Pentium® Processor Family Developer's Manual

1997

TABLE OF CONTENTS

Figures

Tables

1

Architectural Features of the Pentium® Processor Family

CHAPTER 1
ARCHITECTURAL FEATURES OF THE PENTIUM® PROCESSOR FAMILY

This volume covers the desktop Pentium® processor family. This includes both the Pentium processor (75/90/100/120/133/150/166/200)—supporting maximum operating frequencies of 75, 90, 100, 120, 133, 150, 166 and 200 MHz—and the newest member of the Pentium processor family, the Pentium processor with MMX technology—supporting Intel MMX technology and maximum operating frequencies of 166 and 200 MHz. This volume does not cover the 5V Pentium processors (60, 66). Mobile Pentium processor specifications and information are covered in separate documents (refer to Chapter 18 for odering information).

The general terms “Pentium processor” and “Pentium processor family” are used throughout this volume to refer to both the Pentium processor (75/90/100/120/133/150/166/200) and the Pentium processor with MMX technology together. The names “Pentium processor (75/90/100/120/133/150/166/200)” and “Pentium processor with MMX technology” are used to distinguish between the two processors where specific differences exist.

1.1. PROCESSOR FEATURES OVERVIEW

The Pentium processor supports the features of previous Intel Architecture processors and provides significant enhancements including the following:

- Superscalar Architecture
- Dynamic Branch Prediction
- Pipelined Floating-Point Unit
- Improved Instruction Execution Time
- Separate Code and Data Caches[1].
- Writeback MESI Protocol in the Data Cache
- 64-Bit Data Bus
- Bus Cycle Pipelining
- Address Parity
- Internal Parity Checking
- Functional Redundancy Checking[2] and Lock Step operation[2]
- Execution Tracing
- Performance Monitoring
- IEEE 1149.1 Boundary Scan
- System Management Mode

- Virtual Mode Extensions
- Upgradable with a Pentium OverDrive processor[2]
- Dual processing support
- Advanced SL Power Management Features
- Fractional Bus Operation
- On-Chip Local APIC Device

In addition, the Pentium processor with MMX technology offers the following enhancements over the Pentium processor (75/90/100/120/133/150/166/200):[1]

- Support for Intel MMX technology
- Dual power supplies—separate $V_{CC}2$ (core) and $V_{CC}3$ (I/O) voltage inputs
- Separate 16 Kbyte 4-way set-associative code and data caches, each with improved fully associative TLBs
- Pool of 4 write buffers used by both pipes
- Enhanced branch prediction algorithm
- New Fetch pipeline stage between Prefetch and Instruction Decode

The following features are supported by the Pentium processor (75/90/100/120/133/150/166/200), but not supported by the Pentium processor with MMX technology:

- Functional Redundancy Checking and Lock Step operation
- Support for the Intel 82498/82493 and 82497/82492 cache chipset products
- Upgradability with a Pentium OverDrive processor
- Split line accesses to the code cache

1.2. COMPONENT INTRODUCTION

The application instruction set of the Pentium processor family includes the complete instruction set of existing Intel Architecture processors to ensure backward compatibility, with extensions to accommodate the additional functionality of the Pentium processor. All application software written for the Intel386™ and Intel486™ microprocessors will run on the Pentium processor without modification. The on-chip memory management unit (MMU) is completely compatible with the Intel386 and Intel486 CPUs.

Footnotes

[1] The Code and Data caches are each 8Kbytes, a two-way set-associative on the Pentium processor (75/90/100/120/ 133/150/166/200).

[2] This feature is not supported on the Pentium processor with MMX technology.

The Pentium processor with MMX technology adds 57 new instructions and four new data types to accelerate the performance of multimedia and communications software. MMX technology is based on the SIMD technique—Single Instruction, Multiple data—which enables increased performance on a wide variety of multimedia and communications applications. To take advantage of the MMX instructions, software modifications need to be made. However, if the MMX instructions are not utilized, there are no hardware or software modifications needed.

The two instruction pipelines and the floating-point unit on the Pentium processor are capable of independent operation. Each pipeline issues frequently used instructions in a single clock. Together, the dual pipes can issue two integer instructions in one clock, or one floating-point instruction (under certain circumstances, 2 floating-point instructions) in one clock.

The Pentium processor with MMX technology adds the Fetch pipeline stage between the Prefetch and Instruction decode stages, which increases the performance capability of the processor. The Pentium processor with MMX technology also doubles the number of write buffers available to be used by the dual pipelines.

Branch prediction is implemented in the Pentium processor. To support this, the Pentium processor implements two prefetch buffers, one to prefetch code in a linear fashion, and one that prefetches code according to the Branch Target Buffer (BTB) so the needed code is almost always prefetched before it is needed for execution. The branch prediction algorithm has been enhanced on the Pentium processor with MMX technology for increased accuracy.

The Pentium processor includes separate code and data caches integrated on chip to meet its performance goals. Each cache on the Pentium processor with MMX technology is 16 Kbytes in size, and is 4-way set associative. The caches on the Pentium processor (75/90/100/120/133/150/166/200) are each 8 Kbytes in size and 2-way set-associative. Each cache has a dedicated Translation Lookaside Buffer (TLB) to translate linear addresses to physical addresses. The Pentium processor data cache is configurable to be writeback or writethrough on a line-by-line basis and follows the MESI protocol. The data cache tags are triple ported to support two data transfers and an inquire cycle in the same clock. The code cache is an inherently write protected cache. The code cache tags of the Pentium processor (75/90/100/120/133/150/166/200) are also triple ported to support snooping and split-line accesses. The Pentium processor with MMX technology does not support split line accesses to the code cache. As such, its code cache tags are dual ported. Individual pages can be configured as cacheable or non-cacheable by software or hardware. The caches can be enabled or disabled by software or hardware.

The Pentium processor has a 64-bit data bus. Burst read and burst writeback cycles are supported by the Pentium processor. In addition, bus cycle pipelining has been added to allow two bus cycles to be in progress simultaneously. The Pentium processor Memory Management Unit contains optional extensions to the architecture which allow 4 MB page sizes.

The Pentium processor has added significant data integrity and error detection capability. Data parity checking is still supported on a byte-by-byte basis. Address parity checking, and internal parity checking features have been added along with a new exception, the machine check exception.

The Pentium processor (75/90/100/120/133/150/166/200) has implemented functional redundancy checking to provide maximum error detection of the processor and the interface to the processor. When functional redundancy checking is used, a second processor, the "checker" is used to execute in lock step with the "master" processor. The checker samples the master's outputs and compares those values with the values it computes internally, and asserts an error signal if a mismatch occurs. The Pentium processor with MMX technology does not support functional redundancy checking.

As more and more functions are integrated on chip, the complexity of board level testing is increased. To address this, the Pentium processor has increased test and debug capability by implementing IEEE Boundary Scan (Standard 1149.1).

System management mode has been implemented along with some extensions to the SMM architecture. Enhancements to the Virtual 8086 mode have been made to increase performance by reducing the number of times it is necessary to trap to a Virtual 8086 monitor.

Figure 1-1 presents a block diagram overview of the Pentium processor with MMX technology including the two instruction pipelines, the "u" pipe and the "v" pipe. The u-pipe can execute all integer and floating-point instructions. The v-pipe can execute simple integer instructions and the FXCH floating-point instruction.

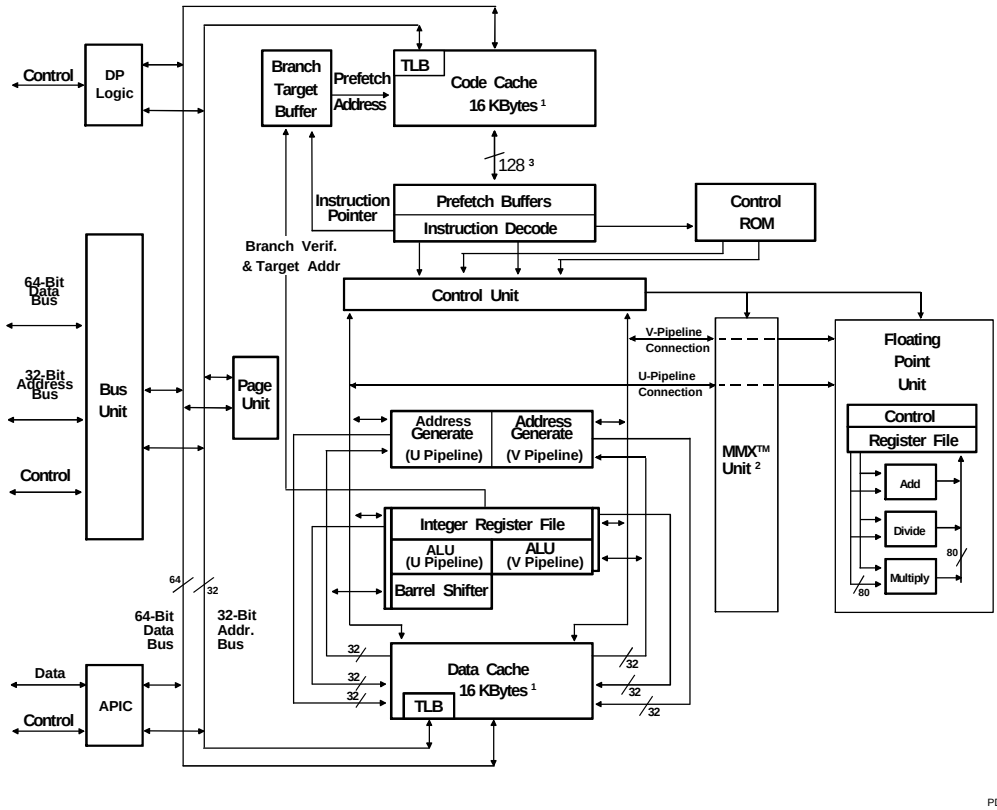
The separate code and data caches are shown. The data cache has two ports, one for each of the two pipes (the tags are triple ported to allow simultaneous inquire cycles). The data cache has a dedicated TLB to translate linear addresses to the physical addresses used by the data cache.

The code cache, branch target buffer and prefetch buffers are responsible for getting raw instructions into the execution units of the Pentium processor. Instructions are fetched from the code cache or from the external bus. Branch addresses are remembered by the branch target buffer. The code cache TLB translates linear addresses to physical addresses used by the code cache.

The decode unit contains two parallel decoders which decode and issue up to the next two sequential instructions into the execution pipeline. The control ROM contains the microcode which controls the sequence of operations performed by the processor. The control unit has direct control over both pipelines.

The Pentium processor contains a pipelined floating-point unit that provides a significant floating-point performance advantage over previous generations of Intel Architecture-based processors.

The Pentium processor includes features to support multi-processor systems, namely an on-chip Advanced Programmable Interrupt Controller (APIC). This APIC implementation supports multiprocessor interrupt management (with symmetric interrupt distribution across all processors), multiple I/O subsystem support, 8259A compatibility, and inter-processor interrupt support.



NOTES:

1. The Code and Data caches are each 8 Kbytes in size on the Pentium® processor (75/90/100/120/133/150/166/200).
2. The MMX Unit is present only on the Pentium processor with MMX™ technology
3. The internal instruction bus is 256 bits wide on the Pentium processor (75/90/100/120/133/150/166/200)

Figure 1-1. Pentium® Processor Block Diagram

The dual processor configuration allows two Pentium processors to share a single L2 cache for a low-cost symmetric multi-processor system. The two processors appear to the system as a single Pentium processor. Multiprocessor operating systems properly schedule computing tasks between the two processors. This scheduling of tasks is transparent to software applications and the end-user. Logic built into the processors support a “glueless” interface for easy system design. Through a private bus, the two Pentium processors arbitrate for the external bus and maintain cache coherency. The Pentium processor can also be used in a conventional multi-processor system in which one L2 cache is dedicated to each processor.

In this document, in order to distinguish between two Pentium processors in dual processing mode, one CPU will be designated as the Primary processor with the other being the Dual processor. Note that this is a different concept than that of “master” and “checker” processors described in the discussion on functional redundancy.

Dual processing is supported in a system only if both processors are operating at identical core and bus frequencies and are the same type of processor (i.e., both Pentium processor (75/90/100/120/133/150/166/200) or both Pentium processor with MMX technology). Within these restrictions, two processors of different steppings may operate together in a system. See the “Component Operation” chapter for more details about Dual processing.

The Pentium processor is produced on Intel’s advanced silicon technology. The Pentium processor also includes SL enhanced power management features. When the clock to the Pentium processor is stopped, power dissipation is virtually eliminated. The low V_{CC} operating voltages and SL enhanced power management features make the Pentium processor a good choice for energy-efficient desktop designs.

Supporting an upgrade socket (Socket 7) in the system will provide end-user upgradability by the addition of a future Pentium OverDrive processor. Typical applications will realize a 40% to 70% performance increase by addition of a future Pentium OverDrive processor.

The Pentium processor supports fractional bus operation. This allows the internal processor core to operate at high frequencies, while communicating with the external bus at lower frequencies. Table 4-3 lists the bus-to-core frequency ratios supported on the Pentium processor.

2

Component Operation

CHAPTER 2
COMPONENT OPERATION

The Pentium processor has an optimized superscalar micro-architecture capable of executing two instructions in a single clock. A 64-bit external bus, separate 8-Kbyte data and instruction caches for Pentium processor (75/90/100/120/133/150/166/200), separate 16-Kbyte data and instruction caches for Pentium processor with MMX technology, write buffers, branch prediction (with an enhanced branch prediction algorithm for the Pentium processor with MMX technology), and a pipelined floating-point unit combine to sustain the high execution rate. These architectural features and their operation are discussed in this chapter.

2.1. PIPELINE AND INSTRUCTION FLOW

The integer instructions traverse a five stage pipeline in the Pentium processor (75/90/100/120/133/150/166/200), while the Pentium processor with MMX technology has an additional pipeline stage. The pipeline stages are as follows:

PF Prefetch

F Fetch (Pentium processor with MMX technology only)

D1 Instruction Decode

D2 Address Generate

EX Execute - ALU and Cache Access

WB Writeback

The Pentium processor is a superscalar machine, built around two general purpose integer pipelines and a pipelined floating-point unit capable of executing two instructions in parallel. Both pipelines operate in parallel allowing integer instructions to execute in a single clock in each pipeline. Figure 2-1 depicts instruction flow in the Pentium processor.

The pipelines in the Pentium processor are called the “u” and “v” pipes and the process of issuing two instructions in parallel is termed “pairing.” The u-pipe can execute any instruction in the Intel architecture, while the v-pipe can execute “simple” instructions as defined in the “Instruction Pairing Rules” section of this chapter. When instructions are paired, the instruction issued to the v-pipe is always the next sequential instruction after the one issued to the u-pipe.

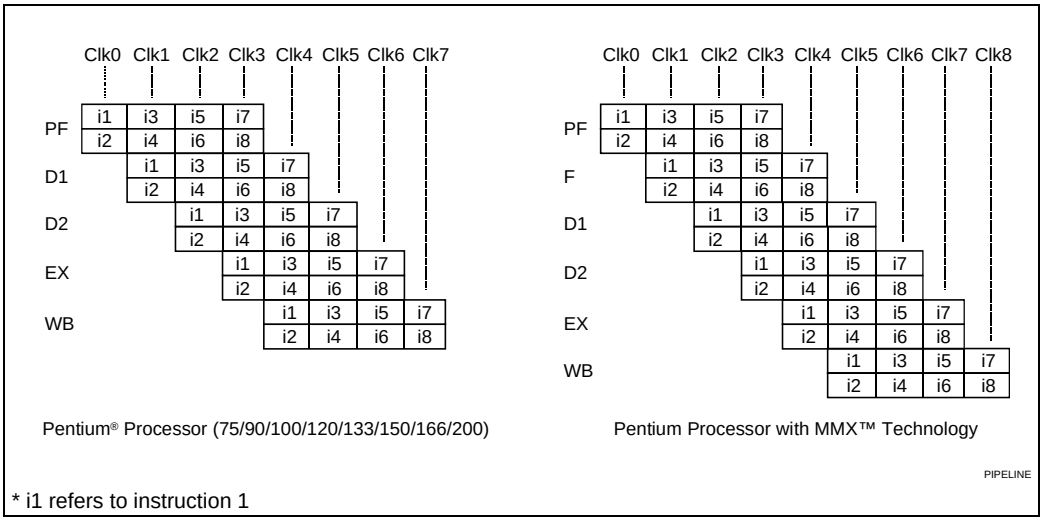


Figure 2-1. Pentium® Processor Pipeline Execution

2.1.1. Pentium® Processor Integer Pipeline Description

The Pentium processor pipeline has been optimized to achieve higher throughput compared to previous generations of Intel Architecture processors.

The first stage of the pipeline is the Prefetch (PF) stage in which instructions are prefetched from the on-chip instruction cache or memory. Because the Pentium processor has separate caches for instructions and data, prefetches do not conflict with data references for access to the cache. If the requested line is not in the code cache, a memory reference is made. In the PF stage of the Pentium processor (75/90/100/120/133/150/166/200), two independent pairs of line-size (32-byte) prefetch buffers operate in conjunction with the branch target buffer. This allows one prefetch buffer to prefetch instructions sequentially, while the other prefetches according to the branch target buffer predictions. The prefetch buffers alternate their prefetch paths. In the Pentium processor with MMX technology, four 16-byte prefetch buffers operate in conjunction with the BTB to prefetch up to four independent instruction streams. See the section titled “Instruction Prefetch” in this chapter for further details on the Pentium processor family prefetch buffers.

In the Pentium processor with MMX technology only, the next pipeline stage is Fetch (F), and it is used for instruction length decode. It replaces the D1 instruction-length decoder and eliminates the need for end-bits to determine instruction length. Also, any prefixes are decoded in the F stage. The Fetch stage is not supported by the Pentium processor (75/90/100/120/133/150/166/200) pipeline.

The Pentium processor with MMX technology also features an instruction FIFO between the F and D1 stages. This FIFO is transparent; it does not add additional latency when it is empty. During every clock cycle, two instructions can be pushed into the instruction FIFO (depending

on availability of the code bytes, and on other factors such as prefixes). Instruction pairs are pulled out of the FIFO into the D1 stage. Since the average rate of instruction execution is less than two per clock, the FIFO is normally full. As long as the FIFO is full, it can buffer any stalls that may occur during instruction fetch and parsing. If such a stall occurs, the FIFO prevents the stall from causing a stall in the execution stage of the pipe. If the FIFO is empty, then an execution stall may result from the pipeline being "starved" for instructions to execute. Stalls at the FIFO entrance may be caused by long instuctions or prefixes, or "extremely misaligned targets" (i.e., Branch targets that reside at the last bytes of 16-aligned bytes).

The pipeline stage after the PF stage in the Pentium processor (75/90/100/120/133/150/166/200) is Decode1 (D1) in which two parallel decoders attempt to decode and issue the next two sequential instructions. The decoders determine whether one or two instructions can be issued contingent upon the instruction pairing rules described in the section titled "Instruction Pairing Rules." The Pentium processor (75/90/100/120/133/150/166/200) requires an extra D1 clock to decode instruction prefixes. Prefixes are issued to the u-pipe at the rate of one per clock without pairing. After all prefixes have been issued, the base instruction will then be issued and paired according to the pairing rules. The one exception to this is that the Pentium processor (75/90/100/120/133/150/166/200) will decode near conditional jumps (long displacement) in the second opcode map (0Fh prefix) in a single clock in either pipeline. The Pentium processor with MMX technology handles 0Fh as part of the opcode and not as a prefix. Consequently, 0Fh does not take one extra clock to get into the FIFO. Note, in the Pentium processor with MMX technology, MMX instructions can be paired as discussed in the "MMX Instruction Pairing Guidelines" section later in this chapter.

The D1 stage is followed by Decode2 (D2) in which addresses of memory resident operands are calculated. In the Intel486™ processor, instructions containing both a displacement and an immediate, or instructions containing a base and index addressing mode require an additional D2 clock to decode. The Pentium processor removes both of these restrictions and is able to issue instructions in these categories in a single clock.

The Pentium processor uses the Execute (EX) stage of the pipeline for both ALU operations and for data cache access; therefore those instructions specifying both an ALU operation and a data cache access will require more than one clock in this stage. In EX all u-pipe instructions and all v-pipe instructions except conditional branches are verified for correct branch prediction. Microcode is designed to utilize both pipelines and thus those instructions requiring microcode execute faster.

The final stage is Writeback (WB) where instructions are enabled to modify processor state and complete execution. In this stage, v-pipe conditional branches are verified for correct branch prediction.

During their progression through the pipeline, instructions may be stalled due to certain conditions. Both the u-pipe and v-pipe instructions enter and leave the D1 and D2 stages in unison. When an instruction in one pipe is stalled, then the instruction in the other pipe is also stalled at the same pipeline stage. Thus both the u-pipe and the v-pipe instructions enter the EX stage in unison. Once in EX if the u-pipe instruction is stalled, then the v-pipe instruction (if any) is also stalled. If the v-pipe instruction is stalled then the instruction paired with it in the u-pipe is not allowed to advance. No successive instructions are allowed to enter the EX stage of either pipeline until the instructions in both pipelines have advanced to WB.

2.1.1.1. INSTRUCTION PREFETCH

In the Pentium processor (75/90/100/120/133/150/166/200) PF stage, two independent pairs of line-size (32-byte) prefetch buffers operate in conjunction with the branch target buffer. Only one prefetch buffer actively requests prefetches at any given time. Prefetches are requested sequentially until a branch instruction is fetched. When a branch instruction is fetched, the branch target buffer (BTB) predicts whether the branch will be taken or not. If the branch is predicted not taken, prefetch requests continue linearly. On a predicted taken branch the other prefetch buffer is enabled and begins to prefetch as though the branch was taken. If a branch is discovered mis-predicted, the instruction pipelines are flushed and prefetching activity starts over.

The Pentium processor with MMX technology's prefetch stage has four 16-byte buffers which can prefetch up to four independent instruction streams, based on predictions made by the BTB. In this case, the Branch Target Buffer predicts whether the branch will be taken or not in the PF stage. The Pentium processor with MMX technology features an enhanced two-stage Branch prediction algorithm, compared to the Pentium processor (75/90/100/120/133/150/166/200).

For more information on branch prediction, see section 2.2.

2.1.2. Integer Instruction Pairing Rules

The Pentium processor can issue one or two instructions every clock. In order to issue two instructions simultaneously they must satisfy the following conditions:

- Both instructions in the pair must be "simple" as defined below
- There must be no read-after-write or write-after-write register dependencies between them
- Neither instruction may contain both a displacement and an immediate
- Instructions with prefixes can only occur in the u-pipe (except for JCC instructions with a 0Fh prefix on the Pentium processor (75/90/100/120/133/150/166/200) and instructions with a 0Fh, 66h or 67h prefix on the Pentium processor with MMX technology).
- Instruction prefixes are treated as separate 1-byte instructions (except for all 0F prefixed instructions in the Pentium processor with MMX technology)

Simple instructions are entirely hardwired; they do not require any microcode control and, in general, execute in one clock. The exceptions are the ALU mem,reg and ALU reg,mem instructions which are three and two clock operations respectively. Sequencing hardware is used to allow them to function as simple instructions. The following integer instructions are considered simple and may be paired:

1. mov reg, reg/mem/imm
2. mov mem, reg/imm
3. alu reg, reg/mem/imm
4. alu mem, reg/imm

5. inc reg/mem
6. dec reg/mem
7. push reg/mem
8. pop reg
9. lea reg,mem
10. jmp/call/jcc near
11. nop
12. test reg, reg/mem
13. test acc, imm

In addition, conditional and unconditional branches may be paired only if they occur as the second instruction in the pair. They may not be paired with the next sequential instruction. Also, SHIFT/ROT by 1 and SHIFT by imm may pair as the first instruction in a pair.

The register dependencies that prohibit instruction pairing include implicit dependencies via registers or flags not explicitly encoded in the instruction. For example, an ALU instruction in the u-pipe (which sets the flags) may not be paired with an ADC or an SBB instruction in the v-pipe. There are two exceptions to this rule. The first is the commonly occurring sequence of compare and branch which may be paired. The second exception is pairs of pushes or pops. Although these instructions have an implicit dependency on the stack pointer, special hardware is included to allow these common operations to proceed in parallel.

Although in general two paired instructions may proceed in parallel independently, there is an exception for paired "read-modify-write" instructions. Read-modify-write instructions are ALU operations with an operand in memory. When two of these instructions are paired there is a sequencing delay of two clocks in addition to the three clocks required to execute the individual instructions.

Although instructions may execute in parallel their behavior as seen by the programmer is exactly the same as if they were executed sequentially.

Information regarding pairing of FPU and MMX instructions is discussed in the "Floating-Point Unit" and "MMX™ Unit" sections of this chapter. For additional details on code optimization, please refer to *Optimizing for Intel's 32-Bit Processors*, Order # 241799.

2.2. BRANCH PREDICTION

The Pentium processor uses a Branch Target Buffer (BTB) to predict the outcome of branch instructions which minimizes pipeline stalls due to prefetch delays.

The Pentium processor (75/90/100/120/133/150/166/200) accesses the BTB with the address of the instruction in the D1 stage. It contains a Branch prediction state machine with four states: (1) strongly not taken, (2) weakly not taken, (3) weakly taken, and (4) strongly taken. In the event of a correct prediction, a branch will execute without pipeline stalls or flushes. Branches which miss the BTB are assumed to be not taken. Conditional and unconditional near branches and near calls execute in 1 clock and may be executed in parallel with other integer

instructions. A mispredicted branch (whether a BTB hit or miss) or a correctly predicted branch with the wrong target address will cause the pipelines to be flushed and the correct target to be fetched. Incorrectly predicted unconditional branches will incur an additional three clock delay, incorrectly predicted conditional branches in the u-pipe will incur an additional three clock delay, and incorrectly predicted conditional branches in the v-pipe will incur an additional four clock delay.

The benefits of branch prediction are illustrated in the following example. Consider the following loop from a benchmark program for computing prime numbers:

```
for(k=i+prime;k<=SIZE;k+=prime)
     flags[k]=FALSE;
```

A popular compiler generates the following assembly code:

(prime is allocated to `ecx`, `k` is allocated to `edx`, and `al` contains the value `FALSE`)

```
inner_loop:
     mov byte ptr flags[edx],al
     add edx,ecx
     cmp edx, SIZE
     jle inner_loop
```

Each iteration of this loop will execute in 6 clocks on the Intel486 CPU. On the Pentium processor, the `mov` is paired with the `add`; the `cmp` with the `jle`. With branch prediction, each loop iteration executes in 2 clocks.

NOTE

The dynamic branch prediction algorithm speculatively runs code fetch cycles to addresses corresponding to instructions executed some time in the past. Such code fetch cycles are run based on past execution history, regardless of whether the instructions retrieved are relevant to the currently executing instruction sequence.

One effect of the branch prediction mechanism is that the Pentium processor may run code fetch bus cycles to retrieve instructions which are never executed. Although the opcodes retrieved are discarded, the system must complete the code fetch bus cycle by returning BRDY#. It is particularly important that the system return BRDY# for all code fetch cycles, regardless of the address.

It should also be noted that upon entering SMM, the branch target buffer (BTB) is not flushed and thus it is possible to get a speculative prefetch to an address outside of SMRAM address space due to branch predictions based on code executed prior to entering SMM. If this occurs, the system must still return BRDY# for each code fetch cycle.

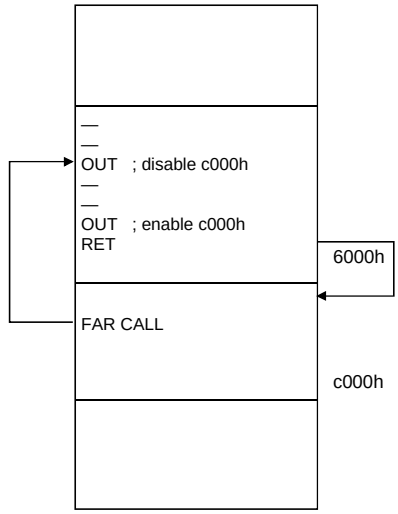
Furthermore, it is possible that the Pentium processor may run speculative code fetch cycles to addresses beyond the end of the current code segment (approximately 100 bytes past end of last executed instruction). Although the Pentium processor may prefetch beyond the CS limit, it will not attempt to execute beyond the CS limit. Instead, it will raise a GP fault. Thus,

segmentation cannot be used to prevent speculative code fetches to inaccessible areas of memory. On the other hand, the Pentium processor never runs code fetch cycles to inaccessible pages (i.e., not present pages or pages with incorrect access rights), so the paging mechanism guards against both the fetch and execution of instructions in inaccessible pages.

For memory reads and writes, both segmentation and paging prevent the generation of bus cycles to inaccessible regions of memory. If paging is not used, branch prediction can be disabled by setting TR12.NBP (bit 0)* and flushing the BTB by loading CR3 before disabling any areas of memory. Branch prediction can be re-enabled after re-enabling memory.

The following is an example of a situation that may occur:

1. Code passes control to segment at address c000h.
2. Code transfers control to code at different address (6000h) by using FAR CALL instruction.
3. This portion of the code does an I/O write to a port that disables memory at address c000h.
4. At the end of this segment, an I/O write is performed to re-enable memory at address c000h.
5. Following the OUT instruction, there is a RET instruction to c000h segment.



The branch prediction mechanism of the Pentium processor, however, predicts that the RET instruction is going to transfer control to the segment at address c000h and performs a prefetch from that address prior to the OUT instruction that re-enables that memory address. The result is that no BRDY is returned for that prefetch cycle and the system hangs.

In this case, branch prediction should be disabled (by setting TR12.NBP* and flushing the BTB by loading CR3) prior to disabling memory at address c000h and re-enabled after the RET instruction by clearing TR12.NBP* as indicated above.

* Please refer to Chapter 16 of this volume.

In the Pentium processor with MMX technology, the Branch prediction algorithm changes from the Pentium processor (75/90/100/120/133/150/166/200) in the following ways:

- BTB Lookup is done when the branch is in the PF stage.
- The BTB Lookup tag is the Prefetch address.
- A Lookup in the BTB performs a search spanning sixteen consecutive bytes.
- BTB can contain four branch instructions for each line of 16 bytes.
- BTB is constructed from four independent Banks. Each Bank contains 64 entries and is 4-way associative.
- Enchanced two-stage Branch prediction algorithm.

2.3. FLOATING-POINT UNIT

The floating-point unit (FPU) of the Pentium processor is integrated with the integer unit on the same chip. It is heavily pipelined. The FPU is designed to be able to accept one floating-point operation every clock. It can receive up to two floating-point instructions every clock, one of which must be an exchange instruction.

For information on code optimization, please refer to *Optimizing for Intel's 32-Bit Processors*, Order Number 241799.

2.3.1. Floating-Point Pipeline Stages

The Pentium processor FPU has 8 pipeline stages, the first five of which it shares with the integer unit. Integer instructions pass through only the first 5 stages. Integer instructions use the fifth (X1) stage as a WB (write-back) stage. The 8 FP pipeline stages, and the activities that are performed in them are summarized below:

PF Prefetch;

F Fetch (applicable to the Pentium processor with MMX technology only);

D1 Instruction Decode;

D2 Address generation;

EX Memory and register read; conversion of FP data to external memory format and memory write;

X1 Floating-Point Execute stage one; conversion of external memory format to internal FP data format and write operand to FP register file; bypass 1 (bypass 1 described in the "Bypasses" section).

X2 Floating-Point Execute stage two;

WF Perform rounding and write floating-point result to register file; bypass 2 (bypass 2 described in the "Bypasses" section).

ER Error Reporting/Update Status Word.

2.3.2. Instruction Issue

Described below are the rules of how floating-point (FP) instructions get issued on the Pentium processor:

1. FP instructions do not get paired with integer instructions. However, a limited pairing of two FP instructions can be performed.
2. When a pair of FP instructions is issued to the FPU, only the FXCH instruction can be the second instruction of the pair. The first instruction of the pair must be one of a set F where F = [FLD single/double, FLD ST(i), all forms of FADD, FSUB, FMUL, FDIV, FCOM, FUCOM, FTST, FABS, FCHS].

3. FP instructions other than the FXCH instruction and other than instructions belonging to set F (defined in rule 2) always get issued singly to the FPU.
4. FP instructions that are not directly followed by an FP exchange instruction are issued singly to the FPU.

