AD18 | |
| 144 | AD17 | |

| Pin # | Pin Name | Notes |
|---|---|---|
| 145 | AD16 | |
| 146 | C/BE2# | |
| 149 | FRAME# | |
| 150 | IRDY# | |
| 151 | TRDY# | |
| 152 | DEVSEL# | |
| 153 | STOP# | |
| 154 | PAR | |
| 155 | C/BE1# | |
| 156 | AD15 | |
| 157 | AD14 | |
| 158 | AD13 | |
| 159 | AD12 | |
| 160 | AD11 | |
| 163 | AD10 | |
| 164 | AD9 | |
| 165 | AD8 | |
| 166 | C/BE0# | |
| 167 | AD7 | |
| 168 | AD6 | |
| 169 | AD5 | |
| 170 | AD4 | |
| 171 | AD3 | |
| 172 | AD2 | |
| 173 | AD1 | |
| 174 | AD0 | |
| 3 | SERR# | |
| 4 | SMOUT5/ DOE# | Final signal of the NAND Tree chain. |
| 5 | IORDY | Output of the NAND Tree chain. |