The Pentium processor stack architecture instruction set requires that all instructions have one source operand on the top of the stack. Since most instructions also have their destination as the top of the stack, most instructions see a “top of stack bottleneck.” New source operands must be brought to the top of the stack before we can issue an arithmetic instruction on them. This calls for extra usage of the exchange instruction, which allows the programmer to bring an available operand to the top of the stack. The Pentium processor FPU uses pointers to access its registers to allow fast execution of exchanges and the execution of exchanges in parallel with other floating-point instructions. An FP exchange that is paired with other FP instructions takes 0 clocks for its execution. Since such exchanges can be executed in parallel on the Pentium processor, it is recommended that one use them when necessary to overcome the stack bottleneck.

Note that when exchanges are paired with other floating-point instructions, they should not be followed immediately by integer instructions. The Pentium processor stalls such integer instructions for a clock if the FP pair is declared safe, or for 4 clocks if the FP pair is unsafe.

Also note that the FP exchange must always follow another FP instruction to get paired. The pairing mechanism does not allow the FP exchange to be the first instruction of a pair that is issued in parallel. If an FP exchange is not paired, it takes 1 clock for its execution.

2.3.3. Safe Instruction Recognition

The Pentium processor FPU performs Safe Instruction Recognition or SIR in the X1 stage of the pipeline. SIR is an early inspection of operands and opcodes to determine whether the instruction is guaranteed not to generate an arithmetic overflow, underflow, or unmasked inexact exception. An instruction is declared safe if it cannot raise any other floating-point exception, and if it does not need microcode assist for delivery of special results. If an instruction is declared safe, the next FP instruction is allowed to complete its E stage operation. If an instruction is declared unsafe, the next FP instruction stalls in the E stage until the current one completes (ER stage) with no exception. This means a 4 clock stall, which is incurred even if the numeric instruction that was declared unsafe does not eventually raise a floating-point exception.

For normal data, the rules used on the Pentium processor for declaring an instruction safe are as follows.

On the Pentium processor (75/90/100/120/133/150/166/200), if FOP = FADD/FSUB/ FMUL/FDIV, the instruction is safe from arithmetic overflow, underflow, and unmasked inexact exceptions if:

1. Both operands have unbiased exponent ≤ 1FFEh AND
2. Both operands have unbiased exponent ≥ -1FFEh AND
3. The inexact exception is masked.

Similarly, on the Pentium processor with MMX technology, if FOP = FADD/FSUB/ FMUL/FDIV, the instruction is safe from arithmetic overflow, underflow, and unmasked inexact exceptions if:

1. Both operands have unbiased exponent ≤ 1000h AND
2. Both operands have unbiased exponent ≥ -0FFFh AND
3. The inexact exception is masked

Note that arithmetic overflow of the double precision format occurs when the unbiased exponent of the result is ≥ 400h, and underflow occurs when the exponent is ≤-3FFh. Hence, the SIR algorithm on the Pentium processor allows improved throughput on a much greater range of numbers than that spanned by the double precision format.

2.3.4. FPU Bypasses

The following section describes the floating-point register file bypasses that exist on the Pentium processor. The register file has two write ports and two read ports. The read ports are used to read data out of the register file in the E stage. One write port is used to write data into the register file in the X1 stage, and the other in the WF stage. A bypass allows data that is about to be written into the register file to be available as an operand that is to be read from the register file by any succeeding floating-point instruction. A bypass is specified by a pair of ports (a write port and a read port) that get circumvented. Using the bypass, data is made available even before actually writing it to the register file.

The following procedures are implemented:

1. Bypass the X1 stage register file write port and the E stage register file read port.
2. Bypass the WF stage register file write port and the E stage register file read port.

With bypass 1, the result of a floating-point load (that writes to the register file in the X1 stage) can bypass the X1 stage write and be sent directly to the operand fetch stage or E stage of the next instruction.

With bypass 2, the result of any arithmetic operation can bypass the WF stage write to the register file, and be sent directly to the desired execution unit as an operand for the next instruction.

Note that the FST instruction reads the register file with a different timing requirement, so that for the FST instruction, which attempts to read an operand in the E stage:

1. There is no bypassing the X1 stage write port and the E stage read port, i.e. no added bypass for FLD followed by FST. Thus FLD (double) followed by FST (double) takes 4 clocks (2 for FLD, and 2 for FST).
2. There is no bypassing the WF stage write port and the E stage read port. The E stage read for the FST happens only in the clock following the WF write for any preceding arithmetic operation.

Furthermore, there is no memory bypass for an FST followed by an FLD from the same memory location.

2.3.5. Branching Upon Numeric Condition Codes

Branching upon numeric condition codes is accomplished by transferring the floating-point SW to the integer FLAGS register and branching on it. The “test numeric condition codes and branch” construct looks like:

FP instruction1; instruction whose effects on the status word are to be examined;

“numeric_test_and_branch_construct”:

FSTSW AX; move the status word to the ax register.

SAHF; transfer the value in ah to the lower half of the eflags register.

JC xyz ; jump upon the condition codes in the eflags register.

Note that all FP instructions update the status word only in the ER stage. Hence there is a built-in status word interlock between FP instruction1 and the FSTSW AX instruction. The above piece of code takes 9 clocks before execution of code begins at the target of the jump. These 9 clocks are counted as:

FP instruction1 :	X1, X2, WF, ER (4 E stage stalls for the FSTSWAX);
FSTSW AX :	2 E clocks;
SAHF :	2 E clocks;
JC xyz :	1 clock if no mispredict on branch.

Note that if there is a branch mispredict, there will be a minimum of 3 clocks added to the clock count of 9.

It is recommended that such attempts to branch upon numeric condition codes be preceded by integer instructions, i.e. one should insert integer instructions in between FP instruction1 and the FSTSW AX instruction which is the first instruction of the “numeric test and branch” construct. This allows the elimination of up to 4 clocks (the 4 E-stage stalls on FSTSW AX) from the cost attributed to this construct, so that numeric branching can be accomplished in 5 clocks.

2.4. MMX™ UNIT

The Intel MMX technology, supported on the Pentium processor with MMX technology, comprises a set of extensions to the Intel Architecture that are designed to greatly enhance the performance of advanced media and communications applications. These extensions (which include new registers, data types, and instructions) are combined with a single-instruction, multiple-data (SIMD) execution model to accelerate the performance of applications such as motion video, combined graphics with video, image processing, audio synthesis, speech synthesis and compression, telephony, video conferencing, and 2D and 3D graphics, which typically use compute-intensive algorithms to perform repetitive operations on large arrays of simple, native data elements.

The MMX technology defines a simple and flexible software model, with no new mode or operating-system visible state. All existing software will continue to run correctly, without modification, on Intel Architecture processors that incorporate the MMX technology, even in the presence of existing and new applications that incorporate this technology.

The following sections of this chapter describe the basic programming environment for the technology, the MMX register set, data types and instruction set. Detailed descriptions of the MMX instructions are provided in Chapter 3 of the *Intel Architecture Software Developer's Manual*, Volume 2. The manner in which the MMX technology extensions fit into the Intel Architecture system programming model is described in Chapter 10 in the *Intel Architecture Software Developer's Manual*, Volume 3.

2.4.1. Overview of the MMX™ Programming Environment

MMX technology provides the following new extensions to the Intel Architecture programming environment:

- Eight MMX registers (MM0 through MM7)
- Four MMX data types (packed bytes, packed words, packed doublewords and quadword)
- The MMX instruction set

2.4.1.1. MMX™ REGISTERS

The MMX register set consists of eight 64-bit registers (see Figure 2-2). The MMX instructions access the MMX registers directly using the register names MM0 through MM7. These registers can only be used to perform calculations on MMX data types; they cannot be used to address memory. Addressing of MMX instruction operands in memory is handled by using the standard Intel Architecture addressing modes and general-purpose registers (EAX, EBX, ECX, EDX, EBP, ESI, EDI and ESP).

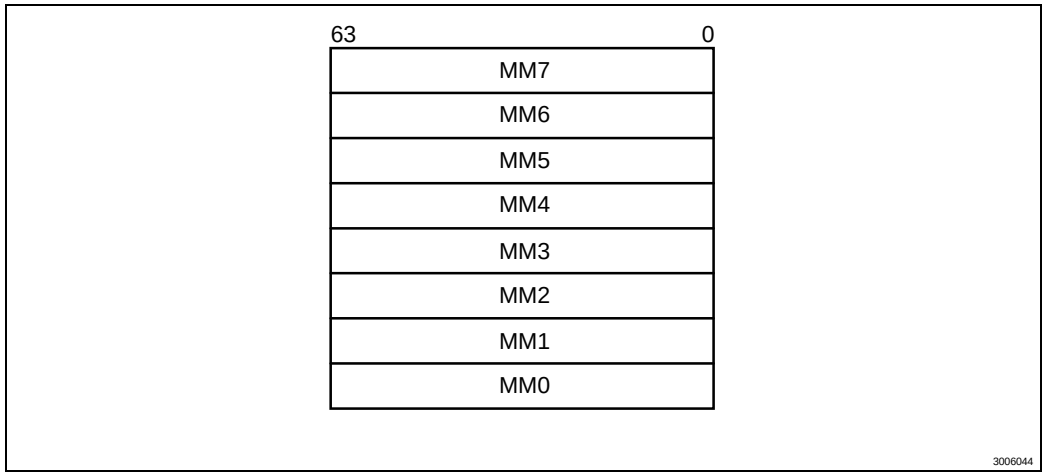


Figure 2-2. MMX™ Register Set

Although the MMX registers are defined in the Intel Architecture as separate registers, they are aliased to the registers in the FPU data register stack (R0 through R7). (See Chapter 10 in the *Intel Architecture Software Developer's Manual*, Volume 3, for a more detailed discussion of MMX register aliasing.)

2.4.1.2. MMX™ DATA TYPES

The MMX technology defines the following new 64-bit data types (see Figure 2-3):

Packed bytes	Eight bytes packed into one 64-bit quantity.
Packed words	Four (16-bit) words packed into one 64-bit quantity.
Packed doublewords	Two (32-bit) doublewords packed into one 64-bit quantity.
Quadword	One 64-bit quantity.

The bytes in the packed bytes data type are numbered 0 through 7, with byte 0 being contained in the least significant bits of the data type (bits 0 through 7) and byte 7 being contained in the most significant bits (bits 56 through 63). The words in the packed words data type and numbered 0 through 4, with word 0 being contained in the bits 0 through 15 of the data type and word 4 being contained in bits 48 through 63. The doublewords in a packed doublewords data type are numbered 0 through 1, with doubleword 0 being contained in bits 0 through 31 and doubleword 1 being contained in bits 32 through 63.

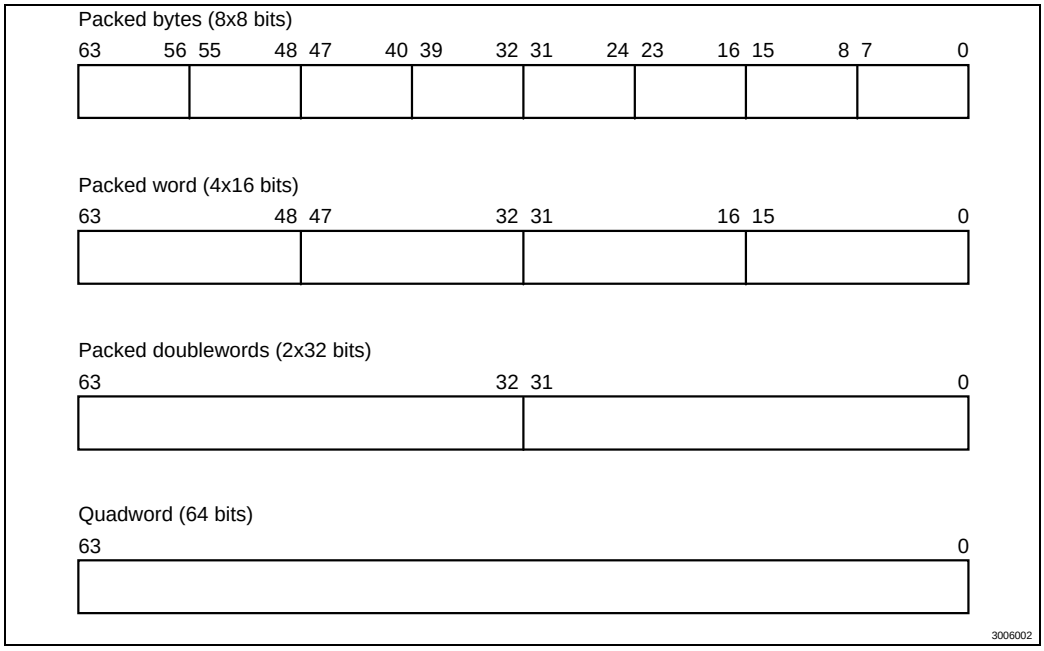


Figure 2-3. Packed Data Types

The MMX instructions move the packed data types (packed bytes, packed words or packed doublewords) and the quadword data type to-and-from memory or to-and-from the Intel Architecture general-purpose registers in 64-bit blocks. However, when performing arithmetic or logical operations on the packed data types, the MMX instructions operate in parallel on the individual bytes, words or doublewords contained in a 64-bit MMX register.

When operating on the bytes, words and doublewords within packed data types, the MMX instructions recognize and operate on both signed and unsigned byte integers, word integers and doubleword integers.

2.4.1.3. SINGLE INSTRUCTION, MULTIPLE DATA (SIMD) EXECUTION MODEL

The MMX technology uses the single instruction, multiple data (SIMD) technique for performing arithmetic and logical operations on the bytes, words or doublewords packed in an MMX packed data type. For example, the PADDSB instruction adds eight signed bytes from the source operand to eight signed bytes in the destination operand and stores eight byte-results in the destination operand. This SIMD technique speeds up software performance by allowing the same operation to be carried out on multiple data elements in parallel. The MMX technology supports parallel operations on byte, word and doubleword data elements when contained in MMX packed data types.

The SIMD execution model supported in the MMX technology directly addresses the needs of modern media, communications and graphics applications, which often use sophisticated algorithms that perform the same operations on a large number of small data types (bytes, words and doublewords). For example, most audio data is represented in 16-bit (word) quantities. The MMX instructions can operate on four of these words simultaneously with one instruction. Video and graphics information is commonly represented as palletized 8-bit (byte) quantities. Here, one MMX instruction can operate on eight of these bytes simultaneously.

2.4.1.4. MEMORY DATA FORMATS

When stored in memory the bytes, words and doublewords in the packed data types are stored in consecutive addresses, with the least significant byte, word or doubleword being stored at the lowest address and the more significant bytes, words or doublewords being stored at consecutively higher addresses (see Figure 2-4). The ordering bytes, words or doublewords in memory is always little endian. That is, the bytes with the lower addresses are less significant than the bytes with the higher addresses.

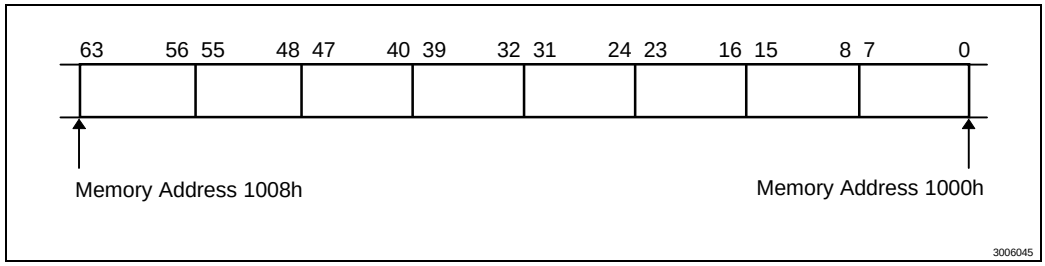


Figure 2-4. Eight Packed Bytes in Memory (at address 1000H)

2.4.1.5. MMX™ REGISTER DATA FORMATS

Values in MMX registers have the same format as a 64-bit quantity in memory. MMX registers have two data access modes: 64-bit access mode and 32-bit access mode.

The 64-bit access mode is used for 64-bit memory access, 64-bit transfer between MMX registers, all pack, logical and arithmetic instructions, and some unpack instructions.

The 32-bit access mode is used for 32-bit memory access, 32-bit transfer between integer registers and MMX registers, and some unpack instructions.

2.4.2. MMX™ Instruction Set

The MMX instruction set consists of 57 instructions, grouped into the following categories:

- Data Transfer Instructions
- Arithmetic Instructions

- Comparison Instructions
- Conversion Instructions
- Logical Instructions
- Shift Instructions
- Empty MMX State (EMMS) Instruction

These instructions provide a rich set of operations that can be performed in parallel on the bytes, words or doublewords of an MMX packed data type.

When operating on the MMX packed data types, the data within a data type is cast by the type specified by the instruction. For example, the PADDB (add packed bytes) instruction adds two groups of eight packed bytes. The PADDW (add packed words) instruction, which adds packed words, can operate on the same 64 bits as the PADDB instruction treating 64 bits as four 16-bit words.

2.4.3. Intel MMX™ Technology Pipeline Stages

The MMX unit of the Pentium processor with MMX technology has six pipeline stages. The integration of the MMX pipeline with the integer pipeline is very similar to that of the floating point pipe.

Pentium processors with MMX technology add an additional fetch stage to the pipeline. The instruction bytes are prefetched from the code cache in the prefetch (PF) stage, and they are parsed into instructions (and prefixes) in the fetch (F) stage. Additionally, any prefixes are decoded in the F stage.

When instructions execute in the two pipes, their behavior is exactly the same as if they were executed sequentially. When a stall occurs, successive instructions are not allowed to pass the stalled instruction in either pipe. Figure 2-5 shows the pipelining structure for this scheme.

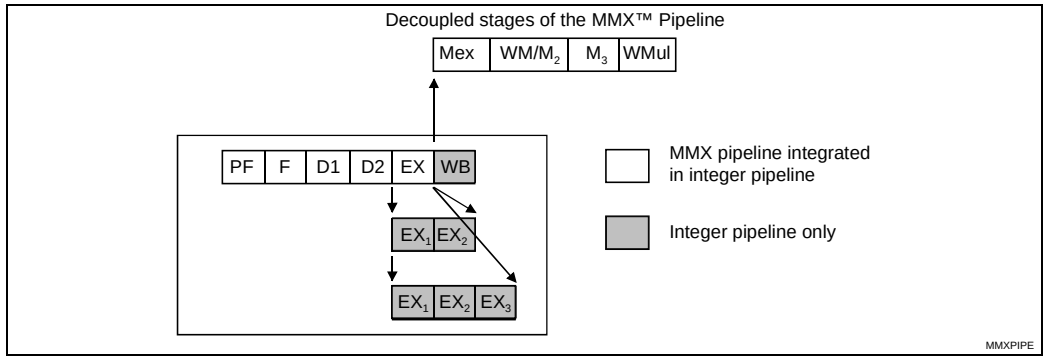


Figure 2-5. MMX™ Pipeline Structure

Instruction parsing is decoupled from the instruction decoding by means of an instruction FIFO, which is situated between the F and D1 (Decode 1) stages. The FIFO has slots for up to four instructions. This FIFO is transparent, it does not add additional latency when it is empty.

Every clock cycle, two instructions can be pushed into the instruction FIFO (depending on availability of the code bytes, and on other factors such as prefixes). Instruction pairs are pulled out of the FIFO into the D1 stage. Since the average rate of instruction execution is less than two per clock, the FIFO is normally full. If the FIFO is full, then the FIFO can buffer a stall that may have occurred during instruction fetch and parsing. If this occurs then that stall will not cause a stall in the execution stage of the pipe. If the FIFO is empty, then an execution stall may result from the pipeline being "starved" for instructions to execute. Also, if the FIFO contains only one instruction, then the instruction will not pair. Additionally, if an instruction is longer than 7 bytes, then only one instruction will be pushed into the FIFO. Figure 2-6 details the MMX pipeline on superscalar processors and the conditions where a stall may occur in the pipeline.

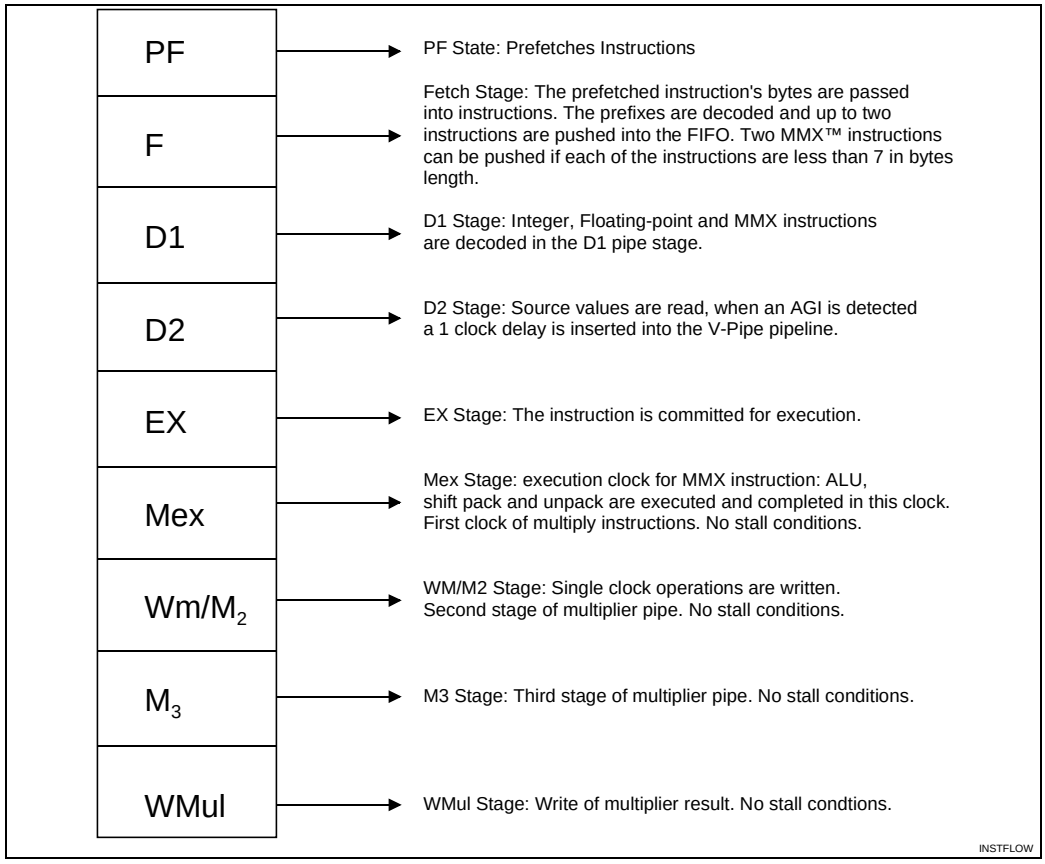


Figure 2-6. MMX™ Instruction Flow in a Pentium® Processor with MMX Technology

PF	Prefetch	Prefetches instructions
F	Fetch	The prefetched instruction bytes are passed into instructions. The prefixes are decoded and up to two instructions are pushed into the FIFO. Two MMX instructions can be pushed if each of the instructions are less than seven in bytes length.
D1	Decode1	Integer, floating-point and MMX instructions are decoded in the D1 pipe stage.
D2	Decode2	Source values are read.
E	Execution	The instruction is committed for execution.
Mex	MMX Execution	Execution clock for MMX instructions. ALU, shift, pack, and unpack instructions are executed and completed in this clock. First clock of multiply instructions. No stall conditions.
WM/M_2	Write/Multiply2	Single clock operations are written. Second stage of multiplier pipe. No stall conditions.
M_3	Multiply3	Third stage of multiplier pipe. No stall conditions.
Wmul	Write of multiply	Write of multiplier result. Not stall conditions.

2.4.4. Instruction Issue

The rules of how MMX instructions get issued on the Pentium processor with MMX technology are summarized as follows:

1. Pairing of two MMX instructions can be performed.
2. Pairing of one MMX instruction with an integer instruction can be performed.
3. MMX instructions do not get paired with floating-point instructions.

2.4.4.1. PAIRING TWO MMX™ INSTRUCTIONS

The rules of how two MMX instructions can be paired are listed below:

- Two MMX instructions which both use the MMX shifter unit (pack, unpack and shift instructions) cannot pair since there is only one MMX shifter unit. Shift operations may be issued in either the u-pipe or the v-pipe but not in both in the same clock cycle.
- Two MMX instructions which both use the MMX multiplier unit (pmull, pmulh, pmadd type instructions) cannot pair since there is only one MMX multiplier unit. Multiply operations may be issued in either the u-pipe or the v-pipe but not in both in the same clock cycle.

- MMX instructions which access either memory or the integer register file can be issued in the u-pipe only. Do not schedule these instructions to the v-pipe as they will wait and be issued in the next pair of instructions (and to the u-pipe).
- The MMX destination register of the u-pipe instruction should not match the source or destination register of the v-pipe instruction (dependency check).
- The EMMS instruction is not pairable.
- If either the CR0.TS or the CR0.EM bits are set, MMX instructions cannot go into the v-pipe.

2.4.4.2. PAIRING AN INTEGER INSTRUCTION IN THE U-PIPE WITH AN MMX™ INSTRUCTION IN THE V-PIPE

The rules of how an integer instruction in the u-pipe is paired with an MMX instruction in the v-pipe are listed below:

- The MMX instruction can not be the first MMX instruction following a floating-point instruction.
- The v-pipe MMX instruction does not access either memory or the integer register file.
- The u-pipe integer instruction is a pairable u-pipe integer instruction.

2.4.4.3. PAIRING AN MMX™ INSTRUCTION IN THE U-PIPE WITH AN INTEGER INSTRUCTION IN THE V-PIPE

The rules of how an MMX instruction in the u-pipe is paired with an integer instruction in the v-pipe are listed below:

- The v-pipe instruction is a pairable integer v-pipe instruction.
- The u-pipe MMX instruction does not access either memory or the integer register file.

2.5. ON-CHIP CACHES

The Pentium processor (75/90/100/120/133/150/166/200) implements two internal caches for a total integrated cache size of 16 Kbytes: an 8 Kbyte data cache and a separate 8 Kbyte code cache. These caches are transparent to application software to maintain compatibility with previous Intel Architecture generations. The Pentium processor with MMX technology doubles the internal cache size to 32 Kbytes: a 16 Kbyte data cache and a separate 16 Kbyte code cache.

The data cache fully supports the MESI (modified/exclusive/shared/invalid) writeback cache consistency protocol. The code cache is inherently write protected to prevent code from being inadvertently corrupted, and as a consequence supports a subset of the MESI protocol, the S (shared) and I (invalid) states.

The caches have been designed for maximum flexibility and performance. The data cache is configurable as writeback or writethrough on a line-by-line basis. Memory areas can be defined as non-cacheable by software and external hardware. Cache writeback and invalidations can be initiated by hardware or software. Protocols for cache consistency and line replacement are implemented in hardware, easing system design.

2.5.1. Cache Organization

On the Pentium processor (75/90/100/120/133/150/166/200), each of the caches are 8 Kbytes in size and each is organized as a 2-way set associative cache. There are 128 sets in each cache, each set containing 2 lines (each line has its own tag address). Each cache line is 32 bytes wide. The Pentium processor with MMX technology has two 16 Kbyte 4-way set-associative caches the with a cache line length of 32 bytes.

In the Pentium processor (75/90/100/120/133/150/166/200), replacement in both the data and instruction caches is handled by the LRU mechanism which requires one bit per set in each of the caches. The Pentium processor with MMX technology employs a pseudo-LRU replacement algorithm which requires three blts per set in each of the caches. When a line must be replaced, the cache will first select which of l0:l1 and l2:l3 was least recently used. Then the cache will determine which of the two lines was least recently used and mark it for replacement. This decision tree is shown in Figure 2-7.

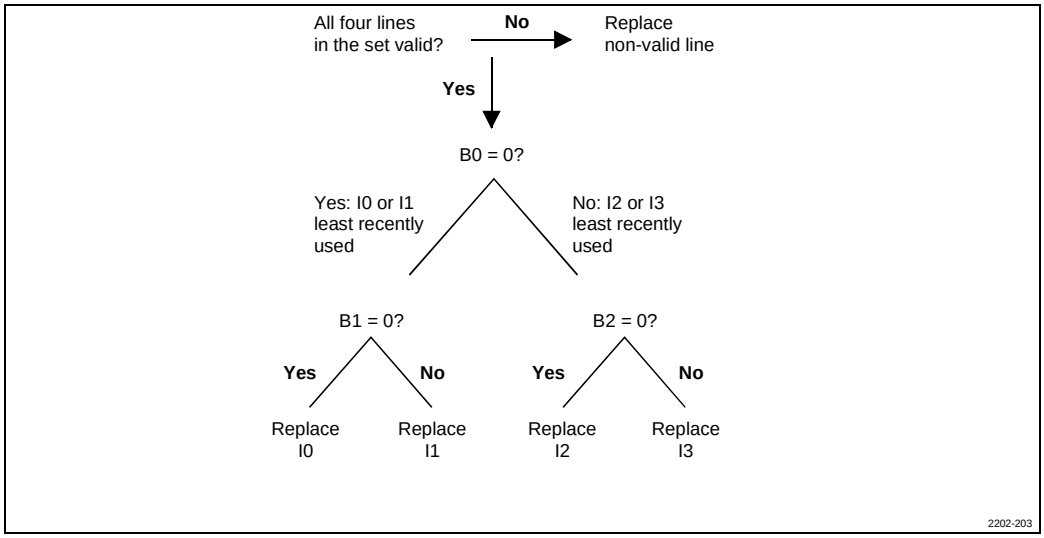


Figure 2-7. Pseudo-LRU Cache Replacement Strategy

The data cache consists of eight banks interleaved on 4-byte boundaries. The data cache can be accessed simultaneously from both pipes, as long as the references are to different cache banks. A conceptual diagram of the organization of the data and code caches is shown in

Figure 2-8. Note that the data cache supports the MESI writeback cache consistency protocol which requires 2 state bits, while the code cache supports the S and I state only and therefore requires only one state bit.

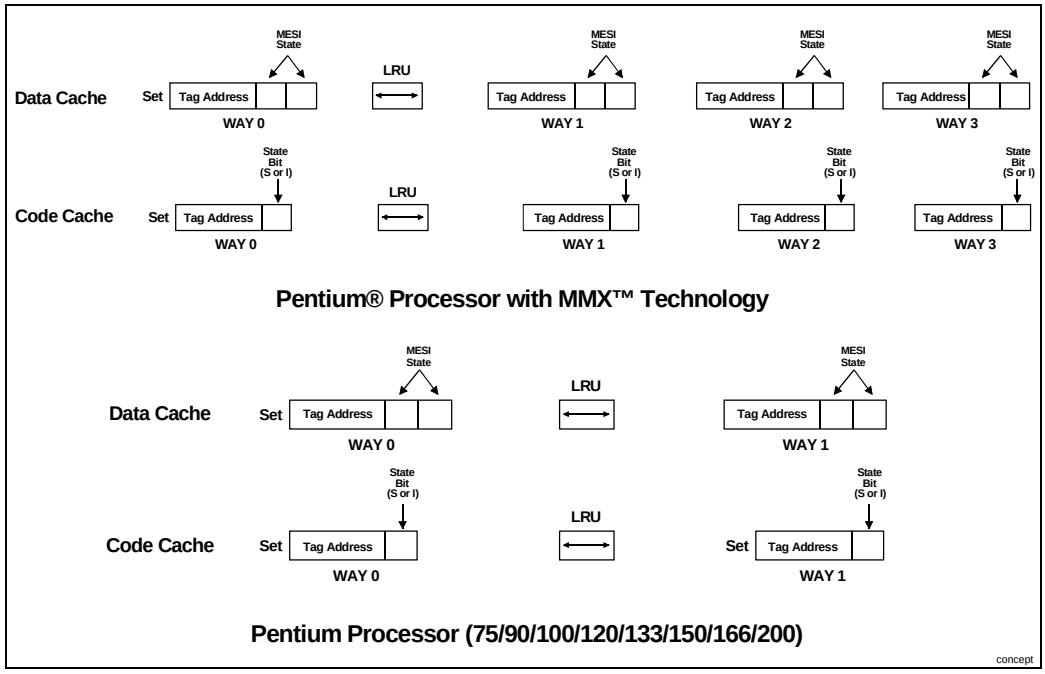


Figure 2-8. Conceptual Organization of Code and Data Caches

2.5.2. Cache Structure

The instruction and data caches can be accessed simultaneously. The instruction cache can provide up to 32 bytes of raw opcodes and the data cache can provide data for two data references all in the same clock. This capability is implemented partially through the tag structure. The tags in the data cache are triple ported. One of the ports is dedicated to snooping while the other two are used to lookup two independent addresses corresponding to data references from each of the pipelines. The instruction cache tags of the Pentium processor (75/90/100/120/133/150/166/200) are also triple ported. Again, one port is dedicated to support snooping and the other two ports facilitate split line accesses (simultaneously accessing upper half of one line and lower half of the next line). Note, the Pentium processor with MMX technology does not support split line accesses to the code cache; as such, its code cache tags are dual ported.

The storage array in the data cache is single ported but interleaved on 4-byte boundaries to be able to provide data for two simultaneous accesses to the same cache line.

Each of the caches are parity protected. In the instruction cache, there are parity bits on a quarter line basis and there is one parity bit for each tag. The data cache contains one parity bit for each tag and a parity bit per byte of data.

Each of the caches are accessed with physical addresses and each cache has its own TLB (translation lookaside buffer) to translate linear addresses to physical addresses. The TLBs associated with the instruction cache are single ported whereas the data cache TLBs are fully dual ported to be able to translate two independent linear addresses for two data references simultaneously. The tag and data arrays of the TLBs are parity protected with a parity bit associated with each of the tag and data entries in the TLBs.

The data cache of the Pentium processor (75/90/100/120/133/150/166/200) has a 4-way set associative, 64-entry TLB for 4-Kbyte pages and a separate 4-way set associative, 8-entry TLB to support 4-Mbyte pages. The code cache has one 4-way set associative, 32-entry TLB for 4-Kbyte pages and 4-Mbyte pages which are cached in 4-Kbyte increments. Replacement in the TLBs is handled by a pseudo LRU mechanism (similar to the Intel486 CPU) that requires 3 bits per set. The Pentium processor with MMX technology has a 64-entry fully associative data TLB and a 32-entry fully associative code TLB. Both TLBs can support 4Kbyte pages as well as 4 Mbyte pages.

2.5.3. Cache Operating Modes

The operating modes of the caches are controlled by the CD (cache disable) and NW (not writethrough) bits in CR0. See Table 2-1 for a description of the modes. For normal operation and highest performance, these bits should both be reset to “0.” The bits come out of RESET as CD = NW = 1.

When the L1 cache is disabled (CR0.NW and CR0.CD bits are both set to ‘1’) external snoops are accepted in a DP system and inhibited in a UP system. Note that when snoops are inhibited, address parity is not checked, and APCHK# will not be asserted for a corrupt address. However, when snoops are accepted, address parity is checked (and APCHK# will be asserted for corrupt addresses).

Table 2-1. Cache Operating Modes

CD	NW	Description
1	1	Read hits access the cache.
		Read misses do not cause linefills.
		Write hits update the cache, but do not access memory.
		Write hits will cause Exclusive State lines to change to Modified State.
		Shared lines will remain in the Shared state after write hits.
		Write misses access memory.
		Inquire and invalidation cycles do not affect the cache state or contents.
		This is the state after reset.
1	0	Read hits access the cache.
		Read misses do not cause linefills.
		Write hits update the cache.
		Writes to Shared lines and write misses update external memory.
		Writes to Shared lines can be changed to the Exclusive State under the control of the WB/WT# pin.
		Inquire cycles (and invalidations) are allowed.
0	1	GP(0)
0	0	Read hits access the cache.
		Read misses may cause linefills.
		These lines will enter the Exclusive or Shared state under the control of the WB/WT# pin.
		Write hits update the cache.
		Only writes to shared lines and write misses appear externally.
		Writes to Shared lines can be changed to the Exclusive State under the control of the WB/WT# pin.
		Inquire cycles (and invalidations) are allowed.

To completely disable the cache, the following two steps must be performed:

1. CD and NW must be set to 1.
2. The caches must be flushed.

If the cache is not flushed, cache hits on reads will still occur and data will be read from the cache. In addition, the cache must be flushed after being disabled to prevent any inconsistencies with memory.

2.5.4. Page Cacheability

Two bits for cache control, PWT and PCD are defined in the page table and page directory entries. The state of these bits are driven out on the PWT and PCD pins during memory access cycles. The PWT bit controls write policy for the second level caches used with the Pentium processor. Setting PWT to 1 defines a writethrough policy for the current page, while clearing PWT to 0 defines a writeback policy for the current page.

The PCD bit controls cacheability on a page-by-page basis. The PCD bit is internally ANDed with the KEN# signal to control cacheability on a cycle-by-cycle basis. PCD = 0 enables cacheing, while PCD = 1 disables it. Cache linefills are enabled when PCD = 0 and KEN# = 0.

2.5.4.1. PCD AND PWT GENERATION

The value driven on PCD is a function of the PWT bits in CR3, the page directory pointer, the page directory entry and the page table entry, and the CD and PG bits in CR0.

The value driven on PWT is a function of the PCD bits in CR3, the page directory pointer, the page directory entry and the page table entry, and the PG bit in CR0 (CR0.CD does not affect PWT).

CR0.CD = 1

If cacheing is disabled, the PCD pin is always driven high. CR0.CD does not affect the PWT pin.

CR0.PG = 0

If paging is disabled, the PWT pin is forced low and the PCD pin reflects the CR0.CD. The PCD and PWT bits in CR3 are assumed 0 during the caching process.

CR0.CD = 0, PG = 1, normal operation

The PCD and PWT bits from the last entry (can be either PDE or PTE, depends on 4 Mbyte or 4 Kbyte mode) are cached in the TLB and are driven anytime the page mapped by the TLB entry is referenced.

CR0.CD = 0, PG = 1, during TLB Refresh

During TLB refresh cycles when the PDE and PTE entries are read, the PWT and PCD bits are obtained as shown in Table 2-2 and Table 2-3.

Table 2-2. 32-Bits/4-Kbyte Pages

PCD/PWT Taken From	During Accesses To
CR3	PDE
PDE	PTE
PTE	All other paged mem references

Table 2-3. 32-Bits/4-Mbyte Pages

PCD/PWT Taken From	During Accesses To
CR3	PDE
PDE	All other paged mem references

Figure 2-9 shows how PCD and PWT are generated.

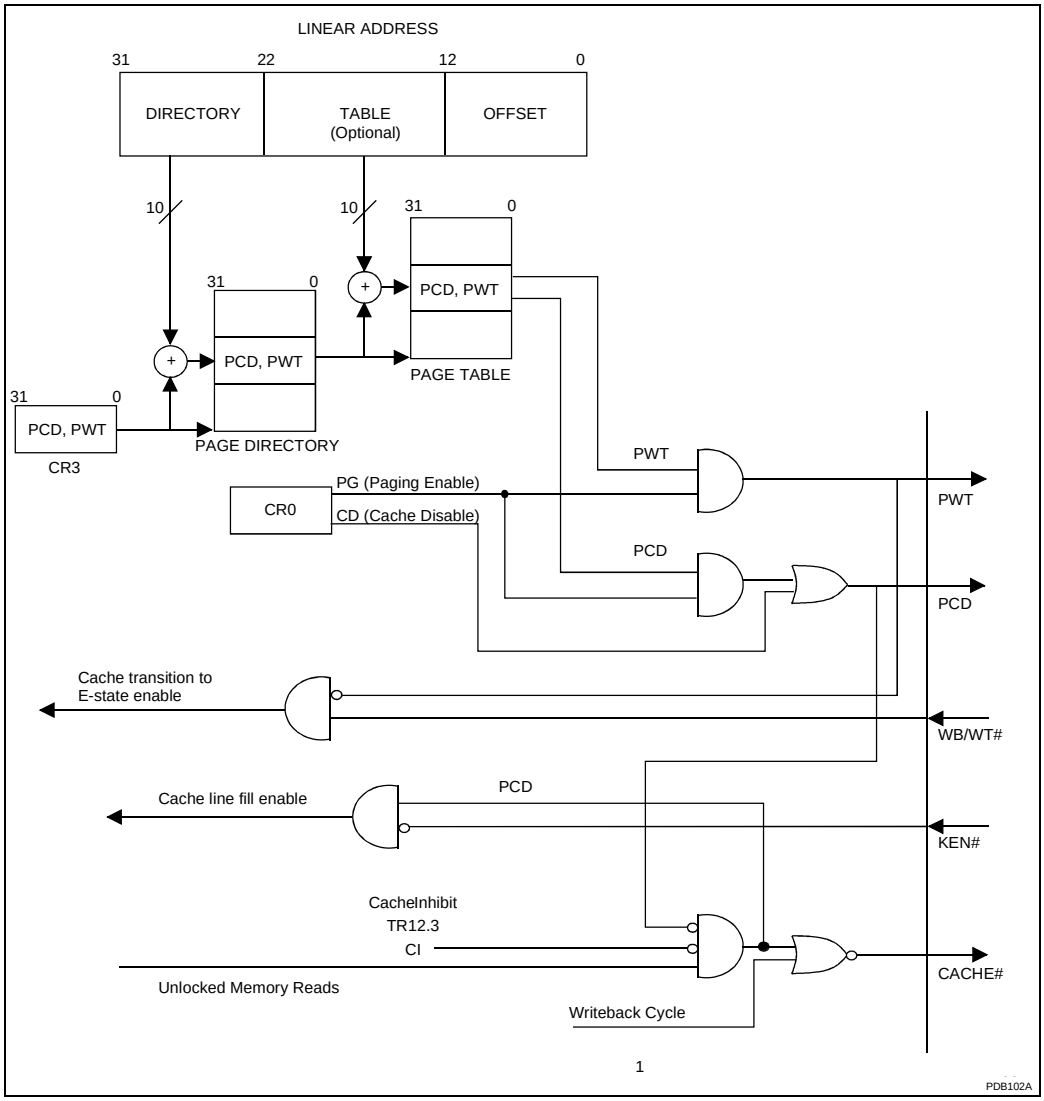


Figure 2-9. PCD and PWT Generation

2.5.5. Inquire Cycles

Inquire cycles are initiated by the system to determine if a line is present in the code or data cache, and what its state is. This document refers to inquire cycles and snoop cycles interchangeably.

Inquire cycles are driven to the Pentium processor when a bus master other than the Pentium processor initiates a read or write bus cycle. Inquire cycles are driven to the Pentium processor when the bus master initiates a read to determine if the Pentium processor data cache contains the latest information. If the snooped line is in the Pentium processor data cache in the modified state, the Pentium processor has the most recent information and must schedule a writeback of the data. Inquire cycles are driven to the Pentium processor when the other bus master initiates a write to determine if the Pentium processor code or data cache contains the snooped line and to invalidate the line if it is present. Inquire cycles are described in detail in the "Bus Functional Description" chapter.

2.5.6. Cache Flushing

The on-chip cache can be flushed by external hardware or by software instructions.

Flushing the cache through hardware is accomplished by driving the FLUSH# pin low. This causes the cache to writeback all modified lines in the data cache and mark the state bits for both caches invalid. The Flush Acknowledge special cycle is driven by the Pentium processor when all writebacks and invalidations are complete.

The INVD and WBINVD instructions cause the on-chip caches to be invalidated also. WBINVD causes the modified lines in the internal data cache to be written back, and all lines in both caches to be marked invalid. After execution of the WBINVD instruction, the Writeback and Flush special cycles are driven to indicate to any external cache that it should writeback and invalidate its contents.

INVD causes all lines in both caches to be invalidated. Modified lines in the data cache are not written back. The Flush special cycle is driven after the INVD instruction is executed to indicate to any external cache that it should invalidate its contents. Care should be taken when using the INVD instruction that cache consistency problems are not created.

Note that the implementation of the INVD and WBINVD instructions are processor dependent. Future processor generations may implement these instructions differently.

2.5.7. Data Cache Consistency Protocol (MESI Protocol)

The Pentium processor Cache Consistency Protocol is a set of rules by which states are assigned to cached entries (lines). The rules apply for memory read/write cycles only. I/O and special cycles are not run through the data cache.

Every line in the Pentium processor data cache is assigned a state dependent on both Pentium processor generated activities and activities generated by other bus masters (snooping). The Pentium processor Data Cache Protocol consists of four states that define whether a line is

valid (HIT/MISS), if it is available in other caches, and if it has been MODIFIED. The four states are the M (Modified), E (Exclusive), S (Shared) and the I (Invalid) states and the protocol is referred to as the MESI protocol. A definition of the states is given below:

M - Modified: An M-state line is available in ONLY one cache and it is also MODIFIED (different from main memory). An M-state line can be accessed (read/written to) without sending a cycle out on the bus.

E - Exclusive: An E-state line is also available in ONLY one cache in the system, but the line is not MODIFIED (i.e., it is the same as main memory). An E-state line can be accessed (read/written to) without generating a bus cycle. A write to an E-state line will cause the line to become MODIFIED.

S - Shared: This state indicates that the line is potentially shared with other caches (i.e. the same line may exist in more than one cache). A read to an S-state line will not generate bus activity, but a write to a SHARED line will generate a write through cycle on the bus. The write through cycle may invalidate this line in other caches. A write to an S-state line will update the cache.

I - Invalid: This state indicates that the line is not available in the cache. A read to this line will be a MISS and may cause the Pentium processor to execute a LINE FILL (fetch the whole line into the cache from main memory). A write to an INVALID line will cause the Pentium processor to execute a write-through cycle on the bus.

2.5.7.1. STATE TRANSITION TABLES

Lines cached in the Pentium processor can change state because of processor generated activity or as a result of activity on the Pentium processor bus generated by other bus masters (snooping). State transitions happen because of processor generated transactions (memory reads/writes) and by a set of external input signals and internally generated variables. The Pentium processor also drives certain pins as a consequence of the Cache Consistency Protocol.

2.5.7.1.1. Read Cycle

Table 2-4 shows the state transitions for lines in the data cache during unlocked read cycles.

Table 2-4. Data Cache State Transitions for UNLOCKED Pentium® Processor Initiated Read Cycles*

Present State	Pin Activity	Next State	Description
M	n/a	M	Read hit; data is provided to processor core by cache. No bus cycle is generated.
E	n/a	E	Read hit; data is provided to processor core by cache. No bus cycle is generated.
S	n/a	S	Read hit; data is provided to the processor by the cache. No bus cycle is generated.
I	CACHE# low AND KEN# low AND WB/WT# high AND PWT low	E	Data item does not exist in cache (MISS). A bus cycle (read) will be generated by the Pentium® processor. This state transition will happen if WB/WT# is sampled high with first BRDY# or NA#.
I	CACHE# low AND KEN# low AND (WB/WT# low OR PWT high)	S	Same as previous read miss case except that WB/WT# is sampled low with first BRDY# or NA#.
I	CACHE# high OR KEN# high	I	KEN# pin inactive; the line is not intended to be cached in the Pentium processor.

NOTE: *Locked accesses to the data cache will cause the accessed line to transition to the Invalid state

Note the transition from I to E or S states (based on WB/WT#) happens only if KEN# is sampled low with the first of BRDY# or NA#, and the cycle is transformed into a LINE FILL cycle. If KEN# is sampled high, the line is not cached and remains in the I state.

2.5.7.1.2. Write Cycle

The state transitions of data cache lines during Pentium processor generated write cycles are illustrated in the next table. Writes to SHARED lines in the data cache are always sent out on the bus along with updating the cache with the write item. The status of the PWT and WB/WT# pins during these write cycles on the bus determines the state transitions in the data cache during writes to S-state lines.

A write to a SHARED line in the data cache will generate a write cycle on the Pentium processor bus to update memory and/or invalidate the contents of other caches. If the PWT pin is driven high when the write cycle is run on the bus, the line will be updated, and will stay in the S-state regardless of the status of the WB/WT# pin that is sampled with the first BRDY# or NA#. If PWT is driven low, the status of the WB/WT# pin sampled along with the first BRDY# or NA# for the write cycle determines what state (E:S) the line transitions to.

The state transition from S to E is the only transition in which the data and the status bits are not updated at the same time. The data will be updated when the write is written to the Pentium processor write buffers. The state transition does not occur until the write has completed on the bus (BRDY# has been returned). Writes to the line after the transition to the E-state will not generate bus cycles. However, it is possible that writes to the same line that were buffered or in the pipeline before the transition to the E-state will generate bus cycles after the transition to E-state.

An inactive EWBE# input will stall subsequent writes to an E- or an M-state line. All subsequent writes to E- or M-state lines are held off until EWBE# is returned active.

Table 2-5. Data Cache State Transitions for Pentium® Processor Initiated Write Cycles

Present State	Pin Activity	Next State	Description
M	n/a	M	Write hit; update data cache. No bus cycle generated to update memory.
E	n/a	M	Write hit; update cache only. No bus cycle generated; line is now MODIFIED.
S	PWT low AND WB/WT# high	E	Write hit; data cache updated with write data item. A write-through cycle is generated on bus to update memory and/or invalidate contents of other caches. The state transition occurs after the writethrough cycle completes on the bus (with the last BRDY#).
S	PWT low AND WB/WT# low	S	Same as above case of write to S-state line except that WB/WT# is sampled low.
S	PWT high	S	Same as above cases of writes to S state lines except that this is a write hit to a line in a writethrough page; status of WB/WT# pin is ignored.
I	n/a	I	Write MISS; a writethrough cycle is generated on the bus to update external memory. No allocation done.

NOTE: Memory writes are buffered while I/O writes are not. There is no guarantee of synchronization between completion of memory writes on the bus and instruction execution after the write. A serializing instruction needs to be executed to synchronize writes with the next instruction if necessary.

2.5.7.1.3. Inquire Cycles (Snooping)

The purpose of inquire cycles is to check whether the address being presented is contained within the caches in the Pentium processor. Inquire cycles may be initiated with or without an INVALIDATION request (INV = 1 or 0). An inquire cycle is run through the data and code caches through a dedicated snoop port to determine if the address is in one of the Pentium processor caches. If the address is in a Pentium processor cache, the HIT# pin is asserted. If the address hits a modified line in the data cache, the HITM# pin is also asserted and the modified line is then written back onto the bus.

The state transition tables for inquire cycles are given below:

Table 2-6. Cache State Transitions During Inquire Cycles

Present State	Next State INV=1	Next State INV=0	Description
M	I	S	Snoop hit to a MODIFIED line indicated by HIT# and HITM# pins low. Pentium® processor schedules the writing back of the modified line to memory.
E	I	S	Snoop hit indicated by HIT# pin low; no bus cycle generated.
S	I	S	Snoop hit indicated by HIT# pin low; no bus cycle generated.
I	I	I	Address not in cache; HIT# pin high.

2.5.7.2. PENTIUM® PROCESSOR CODE CACHE CONSISTENCY PROTOCOL

The Pentium processor code cache follows a subset of the MESI protocol. Accesses to the code cache are either a Hit (Shared) or a Miss (Invalid).

In the case of a read hit, the cycle is serviced internally to the Pentium processor and no bus activity is generated. In the case of a read miss, the read is sent to the external bus and may be converted to a linefill.

Lines are never overwritten in the code cache. Writes generated by the Pentium processor are snooped by the code cache. If the snoop is a hit in the code cache, the line is invalidated. If there is a miss, the code cache is not affected.

2.6. WRITE BUFFERS AND MEMORY ORDERING

The Pentium processor (75/90/100/120/133/150/166/200) has two write buffers, one corresponding to each of the pipelines, to enhance the performance of consecutive writes to memory. These write buffers are one quadword wide (64-bits) and can be filled simultaneously in one clock e.g., by two simultaneous write misses in the two instruction pipelines. Writes in these buffers are driven out on the external bus in the order they were generated by the processor core. No reads (as a result of cache miss) are reordered around previously generated writes sitting in the write buffers. The implication of this is that the write buffers will be

flushed or emptied before a subsequent bus cycle is run on the external bus (unless BOFF# is asserted and a writeback cycle becomes pending, see section 2.6.3.).

The Pentium processor with MMX technology has four write buffers that can be used by either the u-pipe or v-pipe. Posting writes to these buffers enables the pipe to continue advancing when consecutive writes to memory occur. The writes will be executed on the bus as soon as it is free, in FIFO order. Reads cannot bypass writes posted in these buffers.

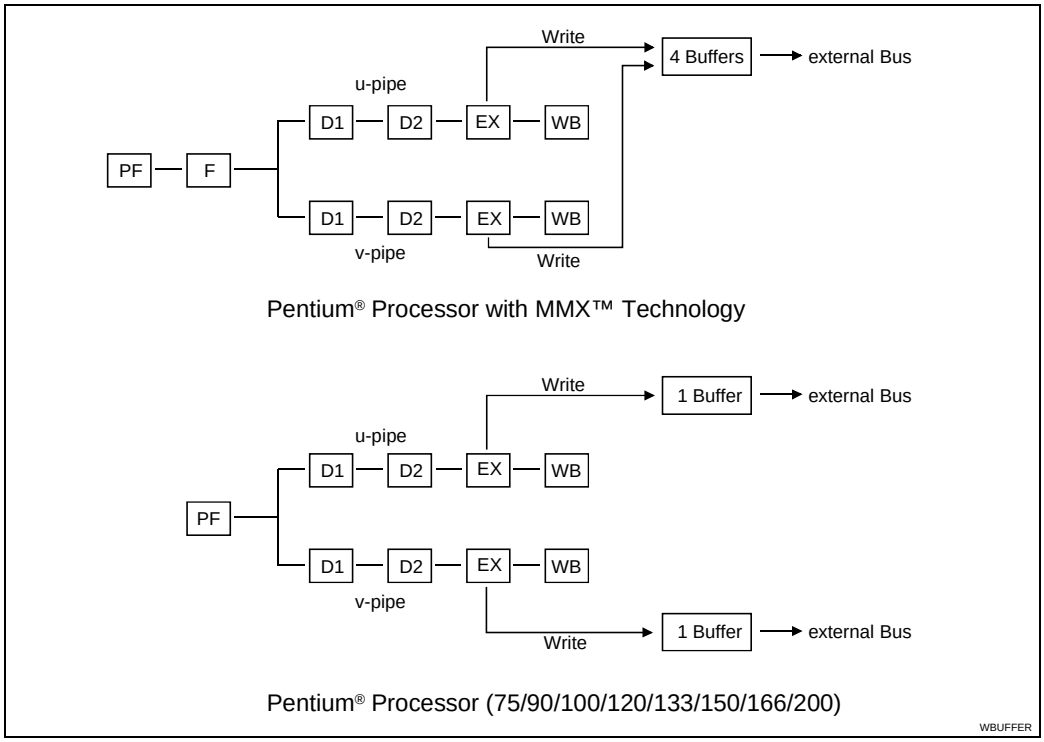


Figure 2-10. Pentium® Processor Write Buffer Implementation

The Pentium processor supports strong write ordering only. That is, writes generated by the Pentium processor will be driven to the bus or updated in the cache in the order that they occur. The Pentium processor will not write to E or M-state lines in the data cache if there is a write in either write buffer, if a write cycle is running on the bus, or if EWBE# is inactive.

Note that only memory writes are buffered and I/O writes are not. There is no guarantee of synchronization between completion of memory writes on the bus and instruction execution after the write. The OUT instruction or a serializing instruction needs to be executed to synchronize writes with the next instruction. Please refer to the “Serializing Operations” section for more information.

No re-ordering of read cycles occurs on the Pentium processor. Specifically, the write buffers are flushed before the IN instruction is executed.

2.6.1. External Event Synchronization

When the system changes the value of NMI, INTR, FLUSH#, SMI# or INIT as the result of executing an OUT instruction, these inputs must be at a valid state three clocks before BRDY# is returned to ensure that the new value will be recognized before the next instruction is executed.

Note that if an OUT instruction is used to modify A20M#, this will not affect previously prefetched instructions. A serializing instruction must be executed to guarantee recognition of A20M# before a specific instruction.

2.6.2. Serializing Operations

After executing certain instructions the Pentium processor serializes instruction execution. This means that any modifications to flags, registers, and memory for previous instructions are completed before the next instruction is fetched and executed. The prefetch queue is flushed as a result of serializing operations.

The Pentium processor serializes instruction execution after executing one of the following instructions: MOV to Debug Register, MOV to Control Register, INVD, INVLPG, IRET, IRETD, LGDT, LLDT, LIDT, LTR, WBINVD, CPUID, RSM and WRMSR.

NOTE

1. The CPUID instruction can be executed at any privilege level to serialize instruction execution.
2. When the Pentium processor serializes instruction execution, it ensures that it has completed any modifications to memory, including flushing any internally buffered stores; it then waits for the EWBE# pin to go active before fetching and executing the next instruction. Pentium processor systems may use the EWBE# pin to indicate that a store is pending externally. In this manner, a system designer may ensure that all externally pending stores will complete before the Pentium processor begins to fetch and execute the next instruction.
3. The Pentium processor does not generally writeback the contents of modified data in its data cache to external memory when it serializes instruction execution. Software can force modified data to be written back by executing the WBINVD instruction.
4. Whenever an instruction is executed to enable/disable paging (that is, change the PG bit of CR0), this instruction must be followed with a jump. The instruction at the target of the branch is fetched with the new value of PG (i.e., paging enabled/disabled), however, the jump instruction itself

is fetched with the previous value of PG. Intel386™, Intel486 and Pentium processors have slightly different requirements to enable and disable paging. In all other respects, an MOV to CR0 that changes PG is serializing. Any MOV to CR0 that does not change PG is completely serializing.

5. Whenever an instruction is executed to change the contents of CR3 while paging is enabled, the next instruction is fetched using the translation tables that correspond to the new value of CR3. Therefore the next instruction and the sequentially following instructions should have a mapping based upon the new value of CR3.
6. The Pentium processor implements branch-prediction techniques to improve performance by prefetching the destination of a branch instruction before the branch instruction is executed. Consequently, instruction execution is not generally serialized when a branch instruction is executed.
7. Although the I/O instructions are not “serializing” because the processor does not wait for these instructions to complete before it prefetches the next instruction, they do have the following properties that cause them to function in a manner that is identical to previous generations. I/O reads are not re-ordered within the processor; they wait for all internally pending stores to complete. Note that the Pentium processor does not sample the EWBE# pin during reads. If necessary, external hardware must ensure that externally pending stores are complete before returning BRDY#. This is the same requirement that exists on Intel386 and Intel486 systems. The OUT and OUTS instructions are also not “serializing,” as they do not stop the prefetcher. They do, however, ensure that all internally buffered stores have completed, that EWBE# has been sampled active indicating that all externally pending stores have completed and that the I/O write has completed before they begin to execute the next instruction. Note that unlike the Intel486 processor, it is not necessary for external hardware to ensure that externally pending stores are complete before returning BRDY#.
8. On the Pentium processor with MMX technology, serializing instructions require an additional clock to complete compared to the Pentium processor (75/90/100/120/133/150/166/200) due to the additional pipeline stage.

2.6.3. Linefill and Writeback Buffers

In addition to the write buffers corresponding to each of the internal pipelines, the Pentium processor has 3 writeback buffers. Each of the writeback buffers are 1 deep and 32-bytes (1 line) wide.

There is a dedicated replacement writeback buffer which stores writebacks caused by a linefill that replaces a modified line in the data cache. There is one external snoop writeback buffer

that stores writebacks caused by an inquire cycle that hits a modified line in the data cache. Finally, there is an internal snoop writeback buffer that stores writebacks caused by an internal snoop cycle that hits a modified line in the data cache. Internal and external snoops are discussed in detail in the Inquire Cycle section of the Bus Functional Description chapter of this document (Chapter 6). Write cycles are driven to the bus with the following priority:

- Contents of external snoop writeback buffer
- Contents of internal snoop writeback buffer
- Contents of replacement writeback buffer
- Contents of write buffers.

Note that the contents of whichever write buffer was written into first is driven to the bus first. If both write buffers were written to in the same clock, the contents of the u-pipe buffer is written out first. In the Pentium processor with MMX technology, the write buffers are written in order as well, even though there is no u-pipe buffer and v-pipe buffer.

The Pentium processor also implements two linefill buffers, one for the data cache and one for the code cache. As information (data or code) is returned to the Pentium processor for a cache linefill, it is written into the linefill buffer. After the entire line has been returned to the processor it is transferred to the cache. Note that the processor requests the needed information first and uses that information as soon as it is returned. The Pentium processor does not wait for the linefill to complete before using the requested information.

If a line fill causes a modified line in the data cache to be replaced, the replaced line will remain in the cache until the linefill is complete. After the linefill is complete, the line being replaced is moved into the replacement writeback buffer and the new linefill is moved into the cache.

2.7. EXTERNAL INTERRUPT CONSIDERATIONS

The Pentium processor recognizes the following external interrupts: BUSCHK#, R/S#, FLUSH#, SMI#, INIT, NMI, INTR and STPCLK#. These interrupts are recognized at instruction boundaries. On the Pentium processor, the instruction boundary is the first clock in the execution stage of the instruction pipeline. This means that before an instruction is executed, the Pentium processor checks to see if any interrupts are pending. If an interrupt is pending, the processor flushes the instruction pipeline and then services the interrupt.

The Pentium processor interrupt priority scheme is shown in Table 2-7.

Table 2-7. Pentium® Processor Interrupt Priority Scheme

	ITR = 0 (default)	ITR = 1
1.	Breakpoint (INT 3)	Breakpoint (INT 3)
2.	BUSCHK#	BUSCHK#
3.	Debug Traps (INT 1)	FLUSH#
4.	R/S#	SMI#
5.	FLUSH#	Debug Traps (INT 1)
6.	SMI#	R/S#
7.	INIT	INIT
8.	NMI	NMI
9.	INTR	INTR
10.	Floating-Point Error	Floating-Point Error
11.	STPCLK#	STPCLK#
12.	Faults on Next Instruction	Faults on Next Instruction

NOTE: ITR is bit 9 of the TR12 register

2.8. INTRODUCTION TO DUAL PROCESSOR MODE

Symmetric dual processing in a system is supported with two Pentium processors sharing a single second-level cache. The processors must be of the same type, either two Pentium processors (75/90/100/120/133/150/166/200) or two Pentium processors with MMX technology. The two processors appear to the system as a single Pentium processor. Multiprocessor operating systems properly schedule computing tasks between the two processors. This scheduling of tasks is transparent to software applications and the end-user. Logic built into the processors support a “glueless” interface for easy system design. Through a private bus, the two Pentium processors arbitrate for the external bus and maintain cache coherency.

In this document, in order to distinguish between two Pentium processors in dual processing mode, one CPU will be designated as the Primary processor with the other being the Dual processor. Note that this is a different concept than that of “master” and “checker” processors.

The Dual processor is a configuration option of the Pentium processor. The Dual processor must operate at the same bus and core frequency and bus/core ratio as the Primary processor.

The Primary and Dual processors include logic to maintain cache consistency between the processors and to arbitrate for the common bus. The cache consistency and bus arbitration activity will cause the dual processor pair to issue extra bus cycles that will not appear in a Pentium processor uniprocessor system.

Chapter 3 describes in detail how the DP bootup, cache consistency, and bus arbitration mechanisms operate. In order to operate properly in dual processing mode, the Primary and Dual processors require private APIC, cache consistency, and bus arbitration interfaces, as well as a multiprocessing-ready operating system.

The dual processor interface allows the Dual processor to be added for a substantial increase in system performance. The interface allows the Primary and Dual processor to operate in a coherent manner that is transparent to the system.

The memory subsystem transparency was the primary goal of the cache coherency and bus arbitration mechanisms.

2.8.1. Dual Processing Terminology

This section defines some terms used in the following discussions. They are here to ensure your understanding of the explanations and examples in remainder of this document.

Symmetric Multi-Processing:	Two or more processors operating with equal priorities in a system. No individual processor is a master, and none is a slave.
DP or Dual Processing:	The Primary and Dual processor operating symmetrically in a system sharing a second-level cache.
MRM or Most Recent Master:	The processor (either the Primary or Dual) which currently owns the processor address bus. When interprocessor pipelining, this is the processor which last issued an ADS#.
LRM or Least Recent Master:	The processor (either the Primary or Dual) which does not own the address bus. The LRM automatically snoops every ADS# from the MRM processor in order to maintain level one cache coherency.
Primary Processor:	The Pentium processor when CPUTYP = V_{SS} (or left floating).
Upgrade Processor:	The future Pentium OverDrive processor.
Dual Processor:	The Pentium processor when CPUTYP = V_{CC}.
OverDrive Processor:	The future Pentium OverDrive processor.

2.8.2. Dual Processing Overview

The Primary and Dual processor both have logic built-in to support “glueless” dual-processing behind a shared L2 cache. Through a set of private handshake signals, the Primary and Dual processors arbitrate for the external bus and maintain cache coherency between themselves. The bus arbitration and cache coherency mechanisms allow the Primary and Dual processors to look like a single Pentium processor to the external bus.

The Primary and Dual processors implement a fair arbitration scheme. If the Least Recent Master (LRM) requests the bus from the Most Recent Master (MRM), the bus is granted. The

Pentium processor arbitration scheme provides no penalty to switch from one master to the next. If pipelining is used, the two processors will pipeline into and out of each other's cycles according to the Pentium processor specification.

Cache coherency is maintained between the two processors by snooping on every bus access. The LRM must snoop with every ADS# assertion of the MRM. Internal cache states are maintained accordingly. If an access hits a modified line, a write back is scheduled as the next cycle in accordance with the Pentium processor specification.

Using the Dual processor may require special design considerations. Please refer to Chapter 4 for more details.

2.8.2.1. CONCEPTUAL OVERVIEW

Dual processing can be viewed in Figure 2-11.

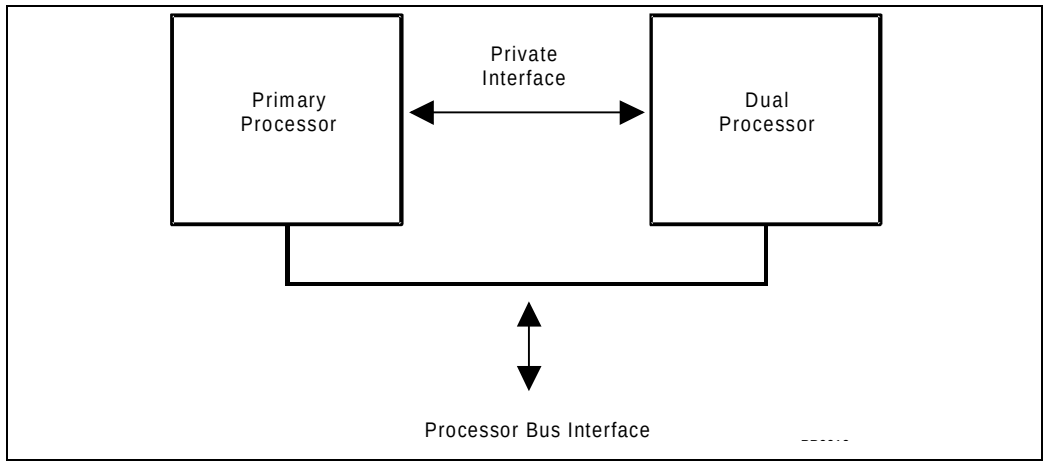


Figure 2-11. Dual Processors

The dual processor pair will appear to the system bus as a single, unified processor. The operation will be identical to a uni-processor Pentium processor, except as noted in Section 6.6. The interface shields the system designer from the cache consistency and arbitration mechanisms that are necessary for dual processor operation.

Both the Primary and Dual processors contain local APIC modules. The system designer is recommended to supply an I/O APIC or other multiprocessing interrupt controller in the chip set that interfaces to the local APIC blocks over a three-wire bus. The APIC allows directed interrupts as well as inter-processor interrupts.

The Primary and Dual processors, when operating in dual processing mode, require the local APIC modules to be hardware enabled in order to complete the bootup handshake protocol. This method is used to "wake up" the Dual processor at an address other than the normal Intel Architecture high memory execution address. On bootup, if the Primary processor detects that

a Dual processor is present, the dual processor cache consistency and arbitration mechanisms are automatically enabled. The bootup handshake process is supported in a protocol that is included in the Pentium processor. See Chapter 3 for more details on the APIC.

2.8.2.2. ARBITRATION OVERVIEW

In the dual processor configuration, there is a single-system bus which provides the processors access to the external system. This bus is a single, shared resource.

The dual processor pair will need to arbitrate for use of the system bus as requests are generated. The processors implement a fair arbitration mechanism.

If the LRM processor needs to run a cycle on the bus it will submit a request for bus ownership to the MRM. The MRM processor will grant the LRM processor bus ownership as soon as all outstanding bus requests have finished on the processor bus. The LRM processor will assume the MRM state, and the processor which was just the MRM, will become the LRM. Figure 2-12 further illustrates this point:

Diagram (a) of Figure 2-12 shows a configuration where the Primary processor is in the MRM state and the Dual processor is in the LRM state. The Primary processor is running a cycle on the system bus when it receives a bus request from the Dual processor. In diagram (b) of Figure 2-12 the MRM (still the Primary processor) has received an indication that the bus request has finished. The bus ownership has transferred in diagram (c) of Figure 2-12, where the Dual processor is now the MRM. At this point, the Dual processor will start a bus transaction and continue to own the bus until the LRM requests the bus.

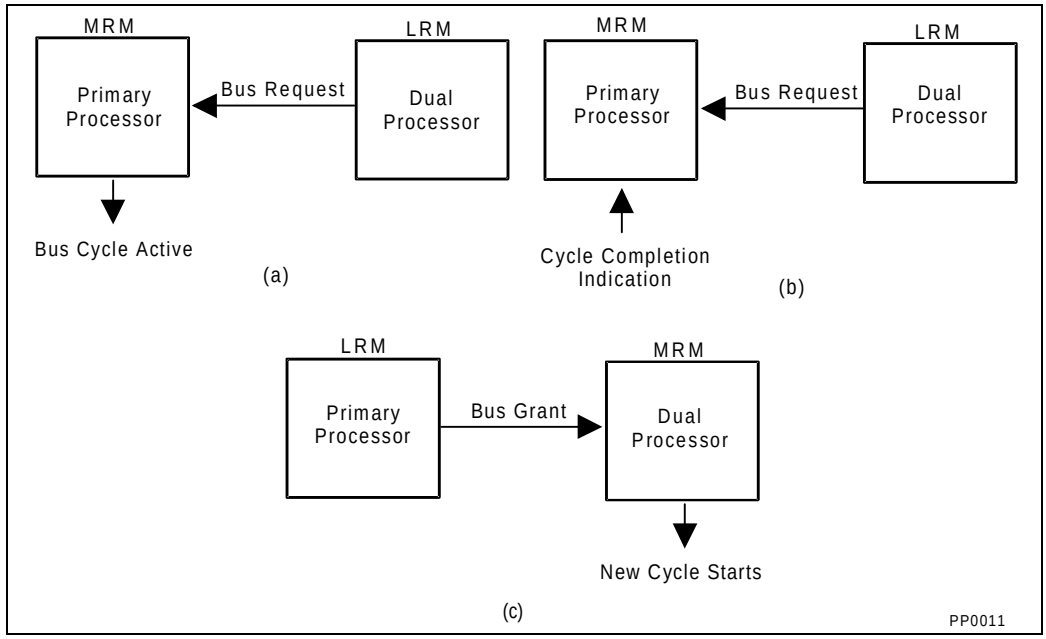


Figure 2-12. Dual Processor Arbitration Mechanism

2.8.2.3. CACHE COHERENCY OVERVIEW

The Primary and Dual processors both contain separate code and data caches. The data cache uses the MESI protocol to enforce cache consistency. A line in the data cache can be in the Modified, Exclusive, Shared or Invalid state, whereas a line in the instruction cache can be either in the valid or invalid state.

A situation can arise where the Primary and Dual processors are operating in dual processor mode with shared code or data. The first level caches will attempt to cache this code and data whenever possible (as indicated by the page cacheability bits and the cacheability pins). The private cache coherency mechanism guarantees data consistency across the processors. If any data is cached in one of the processors, and the other processor attempts to access the data, the processor containing the data will notify the requesting processor that it has cached the data. The state of the cache line in the processor containing the data will change depending on the current state and the type of request that the other processor has made.

In some cases the data returned by the system will be ignored. This constraint is placed on the dual processor cache consistency mechanism so that the dual processor pair will look like a single processor to the system bus. However, in general, bus accesses are minimized to efficiently use the available bus bandwidth.

The basic coherency mechanism requires the processor that is in the LRM state to snoop all MRM bus activity. The MRM processor running a bus cycle will watch the LRM processor

for an indication that the data is contained in the LRM cache. The following diagrams illustrate the basic coherency mechanism.

The series of following figures show an example where the Primary processor (the MRM) is performing a cache line fill of data. In this example, the data requested by the Primary processor is cached by the Dual processor (the LRM), and is in the modified state.

In diagram (a) of Figure 2-13, the Primary processor has already negotiated with the Dual processor for use of the system bus and started a cycle. As the Primary processor starts running the cycle on the system bus, the Dual processor snoops the transaction. The key for the start of the snoop sequence for the LRM processor is an assertion of ADS# by the MRM processor.

Diagram (b) of Figure 2-13 shows the Dual processor indicating to the Primary processor that the requested data is cached and modified in the Dual processor cache. The snoop notification mechanism uses a dedicated, two-signal interface that is private to the dual processor pair. At the same time that the Dual processor indicates that the transaction is contained as Modified in the its cache, the Dual processor will request the bus from the Primary processor (still the MRM). The MRM processor continues with the transaction that is outstanding on the bus, but will ignore the data returned by the system bus.

After the Dual processor notifies the Primary processor that the requested data is modified in the Dual processor cache, the Dual processor will wait for the bus transaction to complete. At this point, the LRM/MRM state will toggle, with the Primary processor becoming the LRM processor and the Dual processor becoming the MRM processor. This sequence of events is shown in diagram (c) of Figure 2-13.

Diagram (c) of Figure 2-13 also shows the Dual processor writing the data back on the system bus. The write back cycle will look like a normal cache line replacement to the system bus. The final state of the line in the Dual processor is determined by the value of the W/R# pin as sampled during the ADS# assertion by the Primary processor.

Finally, diagram (d) of Figure 2-13 shows the Primary processor re-running the bus transaction that started the entire sequence. The requested data will be returned by the system as a normal line fill request without intervention from the LRM processor.

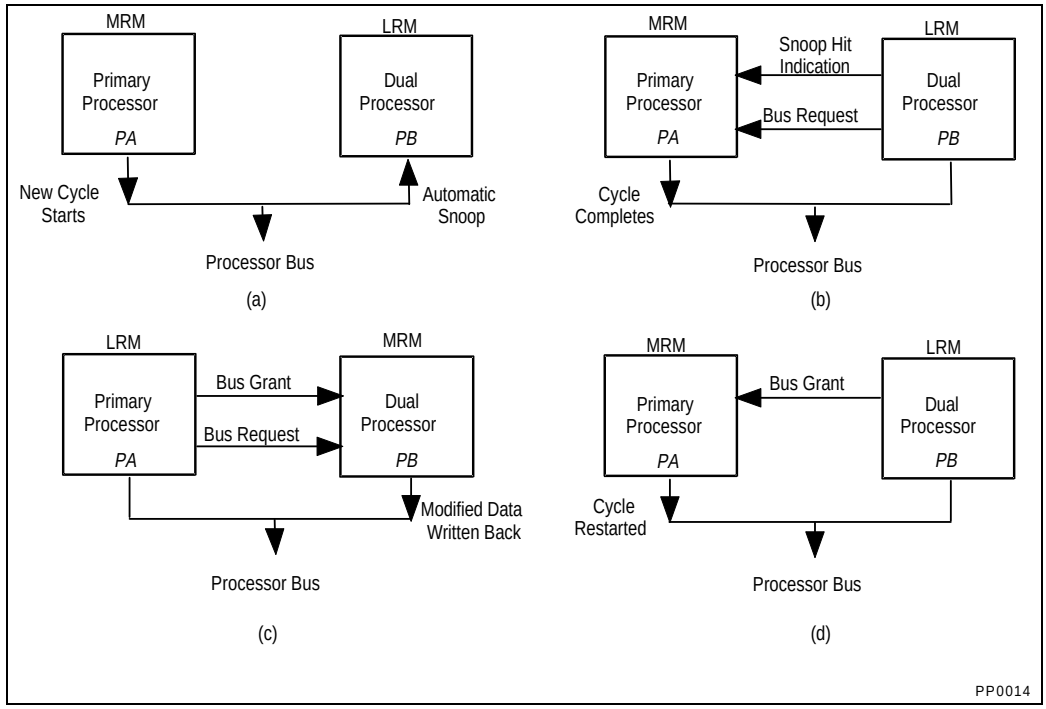


Figure 2-13. Dual Processor L1 Cache Consistency

2.9. APIC INTERRUPT CONTROLLER

The Pentium processor contains implementations of the Advanced Programmable Interrupt Controller architecture. These implementations are capable of supporting a multiprocessing interrupt scheme with an external APIC-compatible controller.

The Advanced Programmable Interrupt Controller (APIC) is an on-chip interrupt controller that supports multiprocessing. In a uniprocessor system, the APIC may be used as the sole system interrupt controller, or may be disabled and bypassed completely.

In a multiprocessor system, the APIC operates with an additional and external I/O APIC system interrupt controller. The dual-processor configuration requires that the APIC be hardware enabled. The APIC of the Primary and Dual processors are used in the bootup procedure to communicate startup information.

NOTE

The APIC is not hardware compatible with the 82489DX.

On the Pentium processor, the APIC uses 3 pins: PICCLK, PICD0, and PICD1. PICCLK is the APIC bus clock while PICD0-1 form the two-wire communication bus.

To use the 8259A interrupt controller, or to completely bypass, the APIC may be disabled using the APICEN pin. You must use the local APICs when using the dual-processor component.

The main features of the APIC architecture include:

- Multiprocessor interrupt management (static and dynamic symmetric interrupt distribution across all processors)
- Dynamic interrupt distribution that includes routing interrupts to the lowest-priority processor
- Inter-processor interrupt support
- Edge or level triggered interrupt programmability
- Various naming/addressing schemes
- System-wide processor control functions related to NMI, INIT, and SMI (see Chapter 14 for APIC handling of SMI)
- 8259A compatibility by becoming virtually transparent with regard to an externally connected 8259A style controller, making the 8259A visible to software
- A 32-bit wide counter used as a timer to generate time slice interrupts local to that processor.

The AC timings of the Pentium processor APIC are described in Chapter 7 of this document. Note that while there are minor software differences from the 82489DX, programming to the integrated APIC model ensures compatibility with the external 82489DX. For additional APIC programming information, please refer to the *MultiProcessor Specification*, Order Number 242016.

In a dual-processor configuration, the local APIC may be used with an additional device similar to the I/O APIC. The I/O APIC is a device which captures all system interrupts and directs them to the appropriate processors via various programmable distribution schemes. An external device provides the APIC system clock. Interrupts which are local to each processor go through the APIC on each chip. A system example is shown in Figure 2-14.

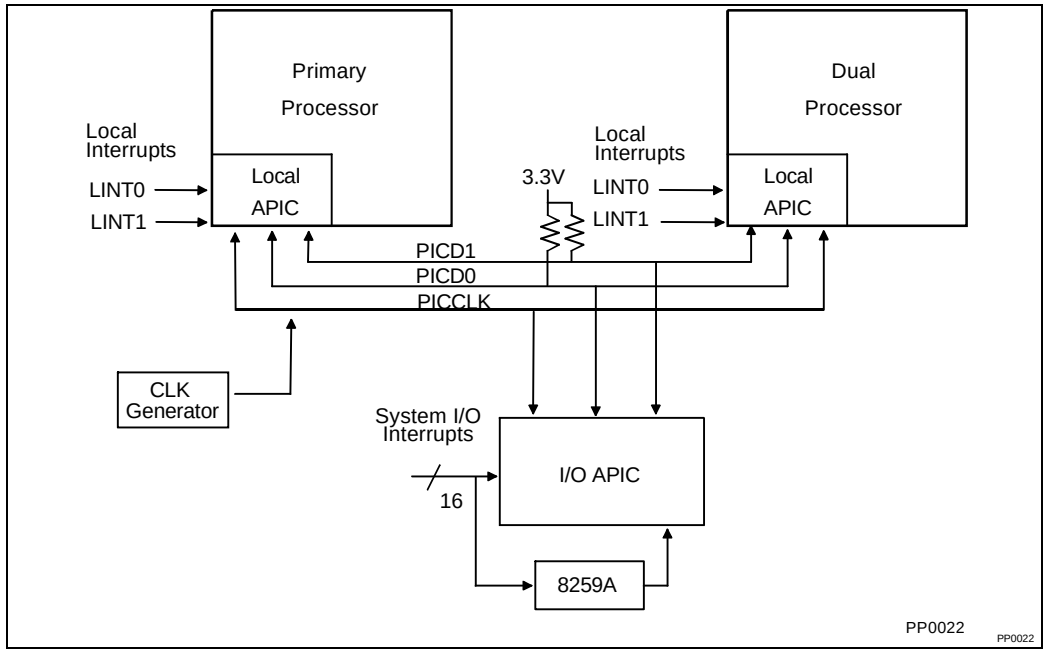


Figure 2-14. APIC System Configuration

The APIC devices in the Primary and Dual processors may receive interrupts from the I/O APIC via the three-wire APIC bus, locally via the local interrupt pins (LINT0, LINT1), or from the other processor via the APIC bus. The local interrupt pins, LINT0 and LINT1, are shared with the INTR and NMI pins, respectively. When the APIC is bypassed (hardware disabled) or programmed in "through local" mode, the 8259A interrupt (INTR) and NMI are connected to the INTR/LINT0 and NMI/LINT1 pins of the processor. Figure 2-15 shows the APIC implementation in the Pentium processor. Note that the PICCLK has a maximum frequency of 16.67 MHz.

When the local APIC is hardware enabled, *data* memory accesses to its 4 Kbyte address space are executed internally and do not generate an ADS# on the processor bus. However, a *code* memory access in the 4 KByte APIC address space will not be recognized by the APIC and will generate a cycle on the processor bus.

NOTE

Internally executed data memory accesses may cause the address bus to toggle even though no ADS# is issued on the processor bus.

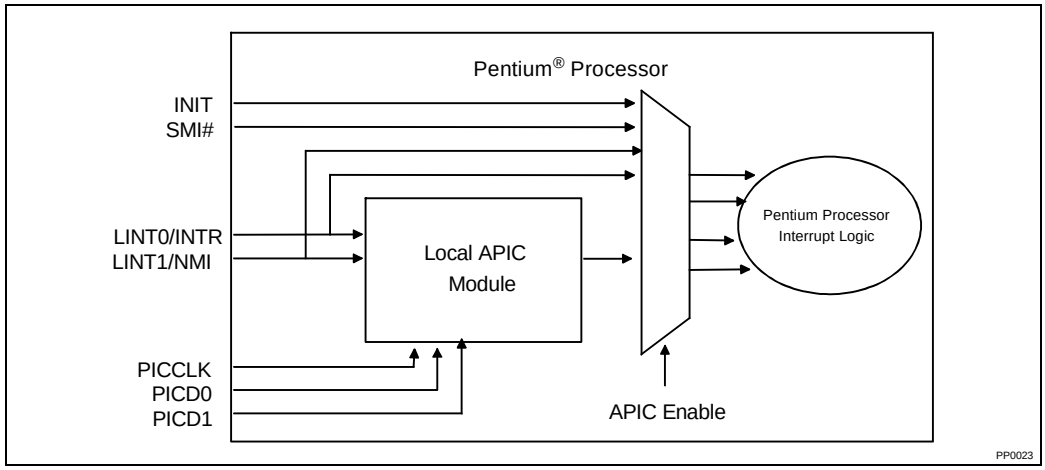


Figure 2-15. Local APIC Interface

2.9.1. APIC Configuration Modes

There are four possible APIC Modes:

1. Normal mode
2. Bypass mode (hardware disable)
3. Through local mode
4. Masked mode (software disable)

2.9.1.1. NORMAL MODE

This is the normal operating mode of the local APIC. When in this mode, the local APIC is both hardware and software enabled.

2.9.1.2. BYPASS MODE

Bypass mode effectively removes (bypasses) the APIC from the Pentium processor causing it to operate as if there were no APIC present. Any accesses to the APIC address space will go to memory. APICEN is sampled at the falling edge of RESET, and later becomes the PICD1 (part of the APIC 3-wire bus) signal. Bypass mode is entered by driving APICEN low at the falling edge of RESET. Since the APIC must be used to enable the Dual processor after RESET, PICD1 must be driven high at reset to ensure APIC is hardware enabled if a second processor is present.

For hardware disabling operations, the following implications must be considered:

1. The INTR and NMI pins become functionally equivalent to the corresponding interrupt pins in the Pentium processor, and the APIC is bypassed.
2. The APIC PICCLK must be tied high.
3. The system will not operate with the Dual Processor if the local APIC is hardware disabled.

2.9.1.3. THROUGH LOCAL MODE

Configuring in through local mode allows the APICs to be used for the dual-processor bootup handshake protocol and then pass interrupts through the local APIC to the core to support an external interrupt controller.

To use the through local mode of the local APIC, the APIC must be enabled in both hardware and software. This is done by programming two local vector table entries, LVT1 and LVT2 at addresses 0FEE00350H and 0FEE00360H, as external interrupts (ExtInt) and NMI, respectively. The 8259A responds to the INTA cycles and returns the interrupt vector to the processor.

The local APIC should not be sent any interrupts prior to it's being programmed. Once the APIC is programmed it can receive interrupts.

Note that although external interrupts and NMI are passed through the local APIC to the core, the APIC can still receive messages on the APIC bus.

2.9.1.4. MASKED MODE

The local APIC is initialized to masked mode once hardware enabled via the APICEN pin. In order to be programmed in normal or through local modes, the APIC must be "software enabled." Once operating in normal mode or through local mode, the APIC may be disabled by software through clearing bit 8 of the APIC's spurious vector interrupt register (Note: this register is normally cleared at RESET and INIT). This register is at address 0FEE000F0H. Disabling APIC in software will return it to Masked mode. With the exception of NMI, SMI, INIT, remote reads and the startup IPI, all interrupts are masked on the APIC bus. The local APIC does not accept any interrupts on LINT0 or LINT1. See the following section for software disabling implications.

2.9.1.4.1. Software Disabling Implications

For the software disabling operations, the following implications must be considered:

1. The 4 Kbyte address space for the APIC is always blocked for data accesses (i.e., external memory in this region must not be accessed).
2. The interrupt control register (ICR) can be read and written (e.g. interprocessor interrupts are sent by writing to this register).
3. The APIC can continue to receive SMI, NMI, INIT, "startup," and remote read messages.
4. Local interrupts are masked.

5. Software can enable/disable the APIC at any time. After software disabling the local APICs, pending interrupts must be handled or masked by software.
6. The APIC PICCLK must be driven at all times.

2.9.1.5. DUAL PROCESSING WITH THE LOCAL APIC

The Dual processor bootup protocol may be used in the normal, through local, or masked modes.

2.9.2. Loading the APIC ID

Loading the APIC ID may be done with external logic that would drive the proper address at reset. If the BE[3:0]# signals are not driven and do not have external resistors to V_{CC} or V_{SS}, the APIC ID value will default to 0000 for the Primary processor and 0001 for the Dual processor.

Table 2-8. APIC ID

APIC ID Register Bit	Pin Latched at RESET
bit 24	BE0#
bit 25	BE1#
bit 26	BE2#
bit 27	BE3#

WARNING

An APIC ID of all 1s is an APIC special case (i.e., a broadcast) and must not be used. Since the Dual processor inverts the lowest order bit of the APIC ID placed on the BE pins, the value "1110" should also be avoided when operating in Dual Processing mode.

In a dual processor configuration, the OEM and Socket 5 should have the four BE pairs tied together. The OEM processor will load the value seen on these four pins at RESET. The dual processor will load the value seen on these pins and automatically invert bit 24 of the APIC ID Register. Thus the two processors will have unique APIC ID values.

In a general multi-processing system consisting of multiple Pentium processor, these pins must not be tied together so each local APIC can have unique ID values.

These four pins must be valid and stable two clocks before and after the falling edge of RESET.

2.9.3. Response to HOLD

While the Pentium processor is accessing the APIC, the processor will respond to a HOLD request with a maximum delay of six clocks. To external agents which are not aware of the APIC bus, this looks like the Pentium processor is not responding to HOLD even though ADS# has not been driven and the processor bus seems idle.

2.10. FRACTIONAL SPEED BUS

The Pentium processor is offered in various bus-to-core frequency ratios. The BF[1:0] configuration pins determine the bus-to-core frequency ratio. The processor will multiply the input CLK by the bus-to-core ratio to achieve higher internal core frequencies.

The external bus frequency is set on power-up RESET through the CLK pin. The Pentium processor will sample the BF[1:0] pins on the falling edge of RESET to determine which bus-to-core ratio to use. If the BF[1:0] pins are left unconnected, the Pentium processor (75/90/100/120/133/150/166/200) defaults to the 2/3 ratio and the Pentium processor with MMX technology defaults to the 1/2 ratio. **BF[1:0] must not change its value while RESET is active**. Changing the external bus speed or bus-to-core ratio requires a “power-on” RESET pulse initialization. Once a frequency is selected, it may not be changed with a warm-reset (15 clocks). The BF pin must meet a 1 ms setup time to the falling edge of RESET.

Each Pentium processor is specified to operate within a single bus-to-core ratio and a specific minimum to maximum bus frequency range (corresponding to a minimum to maximum core frequency range). Operation in other bus-to-core ratios or outside the specified operating frequency range is not supported. For example, the 150 MHz Pentium processor does not operate beyond the 60 MHz bus frequency and only supports the 2/5 bus-to-core ratio; it does not support the 1/3, 1/2, or 2/3 bus-to-core ratios. Table 2-9 clarifies and summarizes these specifications.

Table 2-9. Bus-to-Core Frequency Ratios for the Pentium® Processor

BF1	BF0	Pentium® Processor (75/90/100/120/133/150/166/200) Bus/Core Ratio	Pentium Processor with MMX™ Technology Bus/Core Ratio[4]	Max Bus/Core Frequency (MHz)	Min Bus/Core Frequency (MHz)
0	1	1/3	1/3	66/200	33/100
0	0	2/5	2/5	66/166	33/83
0	0	2/5	2/5	60/150	30/75
1	0	1/2	1/2 [2]	66/133	33/66
1	0	1/2	1/2 [2]	60/120	30/60
1	0	1/2	1/2 [2]	50/100[3]	25/50
1	1	2/3 [1]	reserved	66/100[3]	33/50
1	1	2/3 [1]	reserved	60/90	30/45
1	1	2/3 [1]	reserved	50/75	25/37.5

NOTES:

1. This is the default bus fraction for the Pentium® processor (75/90/100/120/133/150/166/200). If the BF pins are left floating, the processor will be configured for the 2/3 bus to core frequency ratio.
2. This is the default bus fraction for the Pentium processor with MMX™ technology. If the BF pins are left floating, the processor will be configured for the 1/2 bus to core frequency ratio.
3. The 100 MHz (Max Core Frequency) Pentium processors can be operated in both 1/2 and 2/3 Bus/Core Ratios.
4. Currently, the desktop Pentium processor with MMX technology supports 66/200 and 66/166 operation.

The following examples illustrate the Pentium processor synchronization mechanism. The Pentium processor (60, 66) case is given to illustrate how a 1/1 bus operates.

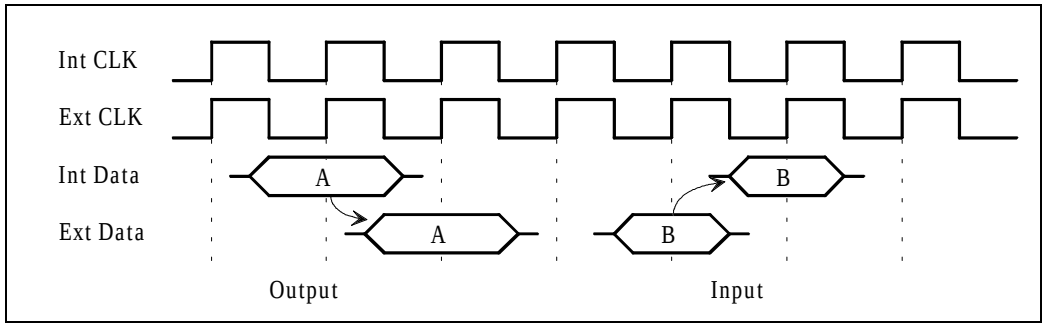


Figure 2-16. Pentium® Processor (60, 66) Synchronous Internal/External Data Movement

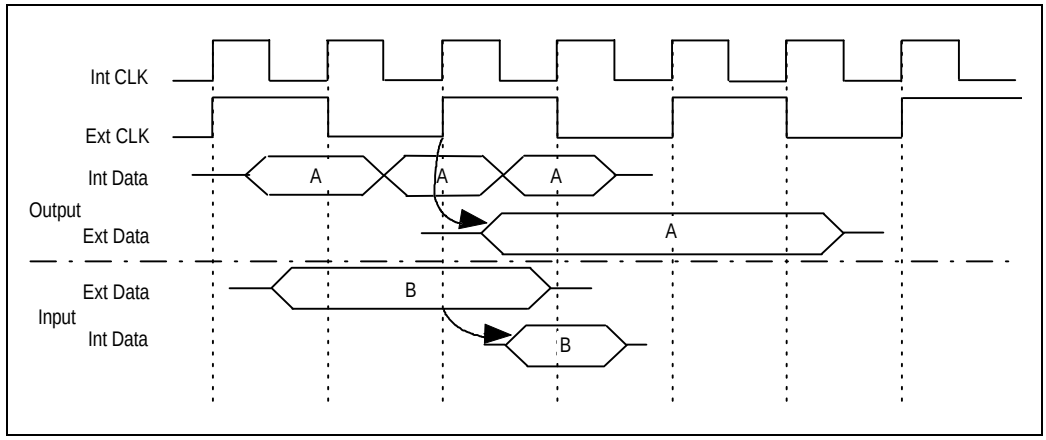


Figure 2-17. Pentium® Processor
1/2 Bus Internal/External Data Movement

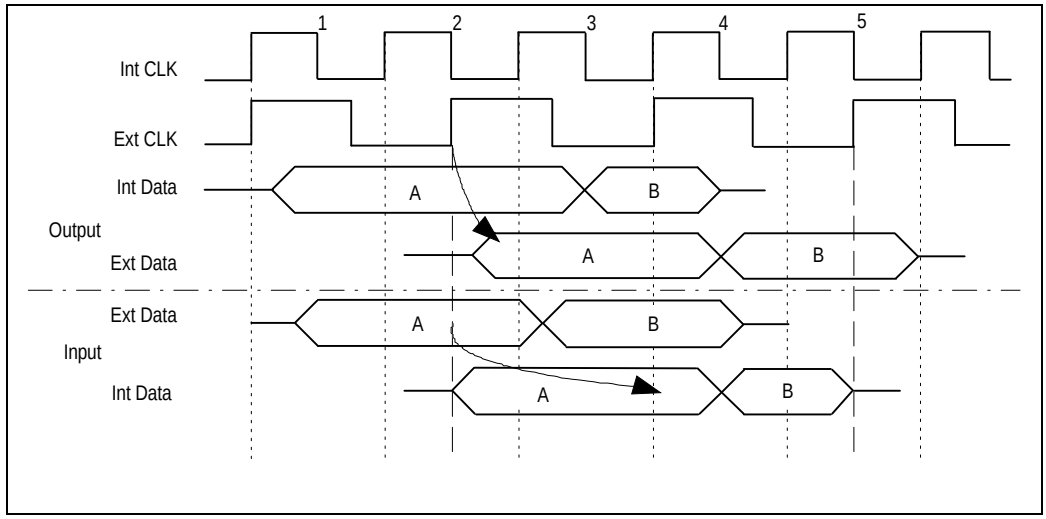


Figure 2-18 Pentium® Processor
2/3 Bus Internal/External Data Movement

Figure 2-19 shows how the Pentium processor prevents data from changing in clock 2, where the 2/3 external clock rising edge occurs in the middle of the internal clock phase, so it can be properly synchronized and driven.

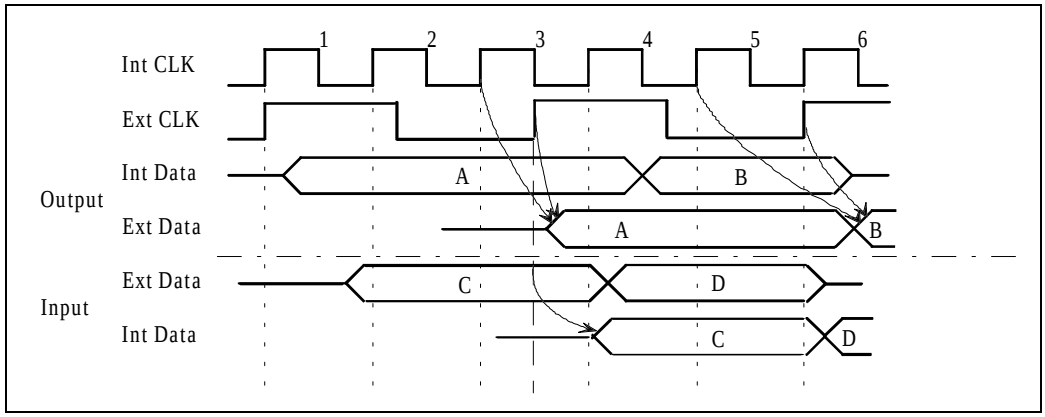


Figure 2-19. Pentium® Processor 2/5 Bus Internal/External Data Movement

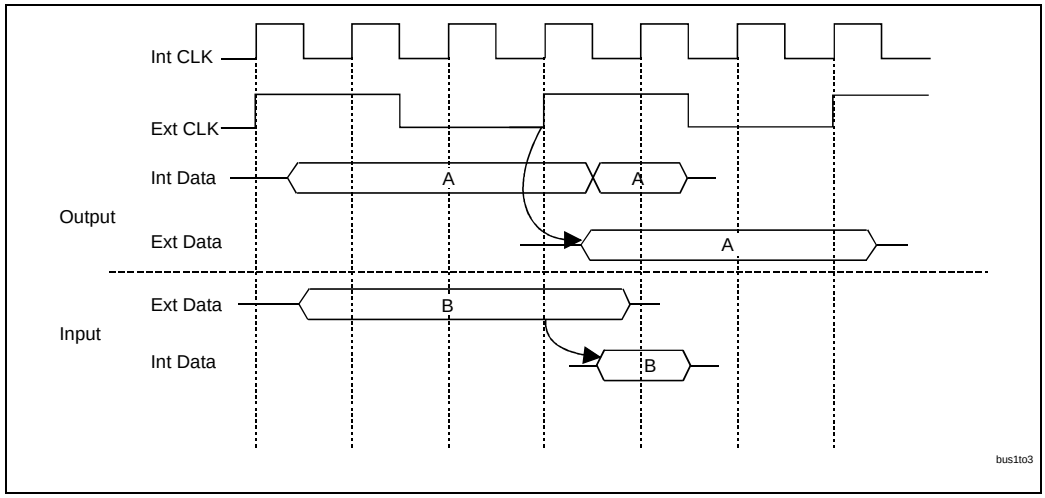


Figure 2-20. Pentium® Processor 1/3 Bus Internal/External Data Movement

2.11. POWER MANAGEMENT

2.11.1. I/O Instruction Restart

I/O Instruction restart is a power management feature of the Pentium processor that allows the Pentium processor to re-execute an I/O instruction. In this way, an I/O instruction can alert a

sleeping device in a system and SMI# can be recognized before the I/O instruction is re-executed. SMI# assertion will cause a wake-up routine to be executed, so the restarted I/O instruction can be executed by the system.

2.11.2. Stop Clock and AutoHalt Powerdown

The Pentium processor uses stop clock and AutoHalt Powerdown to immediately reduce the power of each device. These features cause the clock to be stopped to most of the CPU's internal units and thus significantly reduce power consumption by the CPU as a whole.

Stop clock is enabled by asserting the STPCLK# pin of the Pentium processor. While asserted, the Pentium processor will stop execution and not service interrupts, but will allow external and interprocessor (Primary and Dual processor) snooping.

AutoHalt Powerdown is entered once the Pentium processor executes a HLT instruction. In this state, most internal units are powered-down, but the Pentium processor will recognize all interrupts and snoops.

Pentium processor pin functions (D/P#, etc.) are not affected by STPCLK# or AutoHalt.

For additional details on power management, refer to Chapter 14.

2.12. CPUID INSTRUCTION

The CPUID instruction provides information to software about the vendor, family, model and stepping of the microporcessor on which it is executing. In addition, it indicates the features supported by the processor.

When executing CPUID:

- If the value in EAX is "0," then the 12-byte ASCII string "GenuineIntel" (little endian) is returned in EBX, EDX, and ECX. Also, EAX contains a value of "1" to indicate the largest value of EAX which should be used when executing CPUID.
- If the value in EAX is "1," then the processor version is returned in EAX and the processor capabilities (feature flags) are returned in EDX.
- If the value in EAX is neither "0" nor "1", the Pentium processor writes "0" to EAX, EBX, ECX, and EDX.

The following EAX value is defined for the CPUID instruction executed with EAX = 1. The processor version EAX bit assignments are given in Figure 2-21. Table 2-10 lists the feature flag bits assignment definitions.

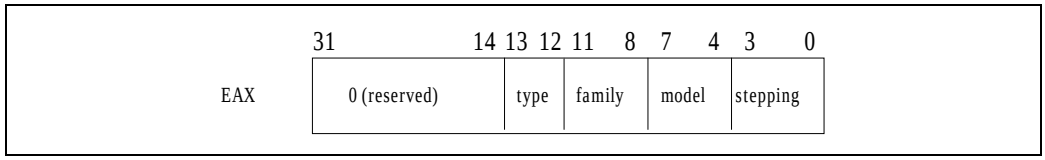


Figure 2-21. EAX Bit Assignments for CPUID

Table 2-10. EDX Bit Assignment Definitions (Feature Flags)

Bit	Name	Value	Description When Flag=1	Comments
0	FPU	1	Floating-point unit on-chip	The processor contains an FPU that supports the Intel 387 floating-point instruction set.
1	VME	1	Virtual Mode Enhancements	The processor supports extensions to virtual-8086 mode.
2	DE	1	Debugging Extension	The processor supports I/O breakpoints, including the CR4.DE bit for enabling debug extensions and optional trapping of access to the DR4 and DR5 registers.
3	PSE	1	Page Size Extension	The processor supports 4-Mbyte pages.
4	TSC	1	Time Stamp Counter	The RDTSC instruction is supported including the CR4.TSD bit for access/privilege control.
5	MSR	1	Pentium® Processor MSR	Model SpecificRegisters are implemented with the RDMSR, WRMSR instructions.
6	PAE	0	Physical Address Extension	Physical addresses greater than 32 bits are supported.
7	MCE	1	Machine Check Exception	Machine Check Exception, Exception 18, and the CR4.MCE enable bit are supported.
8	CX8	1	CMPXCHG8B Instruction Supported	The compare and exchange 8 bytes instruction is supported.
9	APIC	1	On-chip APIC Hardware Enabled*	The processor contains a local APIC.
10-11		R	Reserved	Do not rely on its value.
12	MTRR	0	Memory Type Range Registers	The processor supports the Memory Type Range Registers specifically the MTRR_CAP register.
13	PGE	0	Page Global Enable	The global bit in the PDE's and PTE's and the CR4.PGE enable bit are supported.
14	MCA	0	Machine Check Architecture	The Machine Check Architecture is supported, specifically the MCG_CAP register.
15-22		R	Reserved	Do not rely on its value.
23	MMX technology	1	Intel Architecture MMX™ technology supported	The processor supports the MMX technology instruction set extensions to the Intel Architecture.
24-31		R	Reserved	Do not rely on its value.

* Indicates that APIC is present and hardware enabled (software disabling does not affect this bit).

The family field for the Pentium processor family is 0101 (5H). The model value for the Pentium processor (75/90/100/120/133/150/166/200) is 0010 (2H) or 0111 (7H), and the model value for the Pentium processor with MMX technology is 0100 (4H).

NOTE

Use the MMX technology feature bit (bit 23) in the EFLAGS register, not the model value, to detect the presence of the MMX technology feature set.

For specific information on the stepping field, consult the Pentium processor family Specification Update. The future Pentium OverDrive processor will not have the same values in EAX as the Pentium processor (75/90/100/120/133/150/166/200) or the Pentium processor with MMX technology. The type field is defined in Table 2-11.

Table 2-11. EAX Type Field Values

Bit 13	Bit 12	Processor Type
0	0	Pentium® processor (75/90/100/120/133/150/166/200) or Pentium processor with MMX™ technology.
0	1	Future Pentium OverDrive® processor
1	0	Dual Pentium processor
1	1	Reserved

2.13. MODEL SPECIFIC REGISTERS

Each Pentium processor contains certain Model Specific Registers that are used in execution tracing, performance monitoring, testing, and machine check errors. They are unique to that Pentium processor and may not be implemented in the same way in future processors.

Two instructions, RDMSR and WRMSR (read/write model specific registers) are used to access these registers. When these instructions are executed, the value in ECX specifies which model specific register is being accessed.

Software must not depend on the value of reserved bits in the model specific registers. Any writes to the model specific registers should write “0” into any reserved bits.

For information on Model Specific Registers and Functions, refer to Chapter 16 of this document.

3

Microprocessor Initialization and Configuration

CHAPTER 3
MICROPROCESSOR INITIALIZATION AND CONFIGURATION

This chapter covers microprocessor initialization and configuration information for both uni-processor and dual-processor implementations of the Pentium processor family. For configuration information on symmetric dual-processing mode, refer to section 3.4.

Before normal operation of the Pentium processor can begin, the Pentium processor must be initialized by driving the RESET pin active. The RESET pin forces the Pentium processor to begin execution in a known state. Several features are optionally invoked at the falling edge of RESET: Built-in-Self-Test (BIST), Functional Redundancy Checking and Tristate Test Mode.

In addition to the standard RESET pin, the Pentium processor has implemented an initialization pin (INIT) that allows the processor to begin execution in a known state without disrupting the contents of the internal caches or the floating-point state.

This chapter describes the Pentium processor power up and initialization procedures as well as the test and configuration features enabled at the falling edge of RESET.

3.1. POWER UP SPECIFICATIONS

During power up, RESET must be asserted while V_{CC} is approaching nominal operating voltage to prevent internal bus contention which could negatively affect the reliability of the processor.

It is recommended that CLK begin toggling within 150 ms after V_{CC} reaches its proper operating level. For Pentium processors with MMX technology, it is recommended that the CLK signal should be toggling within 150 ms after the last Vcc plane stabilizes. This recommendation is only to ensure long term reliability of the device.

In order for RESET to be recognized, the CLK input needs to be toggling. RESET must remain asserted for 1 millisecond after V_{CC} and CLK have reached their AC/DC specifications.

3.2. TEST AND CONFIGURATION FEATURES (BIST, FRC, TRISTATE TEST MODE)

The INIT, FLUSH#, and FRCMC# inputs are sampled when RESET transitions from high to low to determine if BIST will be run, or if tristate test mode or checker mode will be entered (respectively).

If RESET is driven synchronously, these signals must be at their valid level and meet setup and hold times on the clock before the falling edge of RESET. If RESET is asserted asynchronously, these signals must be at their valid level two clocks before and after RESET transitions from high to low.

3.2.1. Built In Self-Test

Self-test is initiated by driving the INIT pin high when RESET transitions from high to low.

No bus cycles are run by the Pentium processor during self test. The duration of self test is approximately 2^{19} core clocks. Approximately 70% of the devices in the Pentium processor are tested by BIST.

The Pentium processor BIST consists of two parts: hardware self-test and microcode self-test.

During the hardware portion of BIST, the microcode ROM and all large PLAs are tested. All possible input combinations of the microcode ROM and PLAs are tested.

The constant ROMs, BTB, TLBs, and all caches are tested by the microcode portion of BIST. The array tests (caches, TLBs and BTB) have two passes. On the first pass, data patterns are written to arrays, read back and checked for mismatches. The second pass writes the complement of the initial data pattern, reads it back, and checks for mismatches. The constant ROMs are tested by using the microcode to add various constants and check the result against a stored value.

Upon successful completion of BIST, the cumulative result of all tests are stored in the EAX register. If EAX contains 0h, then all checks passed; any non-zero result indicates a faulty unit. Note that if an internal parity error is detected during BIST, the processor will assert the IERR# pin and attempt to shutdown.

3.2.2. Tristate Test Mode

When the FLUSH# pin is sampled low when RESET transitions from high to low, the Pentium processor enters tristate test mode. The Pentium processor floats all of its output pins and bi-directional pins including pins which are never floated during normal operation (except TDO). Tristate test mode can be initiated in order to facilitate testing board interconnects. The Pentium processor remains in tristate test mode until the RESET pin is asserted again.

3.2.3. Functional Redundancy Checking

The functional redundancy checking master/checker configuration input is sampled when RESET is high to determine whether the Pentium processor is configured in master mode (FRCMC# high) or checker mode (FRCMC# low). Note, the Pentium processor with MMX technology does not support FRC mode.

The final master/checker configuration of the Pentium processor is determined the clock before the falling edge of RESET. When configured as a master, the Pentium processor drives its output pins as required by the bus protocol. When configured as a checker, the Pentium processor tristates all outputs (except IERR#, PICD0, PICD1 and TDO) and samples the output pins (that would normally be driven in master mode). If the sampled value differs from the value computed internally, the Pentium processor asserts IERR# to indicate an error. Note that IERR# will not be asserted due to an FRC mismatch until two clocks after the ADS# of the first bus cycle (or in the third clock of the bus cycle).

FINIT/FNINIT must be used to initialize the FPU state prior to using FSAVE/FNSAVE in FRC mode to avoid an FRC error caused by differences in the unitialized FPU state. The initialization should be done before other FPU activity so that it does not corrupt the previous state.

3.2.4. Lock Step APIC Operation

Lock Step operation is entered by holding BE4# high during the falling edge of RESET. Lock Step operation is not supported by the Pentium processor with MMX technology.

Lock Step operation guarantees recognition of an interrupt on a specific clock by two processors operating together that are using the APIC as the interrupt controller. This functionality is related to FRC operation, but FRC on the APIC pins is not fully supported in this way. There is no FRC comparator on the APIC pins, but mismatches on these pins will result in a mismatch on other pins of the processor.

Fault tolerant systems implemented with multiple processors running identical code sequences and generating identical bus cycles on all clocks may utilize Lock Step operation.

Setup and Hold time specifications PICCLK (in relation to CLK) are added for this funtionality. Additionally, there is a requirement to sustain specific integer ratios between the frequencies of PICCLK and CLK. This ratio should support both the maximum bus frequency of the device as well as the maximum frequency of PICCLK. Details of these specifications can be found in Chapter 7 of this manual.

3.3. INITIALIZATION WITH RESET, INIT AND BIST

Two pins, RESET and INIT, are used to reset the Pentium processor in different manners. A “cold” or “power on” RESET refers to the assertion of RESET while power is initially being applied to the Pentium processor. A “warm” RESET refers to the assertion of RESET or INIT while V_{CC} and CLK remain within specified operating limits.

Table 3-1 shows the effect of asserting RESET and/or INIT.

Table 3-1. Pentium® Processor Reset Modes

RESET	INIT	BIST Run?	Effect on Code and Data Caches	Effect on FP Registers	Effect on BTB and TLBs
0	0	No	n/a	n/a	n/a
0	1	No	None	None	Invalidated
1	0	No	Invalidated	Initailized	Invalidated
1	1	Yes	Invalidated	Initialized	Invalidated

Toggling either the RESET pin or the INIT pin individually forces the Pentium processor to begin execution at address FFFFFFF0h. The internal instruction cache and data cache are invalidated when RESET is asserted (modified lines in the data cache are NOT written back). The instruction cache and data cache are not altered when the INIT pin is asserted without RESET. In both cases, the branch target buffer (BTB) and translation lookaside buffers (TLBs) are invalidated.

After RESET (with or without BIST) or INIT, the Pentium processor will start executing instructions at location FFFFFFF0H. When the first Intersegment Jump or Call instruction is executed, address lines A20-A31 will be driven low for CS-relative memory cycles and the Pentium processor will only execute instructions in the lower one Mbyte of physical memory. This allows the system designer to use a ROM at the top of physical memory to initialize the system.

RESET is internally hardwired and forces the Pentium processor to terminate all execution and bus cycle activity within 2 clocks. No instruction or bus activity will occur as long as RESET is active. INIT is implemented as an edge triggered interrupt and will be recognized when an instruction boundary is reached. As soon as the Pentium processor completes the INIT sequence, instruction execution and bus cycle activity will continue at address FFFFFFF0h even if the INIT pin is not deasserted.

At the conclusion of RESET (with or without self-test) or INIT, the DX register will contain a component identifier. The upper byte will contain 05h and the lower byte will contain a stepping identifier.

Table 3-2 defines the processor state after RESET, INIT and RESET with BIST (built in self-test).

Table 3-2. Register State after RESET, INIT and BIST

Storage Element	RESET (No BIST)	RESET (BIST)	INIT
EAX	0	0 if pass	0
EDX	0500+stepping	0500+stepping	0500+stepping
ECX, EBX, ESP, EBP, ESI, EDI	0	0	0
EFLAGS	2	2	2
EIP	0FFF0	0FFF0	0FFF0

Table 3-2. Register State after RESET, INIT and BIST (Contd.)

Storage Element	RESET (No BIST)	RESET (BIST)	INIT
CS	selector = F000	selector = F000	selector = F000
	AR = P, R/W, A	AR = P, R/W, A	AR = P, R/W, A
	base = FFFF0000	base = FFFF0000	base = FFFF0000
	limit = FFFF	limit = FFFF	limit = FFFF
DS, ES, FE, GS, SS	selector = 0	selector = 0	selector = 0
	AR = P, R/W, A	AR = P, R/W, A	AR = P, R/W, A
	base = 0	base = 0	base = 0
	limit = FFFF	limit = FFFF	limit = FFFF
(I/G/L)DTR, TSS	selector = 0	selector = 0	selector = 0
	base = 0	base = 0	base = 0
	AR = P, R/W	AR = P, R/W	AR = P, R/W
	limit = FFFF	limit = FFFF	limit = FFFF
CR0	60000010	60000010	Note 2
CR2, 3, 4	0	0	0
DR3-0	0	0	0
DR6	FFFF0FF0	FFFF0FF0	FFFF0FF0
DR7	00000400	00000400	00000400
Time Stamp Counter	0	0	Unchanged
Control and Event Select	0	0	Unchanged
TR12	0	0	Unchanged
All other MSR's	Undefined	Undefined	Unchanged
CW	0040	0040	Unchanged
SW	0	0	Unchanged
TW	5555	5555	Unchanged
FIP, FEA, FCS, FDS, FOP	0	0	Unchanged
FSTACK	0	0	Unchanged
SMBASE	30000	30000	Unchanged
Data and Code Cache	Invalid	Invalid	Unchanged
Code Cache TLB, Data Cache TLB, BTB, SDC	Invalid	Invalid	Invalid

NOTES:

1. Register States are given in hexadecimal format.
2. CD and NW are unchanged, bit 4 is set to 1, all other bits are cleared.

3.3.1. Recognition of Interrupts after RESET

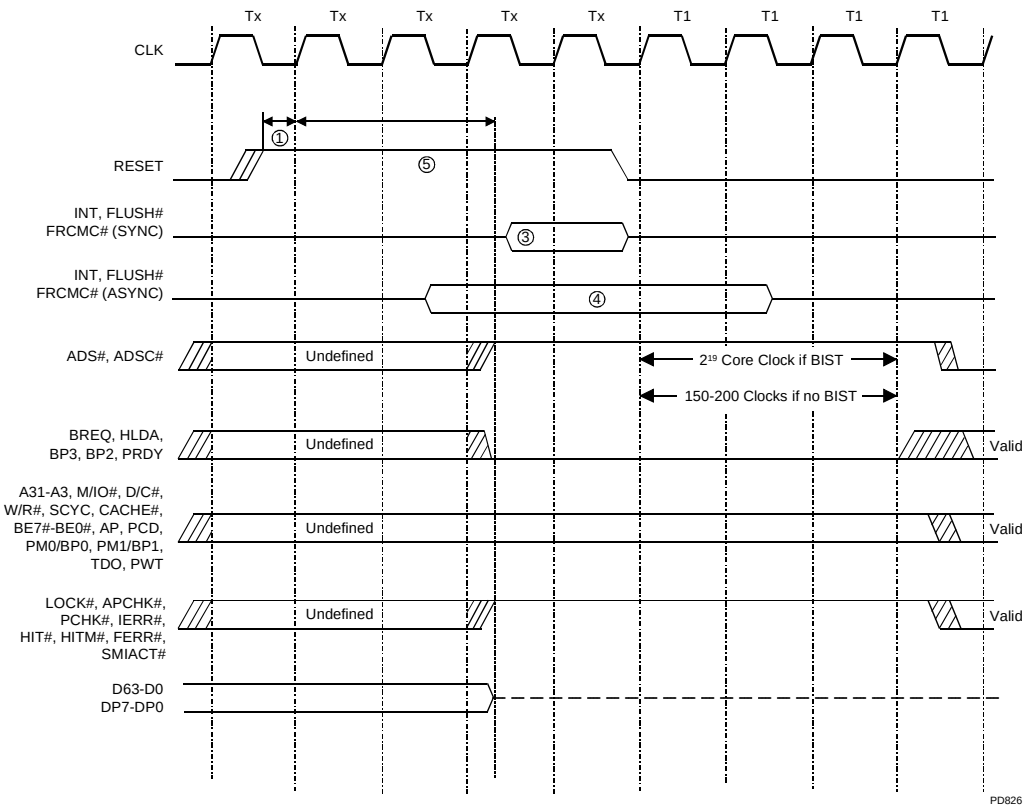
In order to guarantee recognition of the edge sensitive interrupts (FLUSH#, NMI, R/S#, SMI#) after RESET or after RESET with BIST, the interrupt input must not be asserted until four clocks after RESET is deasserted, regardless of whether BIST is run or not.

3.3.2. Pin State during/after RESET

The Pentium processor recognizes and will respond to HOLD, AHOLD and BOFF# during RESET. Figure 3-1 shows the processor state during and after a power on RESET if HOLD, AHOLD, and BOFF# are inactive. Note that the address bus (A31-A3, AP, BE7#-BE0#) and cycle definition pins (M/IO#, D/C#, W/R#, CACHE#, SCYC, PCD, PWT, PM0/BP0, PM1/BP1 and LOCK#) are undefined from the time RESET is asserted until the start of the first bus cycle.

The following lists the state of the output pins after RESET assuming HOLD, AHOLD and BOFF# are inactive, boundary scan is not invoked, and no internal parity error is detected.

- High: LOCK#, ADS#, ADSC#, APCHK#, PCHK#, IERR#, HIT#, HITM#, FERR#, SMIACT#
- Low: HLDA, BREQ, BP3, BP2, PRDY
- High Impedance: D63-D0, DP7-DP0
- Undefined: A31-A3, AP, BE7#-BE0#, W/R#, M/IO#, D/C#, PCD, PWT, CACHE#, TDO, SCYC, PM0/BP0, PM1/BP1



NOTES:

1. RESET must meet setup and hold times to guarantee recognition on a specific clock edge. If RESET does not need to be recognized on a specific clock edge it may be asserted asynchronously.
2. At power up, RESET needs to be asserted for 1 ms after V_{CC} and CLK have reached their AC/DC specifications. For warm reset, RESET needs to be asserted for at least 15 clocks while V_{CC} and CLK remain within specified operating limits.
3. If RESET is driven synchronously, FLUSH#, FRCMC# and INIT must be at their valid level and meet setup and hold times to the clock before the falling edge of RESET.
4. If RESET is driven asynchronously, FLUSH#, FRCMC# and INIT must be at their valid level two clocks before and after the falling edge of RESET.
5. An assertion of RESET takes at least two clocks to affect the pins.

Figure 3-1. Pin States during RESET

3.4. MANAGING AND DESIGNING WITH THE SYMMETRICAL DUAL PROCESSING CONFIGURATION

3.4.1. Dual Processor Bootup Protocol

3.4.1.1. BOOTUP OVERVIEW

Systems using the Pentium processor may be equipped with a second processor socket. For correct system operation, the Pentium processor must be able to identify the presence and type of the second processor (a Dual processor or a future Pentium OverDrive processor). Furthermore, since upgrade processors will typically be installed in the field by end users, system configuration may change between any two consecutive power-down/up sequences. The system must therefore have a mechanism to ascertain the system configuration during boot time. The boot up handshake protocol provides this mechanism.

3.4.1.2. BIOS / OPERATING SYSTEM REQUIREMENTS

The BIOS or HAL (hardware abstraction layer) of the operating system software should be generic, independent of the kind of OEM or upgrade processor present in the system. BIOS/HAL are specific to the system hardware, and should not need any change when an upgrade processor is installed. For dual processors, if the BIOS is not DP-ready, it will be up to the operating system to initialize and configure the dual processor appropriately.

The CPUID instruction is used to deliver processor-specific information. The Pentium processor CPUID status has been extended to supply the processor type information which includes “turbo-upgrade” classification (“type” field: bits 13-12 = 0-1). For upgradability with a future Pentium OverDrive processor, system software must allow the type field of the EAX register following the CPUID instruction to contain the values for both the Pentium processor and the future Pentium OverDrive processor. Refer to Section 2.12 for details. Note also that the model field of the CPUID will change for the future Pentium OverDrive processor.

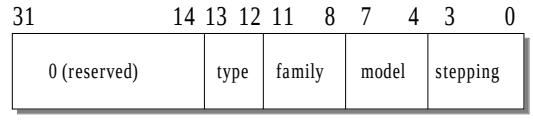


Figure 3-2. EAX Bit Assignments for CPUID

3.4.1.3. SYSTEM REQUIREMENTS

The number of Dual processors per Primary processor is limited to 1.

This bootup handshake protocol requires enabling the local APIC module using the APICEN pin. The startup IPI must be sent via the local APICs. Once the Dual processor has been initialized, software can later disable the local APIC module using several methods. These methods and their considerations are discussed in Section 2.9.

The protocol does not preclude more generic multiprocessing systems where multiple pairs of Pentium processor Primary and Dual processors may exist on the system bus.

3.4.1.4. START-UP BEHAVIOR

On RESET and INIT (message or pin), the Pentium processor begins execution at the reset vector (0FFFFFFF0H). The Dual processor waits for a startup IPI from the BIOS or operating system via the local APIC of the Pentium processor. The INIT IPI can be used to put the Pentium processor or Dual processor to sleep (since, once the INIT IPI is received, the CPU must wait for the startup IPI).

The startup IPI is specifically provided to start the Dual processor's execution from a location other than the reset vector, although it can be used for the Pentium processor as well. The startup IPI is sent by the system software via the local APIC by using a delivery mode of 110B. The startup IPI must include an 8-bit vector which is used to define the starting address. The starting address = *000 VV 000 h*, where *VV* indicates the vector field (in hex) passed through the IPI.

The 8-bit vector defines the address of a 4 Kbyte page in the Intel Architecture Real Mode Space (1 Mbyte space). For example, a vector of 0cdH specifies a startup memory address of 000cd000H. This value is used by the processor to initialize the segment descriptor for the upgrade's CS register as follows:

- The CS selector is set to the startup memory address/16 (real mode addressing)
- The CS base is set to the startup memory address
- The CS limit is set to 64 Kbytes
- The current privilege level (CPL) and instruction pointer (IP) are set to 0

NOTE

Vectors of 0A0H to 0BFH are reserved by Intel.

The benefit of the startup IPI is that it does not require the APIC to be software enabled (the APIC must be hardware enabled via the APICEN pin) and does not require the interrupt table to be programmed. Startup IPIs are non-maskable and can be issued at any time to the Pentium processor or Dual processor. If the startup IPI message is not preceded by a RESET or INIT (message or pin), it will be ignored.

It is the responsibility of the system software to resend the startup IPI message if there is an error in the IPI message delivery. Although the APIC need not be enabled in order to send the

startup IPI, the advantage to enabling the APIC prior to sending the startup IPI is to allow APIC error handling to occur via the APIC error handling entry of the local vector table (ERROR INT or LVT3 at APIC address 0FEE00370). Otherwise, the system software would have to poll the delivery status bit of the interrupt command register to determine if the IPI is pending (Bit 12 of the ICR=1) and resend the startup IPI if the IPI remains pending after an appropriate amount of time.

3.4.1.5. DUAL-PROCESSOR OR UPGRADE PRESENCE INDICATION

The bootup handshake protocol becomes aware that an additional processor is present through the DPEN# pin. The second processor is guaranteed to drive this signal low during RESETs falling edge. If the system needs to remember the presence of a second processor for future use, it must latch the state of the DPEN# pin during the falling edge of RESET.

3.4.2. Dual-Processor Arbitration

The Pentium processor incorporates a private arbitration mechanism that allows the Primary and Dual processors to arbitrate for the shared processor bus without assistance from a bus controller. The arbitration scheme is architected in such a way that the dual processor pair will appear as a single processor to the system.

The arbitration logic uses a fair arbitration scheme. The arbitration state machine was designed to efficiently use the processor bus bandwidth. In this spirit, the dual processor pair supports inter-CPU pipelining of most bus transactions. Furthermore, the arbitration mechanism does not introduce any dead clocks on bus transactions.

3.4.2.1. BASIC DUAL-PROCESSOR ARBITRATION MECHANISM

The basic set of arbitration premises requires that the Pentium processor check the second socket (Socket 7) for a processor every time the processor enters reset. To perform the checking of the Socket 7 and to perform the actual boot sequence, the Pentium processor in the 296-pin socket will always come out of reset as the MRM. This will require the part in the Socket 7 to always come out of reset as the LRM.

The LRM processor will request ownership of the processor bus by asserting the private arbitration request pin, PBREQ#. The processor that is currently the MRM and owns the bus, will grant the bus to the LRM as soon as any pending bus transactions have completed. The MRM will grant the bus to the LRM immediately if that CPU has a pipelined cycle to issue. The MRM will notify that the LRM can assume ownership by asserting the private arbitration grant pin, PBGNT#. The PBREQ# pin is always the output of the LRM and the PBGNT# is always an input to the LRM.

A processor can park on the processor bus if there are no requests from the LRM. A parked processor can be running cycles or just sitting idle on the bus. If a processor just ran a cycle on the bus and has another cycle pending without an LRM request, the processor will run the second cycle on the bus.

Locked cycles present an exception to the simple arbitration rules. All locked cycles will be performed as atomic operations without interrupt from the LRM. The case where a locked access causes an assertion of PHITM# by the LRM provides an exception to this rule. In this case, the MRM will grant the bus to the LRM and allow the writeback to complete.

The normal system arbitration pins (HOLD, HLDA, BOFF#) will function the same as in uni-processor mode. Thus, the dual-processor pair will always factor the state of the processor bus as well as the state of the local arbitration before actually running a cycle on the processor bus.

3.4.2.2. DUAL-PROCESSOR ARBITRATION INTERFACE

Figure 3-3 details the hardware arbitration interface**.**

NOTE

For proper operation, PBREQ# and PBGNT# must not be loaded by the system.

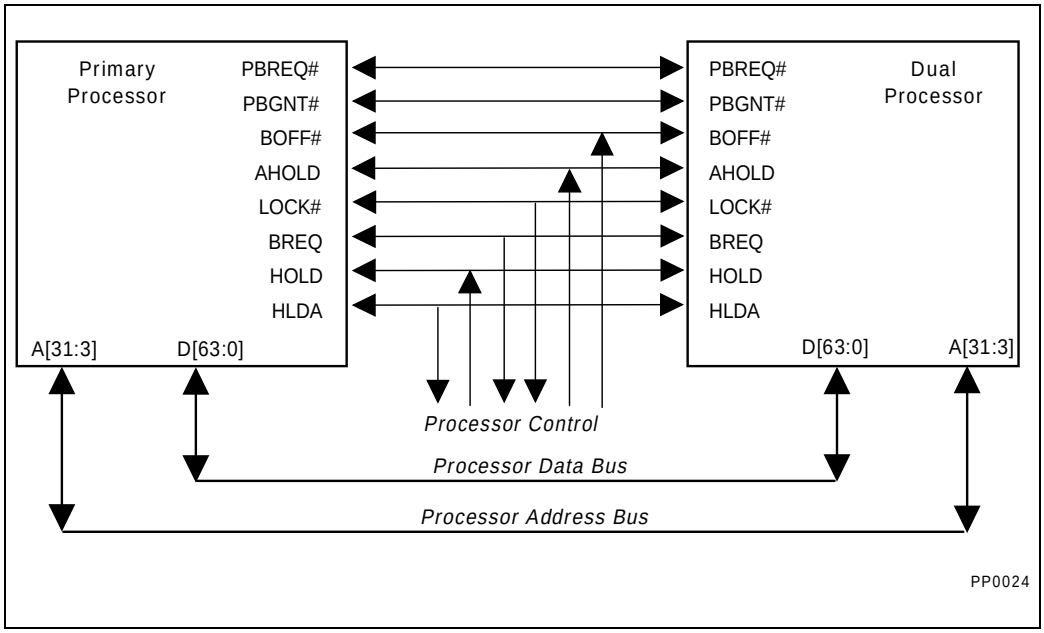


Figure 3-3. Dual-Processor Arbitration Interface

Figure 3-4 shows a typical arbitration exchange.

Diagram (a) of Figure 3-4 shows *PA* running a cycle on the processor bus with a transaction pending. At the same time, *PB* has a cycle pending and has asserted the PBREQ# pin to notify *PA* that *PB* needs the bus.

Diagram (b) of Figure 3-4 shows *PA's* cycle completing with an **NA#** or the last **BRDY#**. Note here that *PA* does not run the pending cycle, instead, *PA* grants the bus to *PB* to allow *PB* to run its pending cycle.

In Diagram (c) of Figure 3-4, *PB* is running the pending transaction on the processor bus, and *PA* asserts a request for the bus to *PB*. The bus is granted to *PA,* and Diagram (d) of Figure 3-4 shows *PA* running the last pending cycle on the bus.

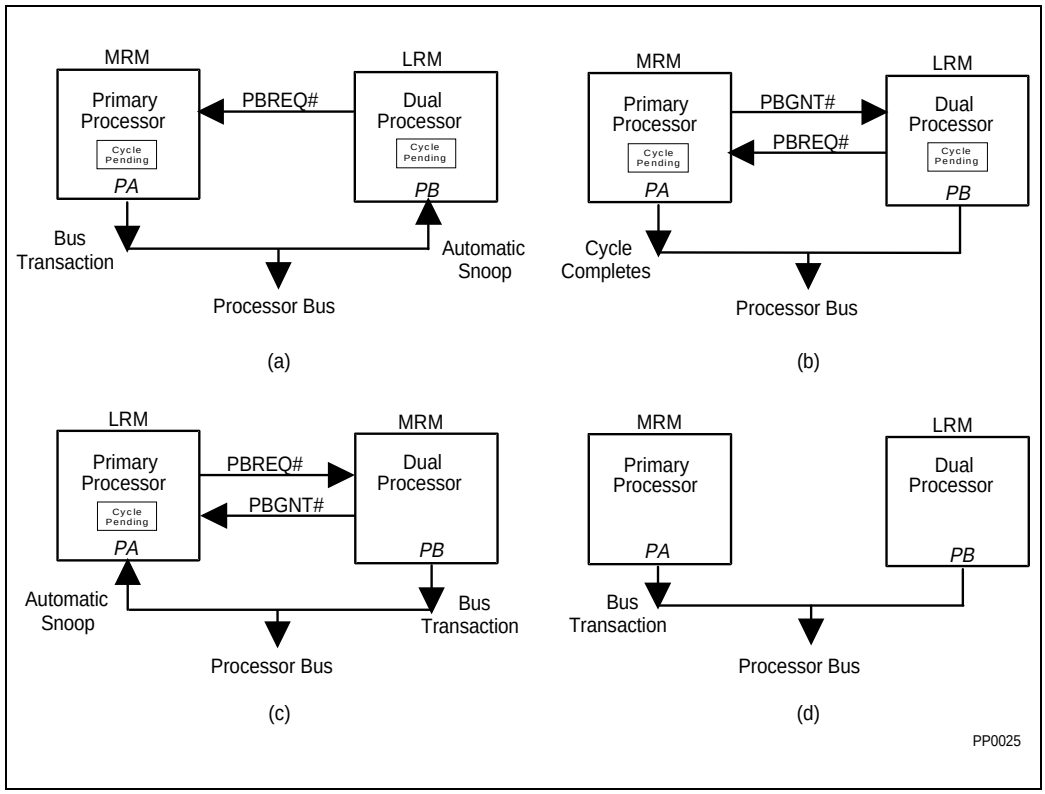


Figure 3-4. Typical Dual-Processor Arbitration Example

3.4.2.3. DUAL-PROCESSOR ARBITRATION FROM A PARKED BUS

When both processors are idle on the CPU bus, and the LRM wants to issue an ADS#, there is an arbitration delay in order that it may become the MRM. Figure 3-5 shows how the Pentium processor dual-processor arbitration mechanism handles this case.

This example shows the arbitration necessary for the LRM to gain control of the idle CPU bus in order to drive a cycle. In this example, *PA* is the Primary processor, and *PB* is the Dual processor.

Diagram (a) of Figure 3-5 shows *PB* requesting the bus from the MRM (*PA*). Diagram (b) of Figure 3-5 shows *PA* granting control of the bus to *PB*. Diagram (c) of Figure 3-5 shows *PB*, now the MRM, issuing a cycle.

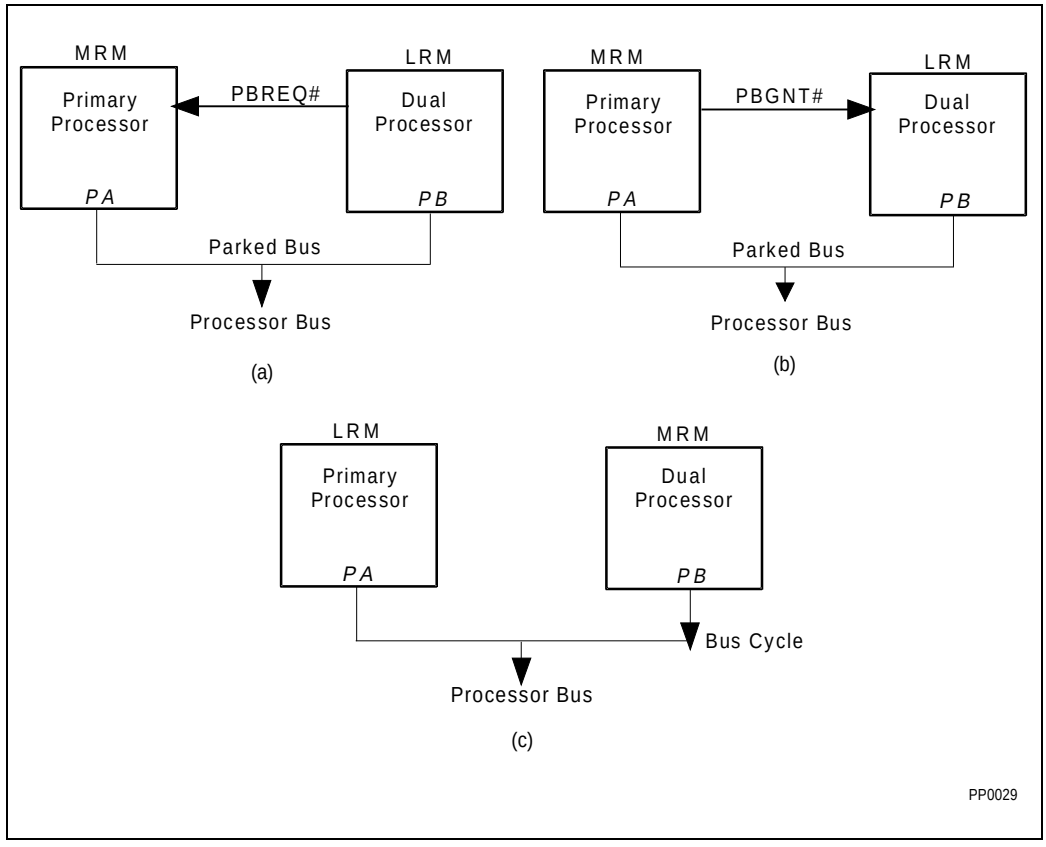


Figure 3-5. Arbitration from LRM to MRM When Bus is Parked

3.4.3. Dual-Processor Cache Consistency

The Pentium processor incorporates a mechanism to maintain cache coherency with the Dual processor. The mechanism allows a dual processor to be inserted into the upgrade socket without special consideration to the system hardware or software. The presence or absence of the dual processor is totally transparent to the system.

3.4.3.1. BASIC CACHE CONSISTENCY MECHANISM

A private snoop interface has been added to the Pentium processor. The interface consists of two pins (PHIT#, PHITM#) that only connect between the two sockets. The dual processors will arbitrate for the system bus via two private arbitration pins (PBREQ#, PBGNT#).

The LRM processor will initiate a snoop sequence for all ADS# cycles to memory that are initiated by the MRM. The LRM processor will assert the private hit indication (PHIT#) if the data accessed (read or written) by the MRM matches a valid cache line in the LRM. In addition, if the data requested by the MRM matches a valid cache line in the LRM that is in the modified state, the LRM will assert the PHITM# signal. The system snooping indication signals (HIT#, HITM#) will not change state as a result of a private snoop.

The Pentium processor will support system snooping via the EADS# pin in the same manner that the Pentium processor supports system snooping.

The private snoop interface is bi-directional. The processor that is currently the MRM will sample the private snoop interface, while the processor that is the LRM will drive the private snoop signals.

The MRM will initiate a self backoff sequence if the MRM detects an assertion of the **PHITM#** signal while running a bus cycle. The self backoff sequence will involve the following steps:

1. The MRM will allow the cycle that was requested on the bus to finish. However, the MRM will ignore the data returned by the system.
2. The MRM-LRM will exchange ownership of the bus (as well as MRM-LRM state) to allow the LRM to write the modified data back to the system.
3. The bus ownership will exchange one more time to allow the original bus master ownership of the bus. At this point the MRM will retry the cycle, receiving the fresh data from the system or writing the data once again.

The MRM will use an assertion of the PHIT# signal as an indication that the requested data is being shared with the LRM. Independent of the WB/WT# pin, a cache line will be placed in the cache in the shared state if PHIT# is asserted. This will make all subsequent writes to that line externally visible until the state of the line becomes exclusive (E or M states). In a uniprocessor system, the line may have been placed in the cache in the E state. In this situation, all subsequent writes to that line will not be visible on the bus until the state is changed to I.

3.4.3.2. CACHE CONSISTENCY INTERFACE

Figure 3-6 details the hardware cache consistency interface**.**

NOTE

For proper operation, PHIT# and PHITM# must not be loaded by the system.

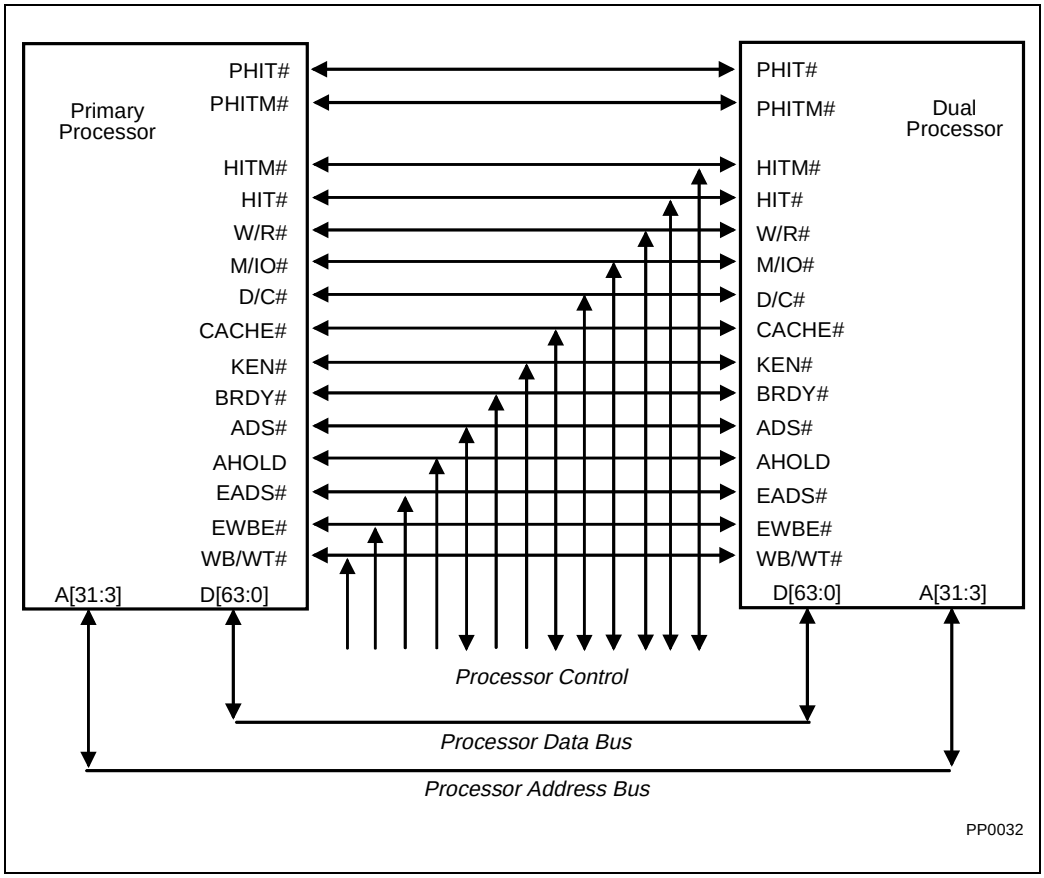


Figure 3-6. Cache Consistency Interface

3.4.3.3. PIN MODIFICATIONS DUE TO THE DUAL-PROCESSOR

The Pentium processor, when operating in dual processing mode, modifies the functionality of the following signals:

- A20M#, ADS#, BE4#-BE0#, CACHE#, D/C#, FERR#, FLUSH#, HIT#, HITM#, HLDA, IGNNE#, LOCK#, M/IO#, PCHK#, RESET, SCYC, SMIACT#, W/R#

Table 3-10 summarizes the functional changes of all the pins in dual processor mode.

3.4.3.4. LOCKED CYCLES

The Pentium processor implements atomic bus transactions by asserting the LOCK# pin. Atomic transactions can be initiated explicitly in software by using a LOCK prefix on specific instructions. In addition, atomic cycles may be initiated implicitly for instructions or transactions that perform locked read-modify-write cycles. By asserting the LOCK# pin, the Pentium processor indicates to the system that the bus transaction in progress can not be interrupted.

3.4.3.4.1. Locked Cycle Cache Consistency

Lock cycles adhere to the following sequence:

1. An unlocked writeback will occur if a cache line is in the modified state in the MRM processor. Two unlocked write back cycles may be required if the locked item spans two cache lines that are both in the modified state.
2. A locked read to a cache line that is in the shared, exclusive or invalid state is always run on the system bus. The cache line will always be moved to the invalid state at the completion of the cycle. A locked read cycle that is run by the MRM could hit a line that is in the modified state in the LRM. In this case, the LRM will assert the PHITM# signal indicating that the requested data is modified in the LRM data cache. The MRM will complete the locked read, but will ignore the data returned by the system. The components will exchange ownership of the bus, allowing the Modified cache line to be written back with LOCK# still active. The sequence will complete with the original bus owner re-running the locked read followed by a locked write. The sequence would be as shown in Figure 3-7.

In Figure 3-7, the small box inside each CPU indicates the state of an individual cache line in the sequence shown above. Diagram (c) of Figure 3-7 shows the locked writeback occurring as a result of the inter-processor snoop hit to the M-state line.

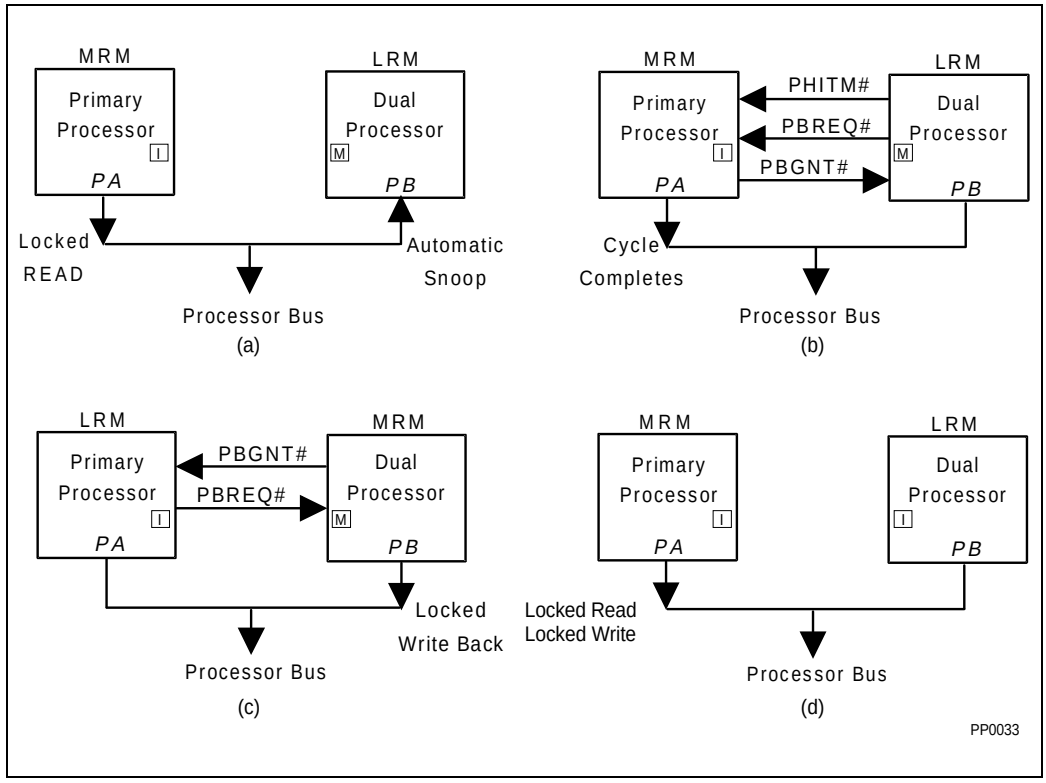


Figure 3-7. Dual-Processor Cache Consistency for Locked Accesses

3.4.3.5. EXTERNAL SNOOP EXAMPLES

3.4.3.5.1. Example 1: During a Write to an M-State Line

The following set of diagrams illustrates the actions performed when one processor attempts a write to a line that is contained in the cache of the other processor. In this situation, the cached line is in the M state in the LRM processor. The external snoop and the write are to the same address in this example.

In this example, *PA* is the Primary processor, and *PB* is the Dual processor.

In diagram (a) of Figure 3-8, processor *PA* starts a write cycle on the bus to a line that is in the M state in processor *PB*. Processor *PB* notifies *PA* that the write transaction has hit an M-state line in diagram (b) of Figure 3-8 by asserting the PHITM# signal. The MRM (*PA*) completes the write cycle on the bus as if the LRM processor did not exist.

In this example, an external snoop happens just as the write cycle completes on the bus, but before *PB* has a chance to write the modified data back to the system memory. Diagram (b) of

Figure 3-8 shows *PB* asserting the HITM# signal, informing the system that the snoop address is cached in the dual processing pair and is in the modified state. The external snoop in this example is hitting the same line that caused the PHITM# signal to be asserted.

Diagram (c) of Figure 3-8 shows that an arbitration exchange has occurred on the bus, and *PB* is now the MRM. Processor *PB* writes back the M state line, and it will appear to the system as if a single processor was completing a snoop transaction.

Finally, diagram (d) of Figure 3-8 shows processor *PA* re-running the original write cycle after *PB* has granted the bus back to *PA*.

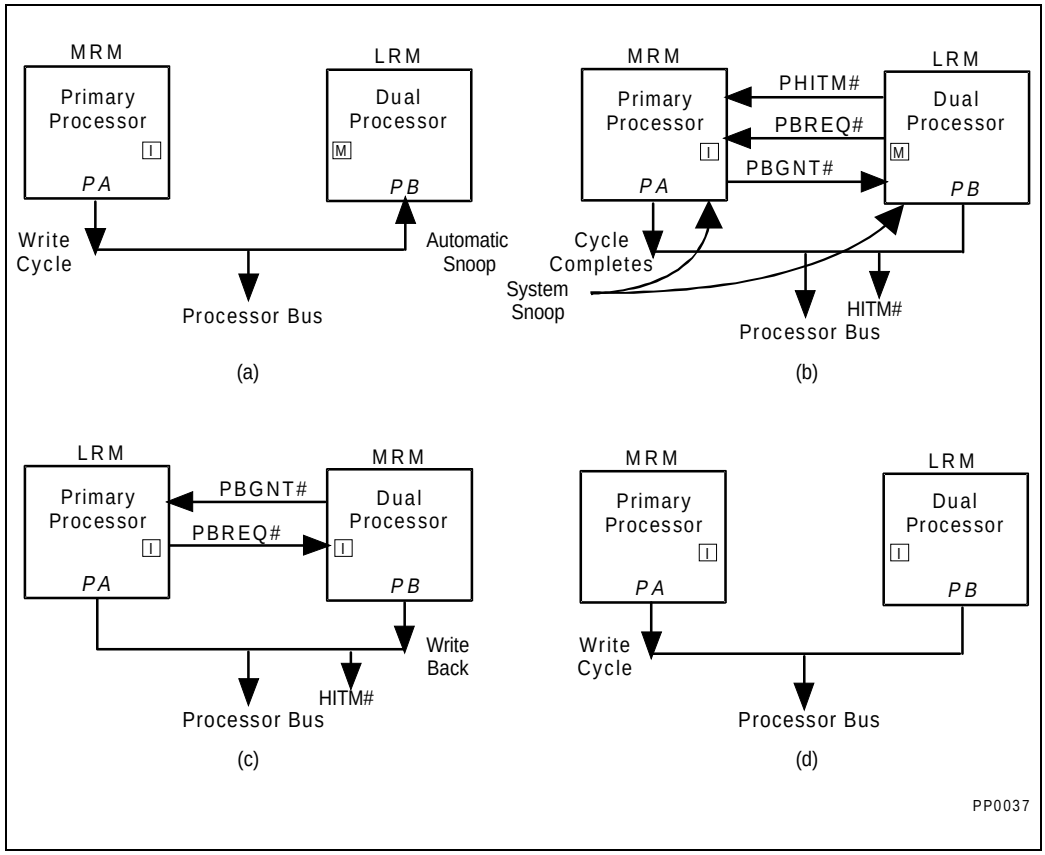


Figure 3-8. Dual-Processor Cache Consistency for External Snoops

3.4.3.5.2. Example 2: During an MRM Self-Backoff

The following diagrams show an example where an external snoop hits an M-state line during a self backoff sequence.

In this example, *PA* is the Primary processor, and *PB* is the Dual processor.

In diagram (a) of Figure 3-9 processor *PA* initiates a write cycle that hits a line that is modified in processor *PB*. In diagram of (b) of Figure 3-9, processor *PB* notifies *PA* that the line is modified in its cache by asserting the PHITM# signal.

Diagram (c) of Figure 3-9 shows an external snoop occurring just as the bus arbitration has exchanged ownership of the bus. Processor *PB* asserts the HITM# signal to notify the system that the external snoop has hit a line in the cache. In this example, the external snoop hits a different line that was just hit on the private snoop.

In diagram (d) of Figure 3-9, processor *PB* takes ownership of the processor bus from *PA*. Processor *PB* initiates a writeback of the data just hit on the external snoop even though a writeback due to the private snoop is pending. The external snoop causes processor *PB* to delay the writeback that was initiated by the private snoop (to line 1).

Diagram (f) of Figure 3-9 shows the writeback of the modified data hit during the initial private snoop. Processor *PA* then restarts the write cycle for the second time, and completes the write cycle in Diagram (h) of Figure 3-9.

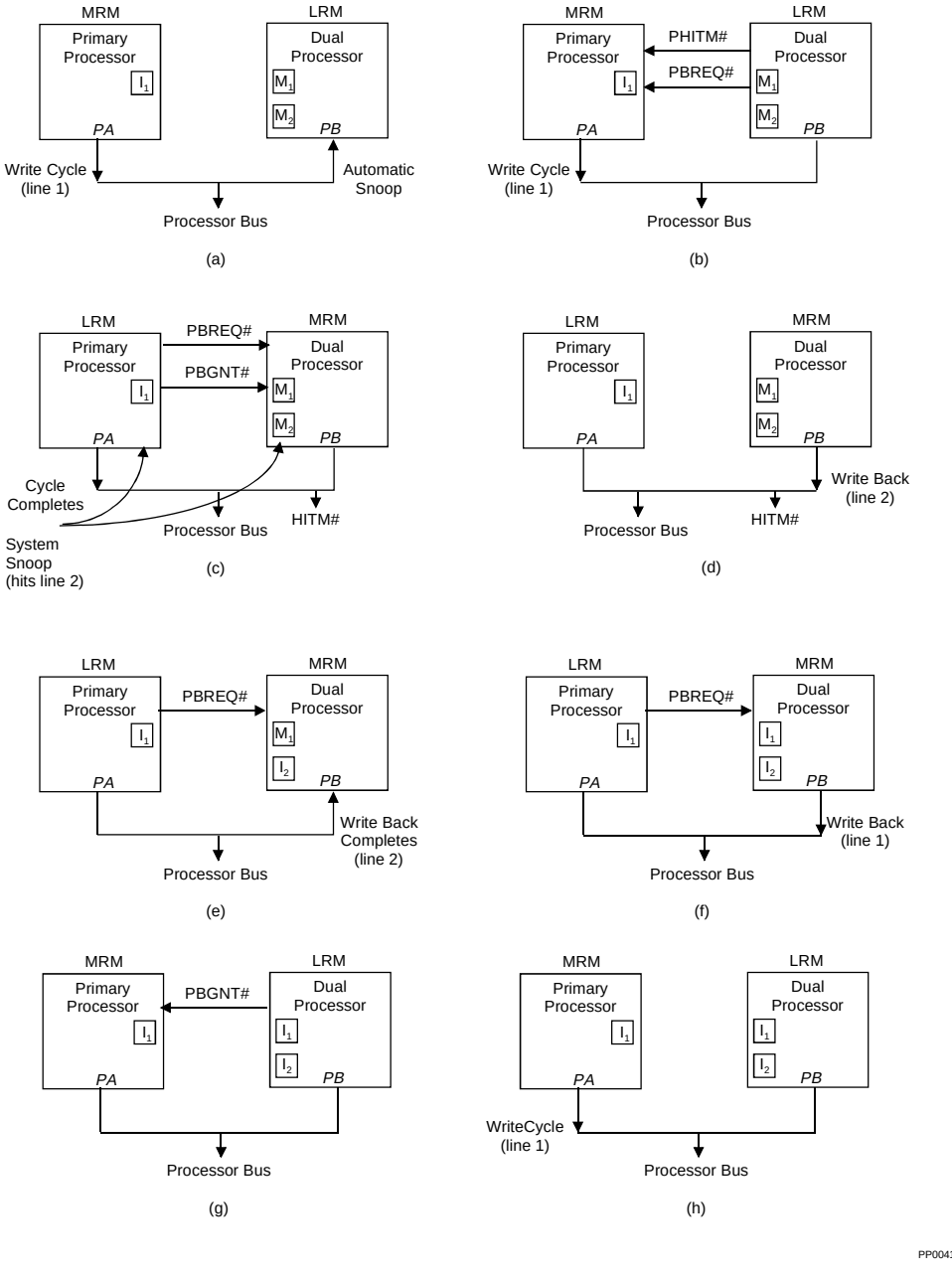


Figure 3-9. Dual-Processor Cache Consistency for External Snoops

3.4.3.6. STATE TRANSITIONS DUE TO DUAL-PROCESSOR CACHE CONSISTENCY

The following tables outline the state transitions that a cache line can encounter during various conditions.

Table 3-3. Read Cycle State Transitions Due to Dual-Processor

Present State	Pin Activity	Next State	Description
M	n/a	M	Read hit. Data is provided to the processor core by the cache. No bus activity.
E	n/a	E	Read hit. Data is provided to the processor core by the cache. No bus activity.
S	n/a	S	Read hit. Data is provided to the processor core by the cache. No bus activity.
I	CACHE#(L) & KEN#(L) & WB/WT#(H) & PHIT#(H) & PWT(L)	E	Cache miss. The cacheability information indicates that the data is cacheable. A bus cycle is requested to fill the cache line. PHIT#(H) indicates that the data is not shared by the LRM processor.
I	CACHE#(L) & KEN#(L) & [WB/WT#(L) + PHIT#(L) + PWT(H)]	S	Cache miss. The line is cacheable and a bus cycle is requested to fill the cache line. In this case, either the system or the LRM is sharing the requested data.
I	CACHE#(H) + KEN#(h)	I	Cache miss. The system or the processor indicates that the line is not cacheable.

NOTE:

The assertion of PHITM# would cause the requested cycle to complete as normal, with the requesting processor ignoring the data returned by the system. The LRM processor would write the data back and the MRM would retry the cycle. This is called a self backoff cycle.

Table 3-4. Write Cycle State Transitions Due to Dual-Processor

Present State	Pin Activity	Next State	Description
M	n/a	M	Write hit. Data is written directly to the cache. No bus activity.
E	n/a	M	Write hit. Data is written directly to the cache. No bus activity.
S	PWT(L) & WB/WT#(H)	E	Write hit. Data is written directly to the cache. A write-through cycle will be generated on the bus to update memory and invalidate the contents of other caches. The LRM will invalidate the line if it is sharing the data. The state transition from S to E occurs AFTER the write completes on the processor bus.
S	PWT(H) + WB/WT#(L)	S	Write hit. Data is written directly to the cache. A write-through cycle will be generated on the bus to update memory and invalidate the contents of other caches. The LRM will invalidate the line if it is sharing the data.
I	n/a	I	Write miss (the Pentium® processor does not support write allocate). The LRM will invalidate the line if it is sharing the data.

Table 3-5. Inquire Cycle State Transitions Due to External Snoop

Present State	Next State (INV=1)	Next State (INV=0)	Description
M	I	S	Snoop hit to an M-state line. HIT# and HITM# will be asserted, followed by a writeback of the line.
E	I	S	Snoop hit. HIT# will be asserted.
S	I	S	Snoop hit. HIT# will be asserted.
I	I	I	Snoop miss.

Table 3-6. State Transitions in the LRM Due to Dual-Processor "Private" Snooping

Present State	Next State (MRM Write)	Next State (MRM Read)	Description
M	I	S	Snoop hit to an M state line. PHIT# and PHITM# will be asserted, followed by a write-back of the line. Note that HIT# and HITM# will NOT be asserted.
E	I	S	Snoop hit. PHIT# will be asserted.
S	I	S	Snoop hit. PHIT# will be asserted.
I	I	I	Snoop miss.

3.5. DESIGNING WITH SYMMETRICAL DUAL PROCESSORS

Figure 3-10 shows how a typical system might be configured to support the Dual processor.

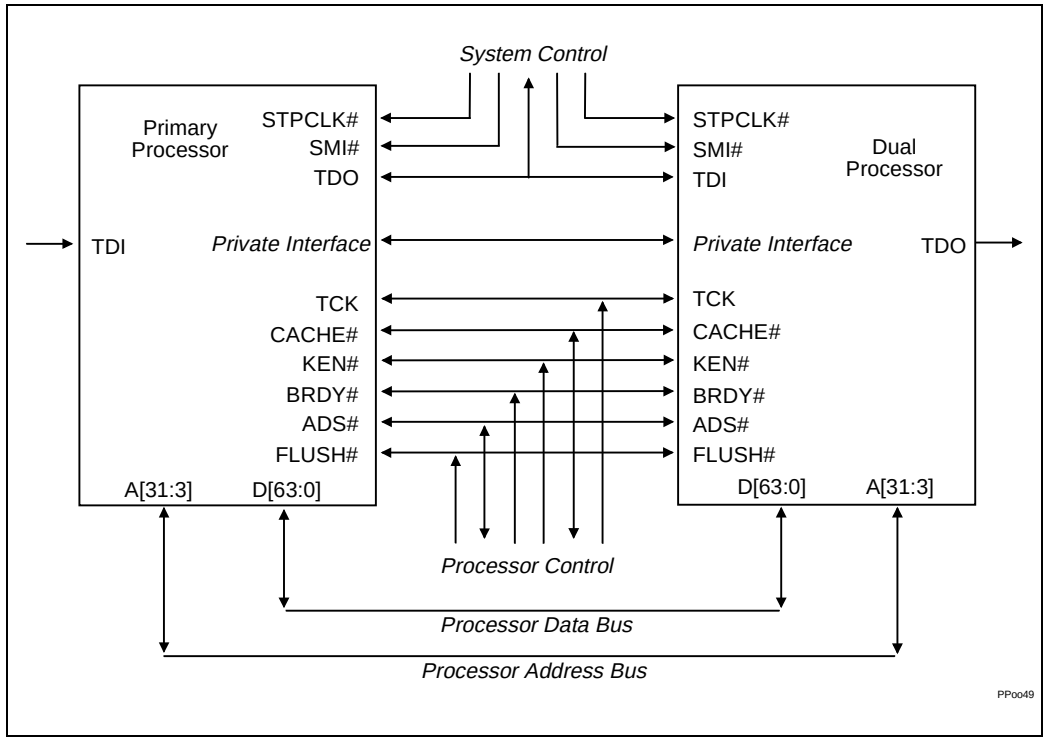


Figure 3-10. Dual-Processor Configuration

Refer toTable 3-**Error! Bookmark not defined.** for a complete list of dual processor signal connection requirements.

3.5.1. Dual Processor Bus Interface

The Pentium processor in the dual-processor configuration is designed to have an identical bus interface to a standard Pentium processor system. The Pentium processor in dual processor mode has the capability to run the following types bus of cycles:

- Single reads and writes from one processor.
- Burst reads and writes from one processor.
- Address pipelining with up to two outstanding bus cycles from one processor.
- Inter-processor address pipelining with up to two outstanding bus cycles, one from each processor.

All cycles run by the two processors are clock accurate to corresponding Pentium processor bus cycles.

3.5.1.1. INTRA- AND INTER-PROCESSOR PIPELINING

In uni-processor mode, the Pentium processor supports bus pipelining with the use of the NA# pin. The bus pipelining concept has been extended to the dual processor pair by allowing inter-CPU pipelining. This mechanism allows an exchange between LRM and MRM on assertions of NA#.

When NA# is sampled low, the current MRM processor may drive one more cycle onto the bus or it may grant the address bus and the control bus to the LRM. The MRM will give the bus to the LRM only if its current cycle can have another cycle pipelined into it.

The cacheability (KEN#) and cache policy (WB/WT#) indicators for the current cycle are sampled either in the same clock that NA# is sampled or with the first BRDY# of the current cycle, whichever comes first.

There are no restrictions on NA# due to dual processing mode.

Inter-CPU pipelining will not be supported in some situations as shown in Table 3-7.

Table 3-7. Primary and Dual Processor Pipelining

		Primary and Dual Processor Pipelining		
Cycle Types		Inter-CPU	Intra-CPU	
First Cycle	Pipelined Cycle	Primary<>Dual	Primary<>Primary	Dual<>Dual
Write Back	X	No	No	No
LOCK#	X	No	No	No
X	Write Back	No	No	No
X	LOCK#	No	No	No
Write	***Write***	***No***	***Yes***	***Yes***
Write	Read	Yes	Yes	Yes
Read	Write	Yes	Yes	Yes
Read	Read	Yes	Yes	Yes
I/O	***I/O****	***Yes***	***No***	***No***

NOTE:

*I/O write cycles may not be inter-processor pipelined into I/O write cycles.

The table indicates that, unlike the uni-processor Pentium processor system, back-to-back write cycles will never be pipelined between the two processors.

The Pentium processor alone may pipeline I/O cycles into non-I/O cycles, non-I/O cycles into I/O cycles, and I/O cycles into I/O cycles only for OUTS or INS (e.g. string instructions). I/O cycles may be pipelined in any combination (barring writes into writes) between the Primary and Dual processors.

3.5.1.2. FLUSH# CYCLES

The on-chip caches can be flushed by asserting the FLUSH# pin. The FLUSH# pin must be connected together to both the Primary and Dual processor parts. All cache lines in the instruction cache as well as all lines in the data cache that are not in the modified state will be invalidated when the FLUSH# pin is asserted. All modified lines in the data cache will be written back to system memory and then marked as invalid in the data cache. The Pentium processor will run a special bus cycle indicating that the flush process has completed.

The Pentium processor incorporates the following mechanism to present a unified view of the cache flush operation to the system when used with a Dual processor part:

1. FLUSH# is asserted by the system.
2. The Dual processor requests the bus (if it is not already MRM when FLUSH# is recognized). The Dual processor will always perform the cache flush operation first, but will not run a flush special cycle on the system bus.
3. The Dual processor completes writebacks of modified cache lines, and invalidates all others.

4. Once the Dual processor caches are completely invalid, the processor grants the bus to the Primary processor.
5. The Primary processor completes any pending cycles. The Primary processor may have outstanding cycles if the Dual processor initiated its flush operation prior to the Primary processor completing pending operations.
6. Primary processor flushes both of its internal caches and runs the cache flush special cycle. The Primary processor maintains its status of MRM. The Dual processor halts all code execution while the Primary processor is flushing its caches, and does not begin executing code until it recognizes the flush acknowledge special cycle.

The atomic flush operation assumes that the system can tolerate potentially longer interrupt latency during flush operations. The interrupt latency in a dual processor system can be double the interrupt latency in a single processor system during flush operations.

The Pentium processor primary cache can be flushed using the WBINVD instruction. In a dual processor system, the WBINVD instruction only flushes the cache in the processor that executed the instruction. The other processor's cache will be intact.

If the FLUSH# signal is de-asserted before the corresponding Flush Acknowledge cycle, the FLUSH# signal **must** not be asserted again until the Flush Acknowledge cycle is completed. Similarly, if the FLUSH# signal is asserted in dual processing mode, it must be deasserted at least one clock prior to BRDY# of the Flush Acknowledge cycle to avoid dual-processor arbitration problems. This requirement does not apply to a uni-processor system. In a dual processor system, a single Flush Acknowledge cycle is generated after the caches in both processors have been flushed.

WARNING

If FLUSH# is recognized active a second time by the Primary and Dual processors prior to the completion of the Flush Acknowledge special cycle, the private bus arbitration state machines will be corrupted.

3.5.1.3. ARBITRATION EXCHANGE — WITH BUS PARKING

The dual processor pair supports a number of different types of bus cycles. Each processor can run single-transfer cycles or burst-transfer cycles. A processor can only initiate bus cycles if it is the MRM. To gain ownership of the bus, the LRM processor will request the bus from the MRM processor by asserting PBREQ#.

In response to PBREQ# the MRM will grant the address and the control buses to the LRM by asserting PBGNT#. If NA# is not asserted or if the current cycle on the bus is not capable of being pipelined, the MRM will wait until the end of the active cycle before granting the bus to the LRM. Once PBGNT# is asserted, since the bus is idling, the LRM will immediately become the MRM. While the MRM, the processor owns the address and the control buses and can therefore start a new cycle.

3.5.1.4. BOFF#

If BOFF# is asserted, the dual-processor pair will immediately (in the next clock) float the address, control, and data buses. Any bus cycles in progress are aborted and any data returned to the processor in the clock BOFF# is asserted is ignored. In response to BOFF#, Primary and Dual processors will float the same pins as it does when HOLD is active.

The Primary and Dual processors may reorder cycles after a BOFF#. The reordering will occur if there is inter-CPU pipelining at the time of the BOFF#, but the system cannot change the cacheability of the cycles after the BOFF#. Note that there could be a change of bus ownership transparent to the system while the processors are in the backed-off state. Table 3-8 illustrates the flow of events which would result in cycle reordering due to BOFF#:

Table 3-8. Cycle Reordering Due to BOFF#

Time*	Processor A	System	Processor B
0	ADS# driven	--	--
1	--	NA# active	--
2	--	--	ADS# driven
3	Bus float	BOFF# active	Bus float
4	--	EADS# active	--
5	--	--	HITM# driven
6	--	BOFF# inactive	--
7	--	--	Write back ‘M’ data
8	--	BRDY#s	--
9	--	--	Restart ADS#
10	Restart ADS#	--	--

NOTE:

*Time is merely sequential, NOT measured in CLKs.

3.5.1.5. BUS HOLD

The Pentium processor supports a bus hold/hold acknowledge protocol using the HOLD and HLDA signals. When the Pentium processor completes all outstanding bus cycles, it will release the bus by floating the external bus, and driving HLDA active. HLDA will normally be driven two clocks after the later of the last BRDY# or HOLD being asserted, but may be up to six clocks due to active internal APIC cycles. Because of this, it is possible that an additional cycle may begin after HOLD is asserted but before HLDA is driven. Therefore, asserting HOLD does not prevent a dual-processor arbitration from occurring before HLDA is driven out. Even if an arbitration switch occurs, no new cycles will be started after HOLD has been active for two clocks.

3.5.2. Dual Processing Power Management

3.5.2.1. STPCLK#

The Primary and Dual processor STPCLK# signals may be tied together or left separate. Refer to Chapter 14 for more information on stop clock and Autohalt.

3.5.2.2. SYSTEM MANAGEMENT MODE

The Pentium processor supports system management mode (SMM) with a processor inserted in the upgrade socket. SMM provides a means to implement power management functions as well as operating system independent functions. SMM in the Pentium processor consists of an interrupt (SMI), an alternate address space and an instruction (RSM). SMM is entered by asserting the SMI# pin or delivering the SMI interrupt via the local APIC.

Although SMM functions the same when a Dual processor is inserted in Socket 5/Socket 7, the dual processor operation of the system must be carefully considered. The SMI# pins may be tied together or not, depending upon the power management features supported.

In order to ensure proper SMM operation when a future Pentium OverDrive processor upgrade is installed in the system, it is recommended that the SMI# and SMIACT# signals be connected together. Refer to Chapter 14 for more details.

3.5.3. Other Dual-Processor Considerations

3.5.3.1. STRONG WRITE ORDERING

The ordering of write cycles in the processor can be controlled with the EWBE# pin. During uniprocessor operation, the EWBE# pin is sampled by the Pentium processor with each BRDY# assertion during a write cycle. The processor will stall all subsequent write operations to E or M state lines if EWBE# is sampled inactive. If the EWBE# pin is sampled inactive, it will continue to be sampled on every clock until it is found to be active.

In dual processing mode, each processor will track EWBE# independently of bus ownership. EWBE# is sampled and handled independently between the two processors. Only the processor which owns the bus (MRM) samples EWBE#. Once sampled inactive, the CPU will stall subsequent write operations.

3.5.3.2. BUS SNARFING

The dual processor pair does not support cache-to-cache transfers (bus snarfing). If a processor *PB* requires data that is modified in processor *PA*, processor *PA* will write the data back to memory. After *PA* has completed the data transfer, *PB* will run a read cycle to memory. Where *PA* is either the Primary or the Dual processor, and *PB* is the other processor.

3.5.3.3. INTERRUPTS

A processor may need to arbitrate for the use of the bus as a result of an interrupt. However, from the simple arbitration model used by the Pentium processor, an interrupt is not a special case. There is no interaction between dual-processor support and the interrupt model in the Pentium processor.

3.5.3.4. INIT SEQUENCES

The INIT operation in dual-processor mode is exactly the same as in uni-processor mode. The two INIT pins must be tied together. However, in dual processor mode, the Primary processor must send an IPI and a starting vector to the Dual processor via the local APIC modules.

3.5.3.5. BOUNDARY SCAN

The Pentium processor supports the full IEEE JTAG specification. The system designer is responsible to configure an upgrade ready system in such a way that the addition of a Dual processor in Socket 7 allows the boundary scan chain to functional as normal. This could be implemented with a jumper in Socket 7 that connects the TDI and TDO pins. The jumper would then be removed when the dual processor is inserted.

Alternatively, Socket 7 could be placed near the end of the boundary scan chain in the system. A multiplexer in the system boundary scan logic could switch between the TDO of the Primary and the dual processors as a Dual processor part is inserted. An illustration of this approach is shown in Figure 3-11.

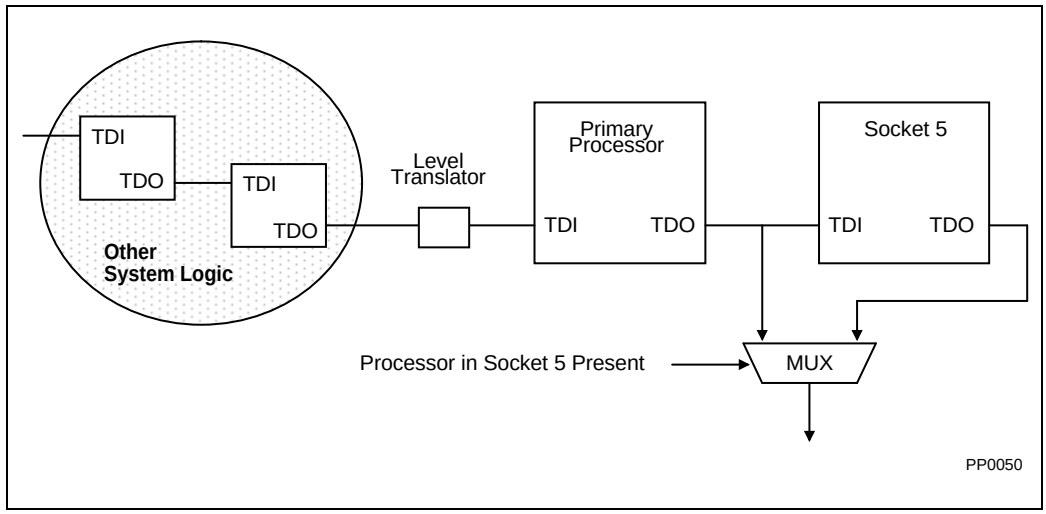


Figure 3-11. Dual-Processor Boundary Scan Connections

3.5.3.6. PRESENCE OF A PROCESSOR IN SOCKET 7

The Dual processor or future Pentium OverDrive processor drives the DPEN# signal low during RESET to indicate to the Primary processor that a processor is present in Socket 7. The Pentium processor samples this line during RESETs falling edge.

DPEN# shares a pin with the APIC PICD0 signal.

3.5.3.7. MRM PROCESSOR INDICATION

In a dual-processor system, the D/P# (Dual processor/Primary processor Indication) signal indicates which processor is running a cycle on the bus. Table 3-9 shows how the external hardware can determine which CPU is the MRM.

Table 3-9. Using D/P# to Determine MRM

D/P#	Bus Owner
0	Primary processor is MRM
1	Dual processor is MRM

D/P# can be sampled by the system with ADS# to determine which processor is driving the cycle on the bus.

D/P# is driven only by the Pentium processor when operating as the Primary processor. Because of this, this signal is never driven by the Dual processor and does not exist on the future Pentium OverDrive processor. When the future Pentium OverDrive processor is installed, the Pentium processor continues to drive the D/P# signal high despite being "shut down."

3.5.4. Dual-Processor Pin Functions

All the inputs pins described in Chapter 4 are sampled with bus clock or test clock, and therefore, must meet setup and hold times with respect to the rising edge of the appropriate clock. In the dual-processor configuration, the RESET and FLUSH# pins have been changed to be synchronous (i.e. meet setup and hold times). There have been no changes to the other existing input pins.

If the FLUSH# signal is deasserted before the corresponding FLUSH ACK cycle, the FLUSH# signal must not be asserted again until the FLUSH ACK cycle is generated. This requirement does not apply to a uni-processor system. In a dual processor system, a single FLUSH ACK cycle is generated after the caches in both processors have been flushed.

All system output pins will be driven from the rising edge of the bus clock and will meet maximum and minimum valid delays with respect to the bus clock. TDO is driven with respect to the rising edge of TCK and PICD0-1 are driven with respect to the rising edge of PICCLK.

Table 3-10 summarizes the functional changes of all the pins in dual-processor mode.

Table 3-10. Dual-Processor Pin Functions vs. Pentium® Processor

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
A[31:3]	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates these signals for one CLK.
A20M#	I	Y	Y	Yes	Used in virtual mode and possibly in real mode by DOS and DOS extenders. Internally masked by the Dual processor. It is necessary to connect this signal to Socket 7 in order for proper future Pentium® OverDrive® processor operation.
ADS#, ADSC#	I/O O	Y	N	Yes	ADS# and ADSC# are tristated by the LRM processor in order to allow the MRM processor to begin driving them. There are no system implications.
AHOLD	I	Y	Y	Yes	
AP	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
APCHK#	O	N	Y	No	Requires a system OR function.
BE[7:5]# BE[4:0]#	O I/O	Y Y	N N	Yes Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates these signals for one CLK. BE[3:0]# are used by the local APIC modules to load the APIC_ID at RESET. BE[3:0]# will be tristated by the Primary and Dual processors during RESET.
BF	I	Y	n/a	Yes	
BOFF#	I	Y	Y	Yes	
BP[3:0]	O	N	N	No	BP[3:0] will now only indicate breakpoint match in the I/O clock. Each processor must have different breakpoints. Note that BP[1:0] are mux'd with PM[1:0].
BRDY#, BRDYC#	I	Y	Y	Yes	

Table 3-10. Dual-Processor Pin Functions vs. Pentium® Processor (Contd.)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
BREQ	O	Y	N	Yes	The MRM drives this signal as a combined bus cycle request for itself and the LRM.
BUSCHK#	I	Y	Y	Yes	
CACHE#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
CLK	I	Y	Y	Yes	Both processors must use the same system clock.
CPUTYP	I	Y	n/a	No	
D/C#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
D/P#	O	n/a	n/a	No	The Primary processor always drives this signal. This output is not defined on the Dual processor or future Pentium OverDrive processor.
D[63:0]	I/O	Y	Y	Yes	
DP[7:0]	I/O	Y	Y	Yes	
EADS#	I	Y	Y	Yes	
EWBE#	I	Y	Y	Yes	This signal is sampled active with BRDY#, but inactive asynchronously. For optimized performance (minimum number of write E/M stalls) the chip set/platform should allow a dead clock between buffer going empty to buffer going full. This will allow this signal to be completely independent between the two processors and not have one stall internal cache writes due to the other filling the external buffer.
FERR#	O	Y	Y	Yes	Used for DOS floating point compatibility. The Primary processor will drive this signal. The Dual processor will never drive this signal.

Table 3-10. Dual-Processor Pin Functions vs. Pentium® Processor (Contd.)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
FLUSH#	I	Y	Y	Yes	In a dual-processor system, the flush operation will be atomic with a single flush acknowledge bus cycle. Therefore, FLUSH# must not be re-asserted until the corresponding FLUSH ACK cycle is generated.
FRCMC#	I	N	Y	Yes	Both processors must be in Master mode. A processor in the Socket 7 cannot be used as a Checker.
HIT#	I/O	Y	N	Yes	This signal is asserted by the MRM based on the combined outcome of the inquire cycle between the two processors.
HITM#	I/O	Y	N	Yes	See HIT#.
HLDA	I/O	Y	N	Yes	Driven by the MRM.
HOLD	I	Y	Y	Yes	
IERR#	O	N	Y	No	
IGNNE#	I	Y	Y	Yes	The Dual processor will ignore this signal.
INIT	I	N	N	Yes	In dual-processor mode, the Dual processor requires an IPI during initialization.
INTR/LINT0	I	N	N	May Be	If the APIC is enabled, then this pin is a local interrupt. If the APIC is hardware disabled, this pin function is not changed.
INV	I	Y	Y	Yes	
KEN#	I	Y	Y	Yes	
LOCK#	I/O	Y	N	Yes	The LRM samples the value of LOCK#, and drives the sampled value in the clock it gets the ownership of the dual-processor bus. If sampled active, then the LRM will keep driving the LOCK# signal until ownership changes again.
M/IO#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.

Table 3-10. Dual-Processor Pin Functions vs. Pentium® Processor (Contd.)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
NA#	I	Y	Y	Yes	
NC	n/a	N	Y	No	
NMI/LINT1	I	N	Y	May Be	If the APIC is enabled, then this pin is a local interrupt. If the APIC is hardware disabled, this pin function is not changed.
PBGNT#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PBREQ#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PCD	O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
PCHK#	O	N	Y	May Be	May be wire-AND'd together in the system, tied together, or the chip set may have two PCHK# inputs for dual-processor data parity.
PEN#	I	Y	Y	Yes	
PHIT#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PHITM#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PHITM#	I/O	n/a	n/a	Yes	This signal is always driven by one of the processors.
PICCLK	I	Y	n/a	Yes	
PICD[1:0]	I/O	Y	n/a	Yes	
PM[1:0]	O	N	N	No	Each processor may track different performance monitoring events. Note that PM[1:0] are mux'd with BP[1:0].
PRDY	O	N	Y	No	
PWT	O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
R/S#	I	N	Y	No	

Table 3-10. Dual-Processor Pin Functions vs. Pentium® Processor (Contd.)

Pin Name	I/O	Load (Note 1)	Same? (Note 2)	Tied Together? (Note 3)	Comments
RESET	I	Y	Y	Yes	In dual-processor mode, RESET must be synchronous to the CPU CLK which goes to the Primary and Dual processors.
SCYC	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
SMI#	I	N	Y	May Be	Refer to Chapter 14.
SMIACT#	O	N	Y	Yes	Refer to Chapter 14.
STPCLK#	I	n/a	n/a	May Be	Refer to Chapter 14.
TCK	I	n/a	n/a	May Be	System dependent
TDI	I	n/a	n/a	No	System dependent
TDO	O	n/a	n/a	No	System dependent
TMS	I	n/a	n/a	May Be	System dependent
TRST#	I	n/a	n/a	May Be	System dependent
V_{CC}	I	N	N	Yes	V_{CC} on the Pentium processor must be connected to 3.3V.
$V_{CC}5$	I	N	Y	no	Two $V_{CC}5$ pins remain on the future Pentium OverDrive processor in order to support a 5V fan/heatsink in the future.
V_{SS}	I	N	Y	Yes	
W/R#	I/O	Y	N	Yes	When the MRM becomes the LRM (and issues PBGNT#), it tristates this signal for one CLK.
WB/WT#	I	Y	Y	Yes	

NOTES:

1. “Load” indicates whether the pin would introduce a capacitive load to the system board due to the dual processor being present.
2. “N” indicates that there is a minor functional change to the pin(s) either as an enhancement to the Pentium processor or due to dual processor operation.
3. “Yes” means that both processors must see the same value on the pin(s) for proper dual-processor operation. “No” means that the system must provide the signal to each processor independently. “May Be” means that the system designer can choose to provide the signal to both processors or provide independent signals to each processor.

4

Pinout

CHAPTER 4
PINOUT

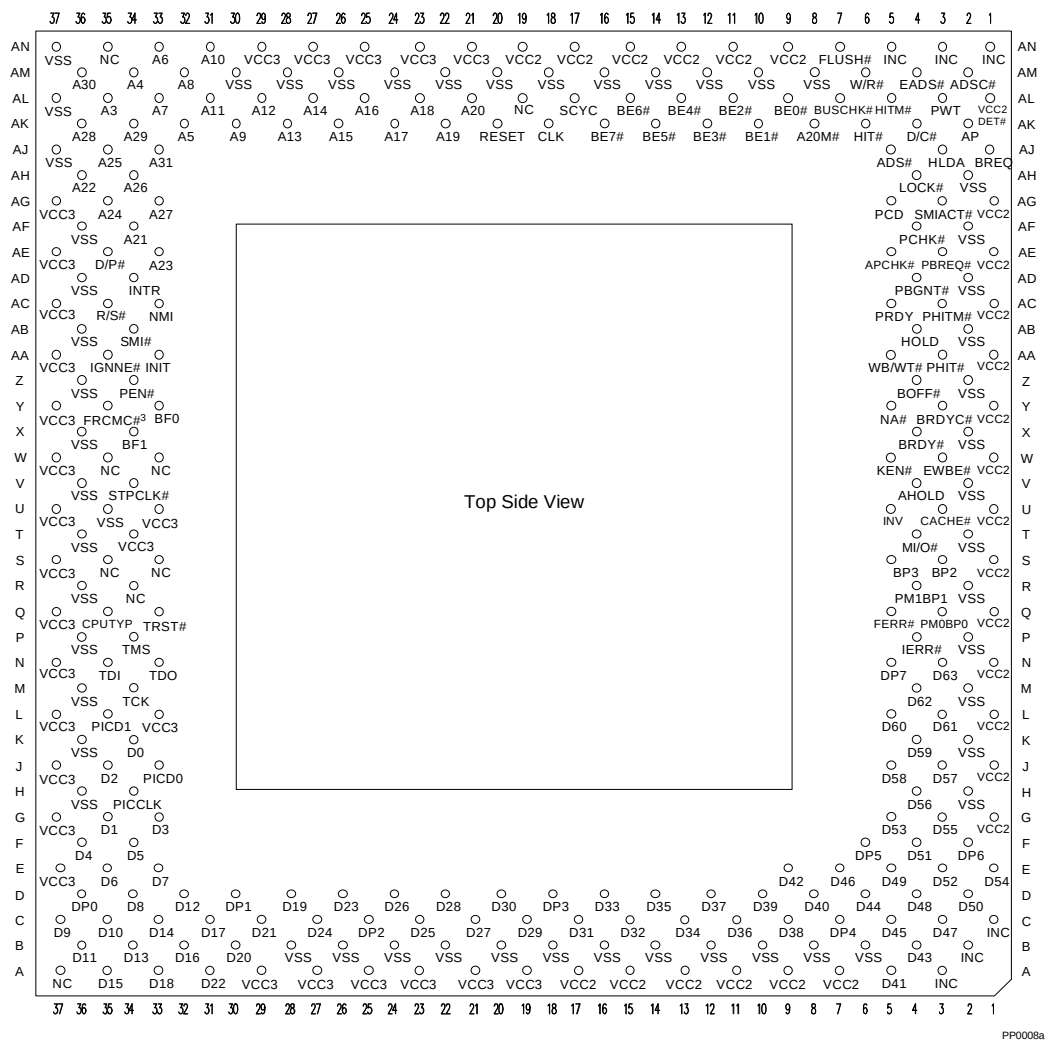
The Pentium® processor with MMX™ technology and the Pentium processor (75/90/100/120/133/150/166/200) are both available in a 296-pin Ceramic Staggered Pin Grid Array (SPGA) package and a Plastic Pin Grid Array (PPGA) package.

4.1. PINOUT AND CROSS REFERENCE TABLES

The text orientation on the top side view drawings in this section represents the orientation of the ink mark on the actual packages. (Note that the text shown in this section is not the actual text which is marked on the packages).

Figure 4-1 and Figure 4-2 illustrate the top side view and the pin side view of the Pentium processor pinout.

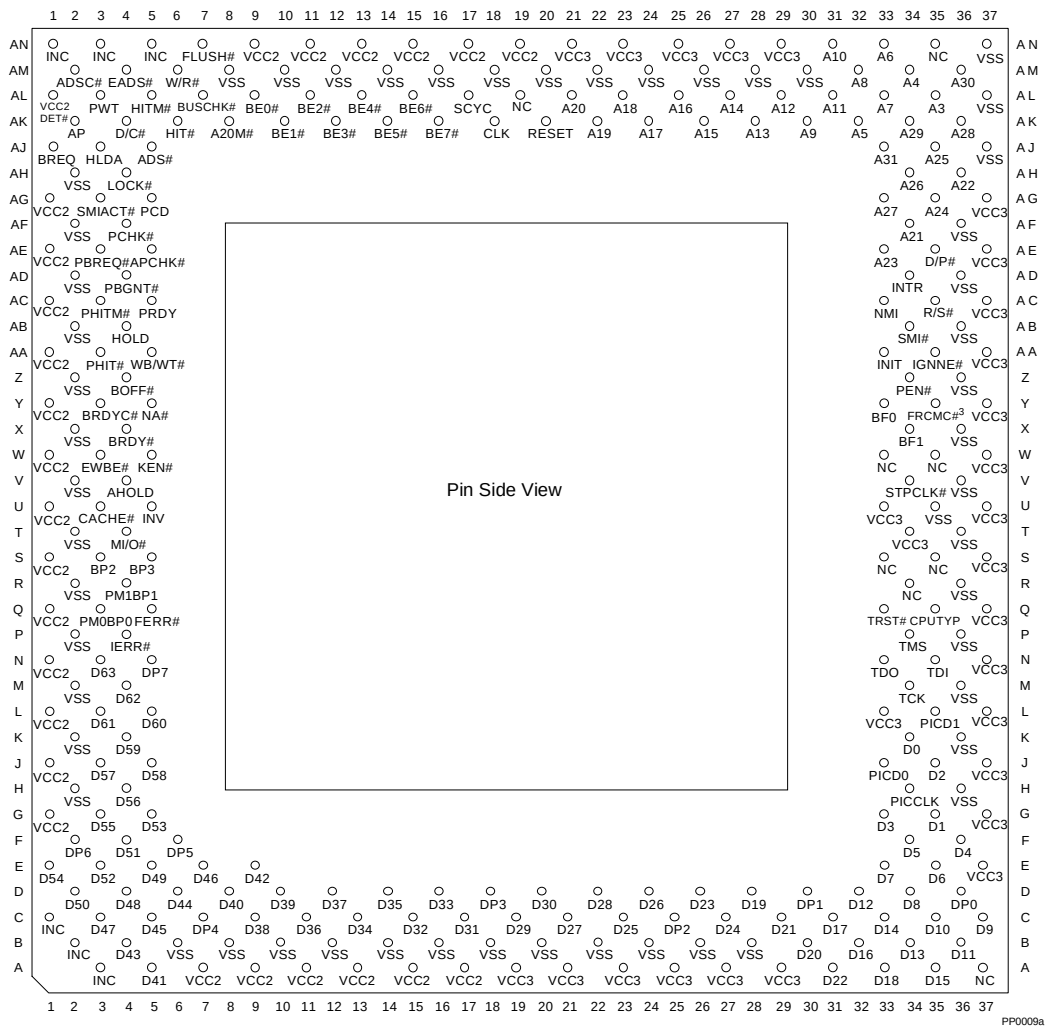
4.1.1. Pinout



NOTES:

1. The $V_{CC}2$ and $V_{CC}3$ pins are collectively defined as V_{CC} pins on the Pentium® processor (75/90/100/120/133/150/166/200).
2. The VCC2DET# pin is defined only for the Pentium processor with MMX™ technology. Pin AL01 is an INC on the Pentium processor (75/90/100/120/133/150/166/200).
3. The FRCMC# pin is defined only for the Pentium processor (75/90/100/120/133/150/166/200). Pin Y35 should be left as a "NC" or tied to $V_{CC}3$ via an external pullup resistor on the Pentium processor with MMX technology.

Figure 4-1. Pentium® Processor Pinout — Top Side View



NOTES:

1. The $V_{CC}2$ and $V_{CC}3$ pins are collectively defined as V_{CC} pins on the Pentium® processor (75/90/100/120/133/150/166/200).
2. The VCC2DET# pin is defined only for the Pentium processor with MMX™ technology. Pin AL01 is an INC on the Pentium processor (75/90/100/120/133/150/166/200).
3. The FRCMC# pin is defined only for the Pentium processor (75/90/100/120/133/150/166/200). Pin Y35 should be left as a "NC" or tied to $V_{CC}3$ via an external pullup resistor on the Pentium processor with MMX technology.

Figure 4-2. Pentium® Processor Pinout — Pin Side View

4.1.2. Pin Cross Reference Table

Table 4-1. Pin Cross Reference by Pin Name

Address									
A3	AL35	A9	AK30	A15	AK26	A21	AF34	A27	AG33
A4	AM34	A10	AN31	A16	AL25	A22	AH36	A28	AK36
A5	AK32	A11	AL31	A17	AK24	A23	AE33	A29	AK34
A6	AN33	A12	AL29	A18	AL23	A24	AG35	A30	AM36
A7	AL33	A13	AK28	A19	AK22	A25	AJ35	A31	AJ33
A8	AM32	A14	AL27	A20	AL21	A26	AH34		
Data									
D0	K34	D13	B34	D26	D24	D39	D10	D52	E03
D1	G35	D14	C33	D27	C21	D40	D08	D53	G05
D2	J35	D15	A35	D28	D22	D41	A05	D54	E01
D3	G33	D16	B32	D29	C19	D42	E09	D55	G03
D4	F36	D17	C31	D30	D20	D43	B04	D56	H04
D5	F34	D18	A33	D31	C17	D44	D06	D57	J03
D6	E35	D19	D28	D32	C15	D45	C05	D58	J05
D7	E33	D20	B30	D33	D16	D46	E07	D59	K04
D8	D34	D21	C29	D34	C13	D47	C03	D60	L05
D9	C37	D22	A31	D35	D14	D48	D04	D61	L03
D10	C35	D23	D26	D36	C11	D49	E05	D62	M04
D11	B36	D24	C27	D37	D12	D50	D02	D63	N03
D12	D32	D25	C23	D38	C09	D51	F04		

Control							
A20M#	AK08	BREQ	AJ01	HIT#	AK06	PRDY	AC05
ADS#	AJ05	BUSCHK#	AL07	HITM#	AL05	PWT	AL03
ADSC#	AM02	CACHE#	U03	HLDA	AJ03	R/S#	AC35
AHOLD	V04	CPUTYP	Q35	HOLD	AB04	RESET	AK20
AP	AK02	D/C#	AK04	IERR#	P04	SCYC	AL17
APCHK#	AE05	D/P#	AE35	IGNNE#	AA35	SMI#	AB34
BE0#	AL09	DP0	D36	INIT	AA33	SMIACT#	AG03
BE1#	AK10	DP1	D30	INTR/LINT0	AD34	TCK	M34
BE2#	AL11	DP2	C25	INV	U05	TDI	N35
BE3#	AK12	DP3	D18	KEN#	W05	TDO	N33

Table 4-1. Pin Cross Reference by Pin Name (Contd.)

Control (Contd.)							
BE4#	AL13	DP4	C07	LOCK#	AH04	TMS	P34
BE5#	AK14	DP5	F06	M/IO#	T04	TRST#	Q33
BE6#	AL15	DP6	F02	NA#	Y05	VCC2DET#	AL01 [1]
BE7#	AK16	DP7	N05	NMI/LINT1	AC33	W/R#	AM06
BOFF#	Z04	EADS#	AM04	PCD	AG05	WB/WT#	AA05
BP2	S03	EWBE#	W03	PCHK#	AF04		
BP3	S05	FERR#	Q05	PEN#	Z34		
BRDY#	X04	FLUSH#	AN07	PM0/BP0	Q03		
BRDYC#	Y03	FRCMC#[2]	Y35	PM1/BP1	R04		
APIC		**Clock Control**		**Dual Processor Private Interface**			
PICCLK	H34 [3]	CLK	AK18 [3]	PBGNT#	AD04		
PICD0	J33	[BF0]	Y33	PBREQ#	AE03		
[DPEN#]		[BF1]	X34	PHIT#	AA03		
PICD1	L35	STPCLK#	V34	PHITM#	AC03		
[APICEN]							

$V_{CC}2$ [4]				
A17	A07	Q01	AA01	AN11
A15	G01	S01	AC01	AN13
A13	J01	U01	AE01	AN15
A11	L01	W01	AG01	AN17
A09	N01	Y01	AN09	AN19

$V_{CC}3$						
A19	A27	J37	Q37	U37	AC37	AN27
A21	A29	L37	S37	W37	AE37	AN25
A23	E37	L33	T34	Y37	AG37	AN23
A25	G37	N37	U33	AA37	AN29	AN21

Table 4-1. Pin Cross Reference by Pin Name (Contd.)

Vss								
B06	B18	H02	P02	U35	Z36	AF36	AM12	AM24
B08	B20	H36	P36	V02	AB02	AH02	AM14	AM26
B10	B22	K02	R02	V36	AB36	AJ37	AM16	AM28
B12	B24	K36	R36	X02	AD02	AL37	AM18	AM30
B14	B26	M02	T02	X36	AD36	AM08	AM20	AN37
B16	B28	M36	T36	Z02	AF02	AM10	AM22	
NC								
A37			S35			AL19		
R34			W33			AN35		
S33			W35			—		
INC								
A03	B02		C01		AN01	AN03	AN05	

NOTES:

1. The VCC2DET# pin is defined only for the Pentium® processor with MMX™ technology. This pin is an INC on the Pentium processor (75/90/100/120/133/150/166/200).
2. The FRCMC# pin is defined only for the Pentium processor (75/90/100/120/133/150/166/200). This pin should be left as a "NC" or tied to $V_{CC}3$ via an external pullup resistor on the Pentium processor with MMX technology.
3. PICCLK and CLK are 3.3V tolerant on the Pentium processor with MMX technology and 5.0V tolerant on the Pentium processor (75/90/100/120/133/150/166/200).
4. The Pentium processor with MMX technology is a split-plane processor; $V_{CC}2$ and $V_{CC}3$ operate at different voltages for split-place processors. The Pentium processor (75/90/100/120/133/150/166/200) is a unified-plane processor; $V_{CC}2$ and $V_{CC}3$ operate at the same voltage for unified-plane processors. The $V_{CC}2$ and $V_{CC}3$ pins are collectively defined as V_{CC} pins on the Pentium processor (75/90/100/120/133/150/166/200).

4.2. DESIGN NOTES

For reliable operation, always connect unused inputs to an appropriate signal level. Unused active low inputs should be connected to V_{CC}. Unused active high inputs should be connected to V_{SS} (GND).

No Connect (NC) pins must remain unconnected. Connection of NC or INC (internal no-connect) pins may result in component failure or incompatibility with processor steppings.

4.3. QUICK PIN REFERENCE

This section gives a brief functional description of each of the pins. **Note that all input pins must meet their AC/DC specifications to guarantee proper functional behavior.**

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. When a # symbol is not present after the signal name, the signal is active, or asserted, at the high voltage level. Square brackets around a signal name indicate that the signal is defined only at RESET. See Chapter 7 for the timing requirements of these signals.

The following pins become I/O pins when either two Pentium processors with MMX technology or two Pentium processors (75/90/100/120/133/150/166/200) are operating in a dual processing environment:

ADS#, BE4#, CACHE#, HIT#, HITM#, HLDA#, LOCK#, M/IO#, D/C#, W/R#, SCYC

Please refer to Chapter 17 for information on how to connect the Pentium processor pins if an upgrade socket is designed in the system.

Table 4-2. Quick Pin Reference

Symbol	Type*	Name and Function
A20M#	I	When the **address bit 20 mask** pin is asserted, the Pentium® processor emulates the address wraparound at 1 Mbyte which occurs on the 8086 by masking physical address bit 20 (A20) before performing a lookup to the internal caches or driving a memory cycle on the bus. The effect of A20M# is undefined in protected mode. A20M# must be asserted only when the processor is in real mode. A20M# is internally masked by the Pentium processor when configured as a Dual processor.
A31-A3	I/O	As outputs, the **address** lines of the processor along with the byte enables define the physical area of memory or I/O accessed. The external system drives the inquire address to the processor on A31-A5.
ADS#	O	The **address strobe** indicates that a new valid bus cycle is currently being driven by the Pentium processor.
ADSC#	O	The **address strobe (copy)** is functionally identical to ADS#.
AHOLD	I	In response to the assertion of **address hold**, the Pentium processor will stop driving the address lines (A31-A3) and AP in the next clock. The rest of the bus will remain active so data can be returned or driven for previously issued bus cycles.
AP	I/O	**Address parity** is driven by the Pentium processor with even parity information on all Pentium processor generated cycles in the same clock that the address is driven. Even parity must be driven back to the Pentium processor during inquire cycles on this pin in the same clock as EADS# to ensure that correct parity check status is indicated by the Pentium processor.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
APCHK#	O	The **address parity check** status pin is asserted two clocks after EADS# is sampled active if the Pentium processor has detected a parity error on the address bus during inquire cycles. APCHK# will remain active for one clock each time a parity error is detected (including during dual processing private snooping).
[APICEN] PICD1	I	**Advanced Programmable Interrupt Controller Enable** enables or disables the on-chip APIC interrupt controller. If sampled high at the falling edge of RESET, the APIC is enabled. APICEN shares a pin with the PICD1 signal.
BE7#-BE4# BE3#-BE0#	O I/O	The **byte enable** pins are used to determine which bytes must be written to external memory or which bytes were requested by the CPU for the current cycle. The byte enables are driven in the same clock as the address lines (A31-3). Additionally, the lower 4-byte enables (BE3#-BE0#) are used on the Pentium processor as APIC ID inputs and are sampled at RESET. In dual processing mode, BE4# is used as an input during Flush cycles. **NOTE:** BE4# is an input/output pin on the Pentium processor (75/90/100/120/133/ 150/166/200)
BF[1:0]	I	The **bus frequency** pins determine the bus-to-core frequency ratio. BF[1:0] are sampled at RESET, and cannot be changed until another non-warm (1 ms) assertion of RESET. Additionally, BF[1:0] must not change values while RESET is active. See Table 4-3 for Bus Frequency Selections.
BOFF#	I	The **backoff** input is used to abort all outstanding bus cycles that have not yet completed. In response to BOFF#, the Pentium processor will float all pins normally floated during bus hold in the next clock. The processor remains in bus hold until BOFF# is negated, at which time the Pentium processor restarts the aborted bus cycle(s) in their entirety.
BP[3:2] PM/BP[1:0]	O	The **breakpoint** pins (BP3-0) correspond to the debug registers, DR3-DR0. These pins externally indicate a breakpoint match when the debug registers are programmed to test for breakpoint matches. BP1 and BP0 are multiplexed with the **performance monitoring** pins (PM1 and PM0). The PB1 and PB0 bits in the Debug Mode Control Register determine if the pins are configured as breakpoint or performance monitoring pins. The pins come out of RESET configured for performance monitoring.
BRDY#	I	The **burst ready** input indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted the Pentium processor data in response to a write request. This signal is sampled in the T2, T12 and T2P bus states.
BRDYC#	I	The **burst ready (copy)** is functionally identical to BRDY#.
BREQ	O	The **bus request** output indicates to the external system that the Pentium processor has internally generated a bus request. This signal is always driven whether or not the Pentium processor is driving its bus.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
BUSCHK#	I	The **bus check** input allows the system to signal an unsuccessful completion of a bus cycle. If this pin is sampled active, the Pentium processor will latch the address and control signals in the machine check registers. If, in addition, the MCE bit in CR4 is set, the Pentium processor will vector to the machine check exception. **NOTE:** To assure that BUSCHK# will always be recognized, STPCLK# must be deasserted any time BUSCHK# is asserted by the system, before the system allows another external bus cycle. If BUSCHK# is asserted by the system for a snoop cycle while STPCLK# remains asserted, usually (if MCE=1) the processor will vector to the exception after STPCLK# is deasserted. But if another snoop to the same line occurs during STPCLK# assertion, the processor can lose the BUSCHK# request.
CACHE#	O	For Pentium processor-initiated cycles the **cache** pin indicates internal cacheability of the cycle (if a read), and indicates a burst write back cycle (if a write). If this pin is driven inactive during a read cycle, the Pentium processor will not cache the returned data, regardless of the state of the KEN# pin. This pin is also used to determine the cycle length (number of transfers in the cycle).
CLK	I	The **clock** input provides the fundamental timing for the Pentium processor. Its frequency is the operating frequency of the Pentium processor external bus, and requires TTL levels. All external timing parameters except TDI, TDO, TMS, TRST#, and PICD0-1 are specified with respect to the rising edge of CLK. This pin is 3.3V tolerant on the Pentium processor with MMX™ technology and 5.0V tolerant on the Pentium processor (75/90/100/120/133/150/166/200). **NOTE:** It is recommended that CLK begin toggling within 150 ms after V_{CC} reaches its proper operating level. This recommendation is to ensure long-term reliability of the device.
CPUTYP	I	**CPU type** distinguishes the Primary processor from the Dual processor. In a single processor environment, or when the Pentium processor is acting as the Primary processor in a dual processing system, CPUTYP should be strapped to V_{SS}. The Dual processor should have CPUTYP strapped to V_{CC} (V_{CC}3).
D/C#	O	The **data/code** output is one of the primary bus cycle definition pins. It is driven valid in the same clock as the ADS# signal is asserted. D/C# distinguishes between data and code or special cycles.
D/P#	O	The **dual/primary** processor indication. The Primary processor drives this pin low when it is driving the bus, otherwise it drives this pin high. D/P# is always driven. D/P# can be sampled for the current cycle with ADS# (like a status pin). This pin is defined only on the Primary processor. Dual processing is supported in a system only if both processors are operating at identical core and bus frequencies. Within these restrictions, two processors of different steppings may operate together in a system.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
D63-D0	I/O	These are the 64 **data lines** for the processor. Lines D7-D0 define the least significant byte of the data bus; lines D63-D56 define the most significant byte of the data bus. When the CPU is driving the data lines, they are driven during the T2, T12, or T2P clocks for that cycle. During reads, the CPU samples the data bus when BRDY# is returned.
DP7-DP0	I/O	These are the **data parity** pins for the processor. There is one for each byte of the data bus. They are driven by the Pentium processor with even parity information on writes in the same clock as write data. Even parity information must be driven back to the Pentium processor on these pins in the same clock as the data to ensure that the correct parity check status is indicated by the Pentium processor. DP7 applies to D63-56, DP0 applies to D7-0.
[DPEN#] PICD0	I/O	**Dual processing enable** is an output of the Dual processor and an input of the Primary processor. The Dual processor drives DPEN# low to the Primary processor at RESET to indicate that the Primary processor should enable dual processor mode. DPEN# may be sampled by the system at the falling edge of RESET to determine if the dual-processor socket is occupied. DPEN# is multiplexed with PICD0.
EADS#	I	This signal indicates that a valid **external address** has been driven onto the Pentium processor address pins to be used for an inquire cycle.
EWBE#	I	The **external write buffer empty** input, when inactive (high), indicates that a write cycle is pending in the external system. When the Pentium processor generates a write, and EWBE# is sampled inactive, the Pentium processor will hold off all subsequent writes to all E- or M-state lines in the data cache until all write cycles have completed, as indicated by EWBE# being active.
FERR#	O	The **floating point error** pin is driven active when an unmasked floating point error occurs. FERR# is similar to the ERROR# pin on the Intel387™ math coprocessor. FERR# is included for compatibility with systems using DOS type floating point error reporting. FERR# is never driven active by the Dual processor.
FLUSH#	I	When asserted, the **cache flush** input forces the Pentium processor to write back all modified lines in the data cache and invalidate its internal caches. A Flush Acknowledge special cycle will be generated by the Pentium processor indicating completion of the write back and invalidation. If FLUSH# is sampled low when RESET transitions from high to low, tristate test mode is entered. If two Pentium processors are operating in dual processing mode and FLUSH# is asserted, the Dual processor will perform a flush first (without a flush acknowledge cycle), then the Primary processor will perform a flush followed by a flush acknowledge cycle. **NOTE:** If the FLUSH# signal is asserted in dual processing mode, it must be deasserted at least one clock prior to BRDY# of the FLUSH Acknowledge cycle to avoid DP arbitration problems.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
FRCMC#	I	The **functional redundancy checking master/checker** mode input is used to determine whether the Pentium processor is configured in master mode or checker mode. When configured as a master, the Pentium processor drives its output pins as required by the bus protocol. When configured as a checker, the Pentium processor tristates all outputs (except IERR#, PICD0, PICD1 and TDO) and samples the output pins. The configuration as a master/checker is set after RESET and may not be changed other than by a subsequent RESET. Functional Redundancy Checking is not supported on the Pentium processor with MMX technology. The FRCMC# pin is defined only for the Pentium processor (75/90/100/120/133/150/166/200). This pin should be left as a “NC” or tied to $V_{CC}3$ via an external pullup resistor on the Pentium processor with MMX technology.
HIT#	O	The **hit** indication is driven to reflect the outcome of an inquire cycle. If an inquire cycle hits a valid line in either the Pentium processor data or instruction cache, this pin is asserted two clocks after EADS# is sampled asserted. If the inquire cycle misses the Pentium processor cache, this pin is negated two clocks after EADS#. This pin changes its value only as a result of an inquire cycle and retains its value between the cycles.
HITM#	O	The **hit to a modified line** output is driven to reflect the outcome of an inquire cycle. It is asserted after inquire cycles which resulted in a hit to a modified line in the data cache. It is used to inhibit another bus master from accessing the data until the line is completely written back.
HLDA	O	The **bus hold acknowledge** pin goes active in response to a hold request driven to the processor on the HOLD pin. It indicates that the Pentium processor has floated most of the output pins and relinquished the bus to another local bus master. When leaving bus hold, HLDA will be driven inactive and the Pentium processor will resume driving the bus. A pending bus cycle will be driven in the same clock that HLDA is de-asserted by the Pentium processor (75/90/100/120/133/150/166/200) and one clock after HLDA is deasserted by the Pentium processor with MMX technology.
HOLD	I	In response to the **bus hold request**, the Pentium processor will float most of its output and input/output pins and assert HLDA after completing all outstanding bus cycles. The Pentium processor will maintain its bus in this state until HOLD is de-asserted. HOLD is not recognized during LOCK cycles. The Pentium processor will recognize HOLD during reset.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
IERR#	O	The **internal error** pin is used to indicate two types of errors, internal parity errors and functional redundancy errors. If a parity error occurs on a read from an internal array, the Pentium processor will assert the IERR# pin for one clock and then shutdown. If the Pentium processor is configured as a checker and a mismatch occurs between the value sampled on the pins and the corresponding value computed internally, the Pentium processor will assert IERR# two clocks after the mismatched value is returned. Note: Functional Redundancy Checking is not supported on Pentium processors with MMX technology.
IGNNE#	I	This is the **ignore numeric error** input. This pin has no effect when the NE bit in CR0 is set to 1. When the CR0.NE bit is 0, and the IGNNE# pin is asserted, the Pentium processor will ignore any pending unmasked numeric exception and continue executing floating-point instructions for the entire duration that this pin is asserted. When the CR0.NE bit is 0, IGNNE# is not asserted, a pending unmasked numeric exception exists (SW.ES = 1), and the floating point instruction is one of FINIT, FCLEX, FSTENV, FSAVE, FSTSW, FSTCW, FENI, FDISI, or FSETPM, the Pentium processor will execute the instruction in spite of the pending exception. When the CR0.NE bit is 0, IGNNE# is not asserted, a pending unmasked numeric exception exists (SW.ES = 1), and the floating-point instruction is one other than FINIT, FCLEX, FSTENV, FSAVE, FSTSW, FSTCW, FENI, FDISI, or FSETPM, the Pentium processor will stop execution and wait for an external interrupt. IGNNE# is internally masked when the Pentium processor is configured as a Dual processor.
INIT	I	The Pentium processor **initialization** input pin forces the Pentium processor to begin execution in a known state. The processor state after INIT is the same as the state after RESET except that the internal caches, write buffers, and floating point registers retain the values they had prior to INIT. INIT may NOT be used in lieu of RESET after power-up. If INIT is sampled high when RESET transitions from high to low, the Pentium processor will perform built-in self test prior to the start of program execution.
INTR/LINT0	I	An active **maskable interrupt** input indicates that an external interrupt has been generated. If the IF bit in the EFLAGS register is set, the Pentium processor will generate two locked interrupt acknowledge bus cycles and vector to an interrupt handler after the current instruction execution is completed. INTR must remain active until the first interrupt acknowledge cycle is generated to assure that the interrupt is recognized. If the local APIC is enabled, this pin becomes LINT0.
INV	I	The **invalidation** input determines the final cache line state (S or I) in case of an inquire cycle hit. It is sampled together with the address for the inquire cycle in the clock EADS# is sampled active.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
KEN#	I	The **cache enable** pin is used to determine whether the current cycle is cacheable or not and is consequently used to determine cycle length. When the Pentium processor generates a cycle that can be cached (CACHE# asserted) and KEN# is active, the cycle will be transformed into a burst line fill cycle.
LINT0/INTR	I	If the APIC is enabled, this pin is **local interrupt 0**. If the APIC is disabled, this pin is INTR.
LINT1/NMI	I	If the APIC is enabled, this pin is **local interrupt 1**. If the APIC is disabled, this pin is NMI.
LOCK#	O	The **bus lock** pin indicates that the current bus cycle is locked. The Pentium processor will not allow a bus hold when LOCK# is asserted (but AHOLD and BOFF# are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the BRDY# is returned for the last locked bus cycle. LOCK# is guaranteed to be de-asserted for at least one clock between back-to-back locked cycles.
M/IO#	O	The **memory/input-output** is one of the primary bus cycle definition pins. It is driven valid in the same clock as the ADS# signal is asserted. M/IO# distinguishes between memory and I/O cycles.
NA#	I	An active **next address** input indicates that the external memory system is ready to accept a new bus cycle although all data transfers for the current cycle have not yet completed. The Pentium processor will issue ADS# for a pending cycle two clocks after NA# is asserted. The Pentium processor supports up to 2 outstanding bus cycles.
NMI/LINT1	I	The **non-maskable interrupt** request signal indicates that an external non-maskable interrupt has been generated. If the local APIC is enabled, this pin becomes LINT1.
PBGNT#	I/O	**Private bus grant** is the grant line that is used when two Pentium processors are configured in dual processing mode, in order to perform private bus arbitration. PBGNT# should be left unconnected if only one Pentium processor exists in a system.
PBREQ#	I/O	**Private bus request** is the request line that is used when two Pentium processor are configured in dual processing mode, in order to perform private bus arbitration. PBREQ# should be left unconnected if only one Pentium processor exists in a system.
PCD	O	The **page cache disable** pin reflects the state of the PCD bit in CR3, the Page Directory Entry, or the Page Table Entry. The purpose of PCD is to provide an external cacheability indication on a page by page basis.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
PCHK#	O	The **parity check** output indicates the result of a parity check on a data read. It is driven with parity status two clocks after BRDY# is returned. PCHK# remains low one clock for each clock in which a parity error was detected. Parity is checked only for the bytes on which valid data is returned. When two Pentium processors are operating in dual processing mode, PCHK# may be driven two or three clocks after BRDY# is returned.
PEN#	I	The **parity enable** input (along with CR4.MCE) determines whether a machine check exception will be taken as a result of a data parity error on a read cycle. If this pin is sampled active in the clock a data parity error is detected, the Pentium processor will latch the address and control signals of the cycle with the parity error in the machine check registers. If, in addition, the machine check enable bit in CR4 is set to “1”, the Pentium processor will vector to the machine check exception before the beginning of the next instruction.
PHIT#	I/O	**Private hit** is a hit indication used when two Pentium processors are configured in dual processing mode, in order to maintain local cache coherency. PHIT# should be left unconnected if only one Pentium processor exists in a system.
PHITM#	I/O	**Private modified hit** is a hit on a modified cache line indication used when two Pentium processors are configured in dual processing mode, in order to maintain local cache coherency. PHITM# should be left unconnected if only one Pentium processor exists in a system.
PICCLK	I	The APIC interrupt controller serial data bus clock is driven into the **programmable interrupt controller clock** input of the Pentium processor. This pin is 3.3V tolerant on the Pentium processor with MMX technology, and 5.0V tolerant on the Pentium processor (75/90/100/120/133/150/166/200).
PICD0-1 [DPEN#] [APICEN]	I/O	**Programmable interrupt controller data lines 0-1** of the Pentium processor comprise the data portion of the APIC 3-wire bus. They are open-drain outputs that require external pull-up resistors. These signals are multiplexed with DPEN# and APICEN respectively.
PM/BP[1:0]	O	These pins function as part of the performance monitoring feature. The breakpoint 1-0 pins are multiplexed with the **performance monitoring 1-0** pins. The PB1 and PB0 bits in the Debug Mode Control Register determine if the pins are configured as breakpoint or performance monitoring pins. The pins come out of RESET configured for performance monitoring.
PRDY	O	The **probe ready** output pin is provided for use with the Intel debug port described in the “Debugging” chapter.
PWT	O	The **page write through** pin reflects the state of the PWT bit in CR3, the Page Directory Entry, or the Page Table Entry. The PWT pin is used to provide an external write back indication on a page-by-page basis.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
R/S#	I	The **run/stop** input is provided for use with the Intel debug port described in the "Debugging" chapter.
RESET	I	**RESET** forces the Pentium processor to begin execution at a known state. All the Pentium processor internal caches will be invalidated upon the RESET. Modified lines in the data cache are not written back. FLUSH#, FRCMC# and INIT are sampled when RESET transitions from high to low to determine if tristate test mode or checker mode will be entered, or if BIST will be run. Note: Functional Redundancy Checking is not supported on Pentium processors with MMX technology.
SCYC	O	The **split cycle** output is asserted during misaligned LOCKed transfers to indicate that more than two cycles will be locked together. This signal is defined for locked cycles only. It is undefined for cycles which are not locked.
SMI#	I	The **system management interrupt** causes a system management interrupt request to be latched internally. When the latched SMI# is recognized on an instruction boundary, the processor enters System Management Mode.
SMIACT#	O	An active **system management interrupt active** output indicates that the processor is operating in System Management Mode.
STPCLK#	I	Assertion of the **stop clock** input signifies a request to stop the internal clock of the Pentium processor, thereby causing the core to consume less power. When the CPU recognizes STPCLK#, the processor will stop execution on the next instruction boundary, unless superseded by a higher priority interrupt, and generate a stop grant acknowledge cycle. When STPCLK# is asserted, the Pentium processor will still respond to interprocessor and external snoop requests.
TCK	I	The **testability clock** input provides the clocking function for the Pentium processor boundary scan in accordance with the IEEE Boundary Scan interface (Standard 1149.1). It is used to clock state information and data into and out of the Pentium processor during boundary scan.
TDI	I	The **test data input** is a serial input for the test logic. TAP instructions and data are shifted into the Pentium processor on the TDI pin on the rising edge of TCK when the TAP controller is in an appropriate state.
TDO	O	The **test data output** is a serial output of the test logic. TAP instructions and data are shifted out of the Pentium processor on the TDO pin on TCK's falling edge when the TAP controller is in an appropriate state.
TMS	I	The value of the **test mode select** input signal sampled at the rising edge of TCK controls the sequence of TAP controller state changes.
TRST#	I	When asserted, the **test reset** input allows the TAP controller to be asynchronously initialized.

Table 4-2. Quick Pin Reference (Contd.)

Symbol	Type*	Name and Function
V_{CC}	I	The Pentium processor (75/90/100/120/133/150/166/200) has 53 3.3V **power** inputs.
$V_{CC}2$		The Pentium processor with MMX technology has 25 2.8V **power** inputs.
$V_{CC}3$		The Pentium processor with MMX technology has 28 3.3V **power** inputs.
VCC2DET#	O	**$V_{CC}2$ detect** is defined only on the Pentium processor with MMX technology and can be used in flexible motherboard implementations to configure the voltage output set-point appropriately for the $V_{CC}2$ inputs of the processor.
V_{SS}	I	The Pentium processor has 53 **ground** inputs.
W/R#	O	**Write/read** is one of the primary bus cycle definition pins. It is driven valid in the same clock as the ADS# signal is asserted. W/R# distinguishes between write and read cycles.
WB/WT#	I	The **write back/write through** input allows a data cache line to be defined as write back or write through on a line-by-line basis. As a result, it determines whether a cache line is initially in the S or E state in the data cache.

NOTE:

* The pins are classified as Input or Output based on their function in Master Mode. See the Functional Redundancy Checking section in the “Error Detection” chapter for further information.

Each Pentium processor is specified to operate within a single bus-to-core ratio and a specific minimum to maximum bus frequency range (corresponding to a minimum to maximum core frequency range). Operation in other bus-to-core ratios or outside the specified operating frequency range is not supported. For example, the 150 MHz Pentium processor does not operate beyond the 60 MHz bus frequency and only supports the 2/5 bus-to-core ratio; it does not support the 1/3, 1/2, or 2/3 bus-to-core ratios. Table 4-3 clarifies and summarizes these specifications.

Table 4-3. Bus to Core Frequency Ratios for the Pentium® Processor

BF1	BF0	Pentium® Processor (75/90/100/120/133/150/166/200) Bus/Core Ratio	Pentium Processor with MMX™ Technology Bus/Core Ratio[4]	Max Bus/Core Frequency (MHz)	Min Bus/Core Frequency (MHz)
0	1	1/3	1/3	66/200	33/100
0	0	2/5	2/5	66/166	33/83
0	0	2/5	2/5	60/150	30/75
1	0	1/2	1/2 [2]	66/133	33/66
1	0	1/2	1/2 [2]	60/120	30/60
1	0	1/2	1/2 [2]	50/100[3]	25/50
1	1	2/3 [1]	reserved	66/100[3]	33/50
1	1	2/3 [1]	reserved	60/90	30/45
1	1	2/3 [1]	reserved	50/75	25/37.5

NOTES:

1. This is the default bus fraction for the Pentium® processor (75/90/100/120/133/150/166/200). If the BF pins are left floating, the processor will be configured for the 2/3 bus to core frequency ratio.
2. This is the default bus fraction for the Pentium processor with MMX™ technology. If the BF pins are left floating, the processor will be configured for the 1/2 bus to core frequency ratio.
3. The 100 MHz (Max Core Frequency) Pentium processors can be operated in both 1/2 and 2/3 Bus/Core Ratios.
4. Currently, the desktop Pentium processor with MMX technology supports 66/200 and 66/166 operation.

4.3.1. Pin Reference Tables

Table 4-4. Output Pins

Name	Active Level	When Floated
ADS#[1]	Low	Bus Hold, BOFF#
ADSC#	Low	Bus Hold, BOFF#
APCHK#	Low	
BE7#-BE4# [5]	Low	Bus Hold, BOFF#
BREQ	High	
CACHE#[1]	Low	Bus Hold, BOFF#
D/P#[2]	n/a	
FERR#[2]	Low	
HIT#[1]	Low	
HITM#[1,4]	Low	
HLDA[1]	High	
IERR#	Low	
LOCK#[1]	Low	Bus Hold, BOFF#
M/IO#[1], D/C#[1], W/R#[1]	n/a	Bus Hold, BOFF#
PCHK#	Low	
BP3-2, PM1/BP1, PM0/BP0	High	
PRDY	High	
PWT, PCD	High	Bus Hold, BOFF#
SCYC[1]	High	Bus Hold, BOFF#
SMIACT#	Low	
TDO	n/a	All states except Shift-DR and Shift-IR
VCC2DET#[3]	Low	

NOTES:

All output and input/output pins are floated during tristate test mode (except TCO) and checker mode (except IERR#, PICD0, PICD1 and TDO).

1. These are I/O signals when two Pentium® processors are operating in dual processing mode.
2. These signals are undefined when the CPU is configured as a Dual Processor.
3. VCC2DET# is defined only for the Pentium processor with MMX™ technology.
4. The HITM# pin has an internal pull-up resistor.
5. BE4# is an input/output pin on the Pentium processor (75/90/100/120/133/150/166/200). BE4# has an internal pulldown during RESET only.

Table 4-5. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Resistor	Qualified
A20M#[1]	Low	Asynchronous		
AHOLD	High	Synchronous		
APICEN	High	Synchronous/RESET	Pullup	
BF0	High	Synchronous/RESET	Pullup/Pulldown[2]	
BF1	High	Synchronous/RESET	Pullup	
BOFF#	Low	Synchronous		
BRDY#	Low	Synchronous	Pullup	Bus State T2, T12, T2P
BRDYC#	Low	Synchronous	Pullup	Bus State T2, T12, T2P
BUSCHK#	Low	Synchronous	Pullup	BRDY#
CLK	n/a			
CPUTYP	High	Synchronous/RESET	Pulldown	
EADS#	Low	Synchronous		
EWBE#	Low	Synchronous		BRDY#
FLUSH#	Low	Asynchronous		
FRCMC#[3]	Low	Asynchronous	Pullup	
HOLD	High	Synchronous		
IGNNE#[1]	Low	Asynchronous		
INIT	High	Asynchronous		
INTR	High	Asynchronous		
INV	High	Synchronous		EADS#
LINT[1:0]	High	Asynchronous		APICEN at RESET
KEN#	Low	Synchronous		First BRDY#/NA#
NA#	Low	Synchronous		Bus State T2,TD,T2P
NMI	High	Asynchronous		
PEN#	Low	Synchronous		BRDY#
PICCLK	High	Asynchronous	Pullup	
R/S#	n/a	Asynchronous	Pullup	

Table 4-5. Input Pins (Contd.)

Name	Active Level	Synchronous/ Asynchronous	Internal Resistor	Qualified
RESET	High	Asynchronous		
SMI#	Low	Asynchronous	Pullup	
STPCLK#	Low	Asynchronous	Pullup	
TCK	n/a		Pullup	
TDI	n/a	Synchronous/TCK	Pullup	TCK
TMS	n/a	Synchronous/TCK	Pullup	TCK
TRST#	Low	Asynchronous	Pullup	
WB/WT#	n/a	Synchronous		First BRDY#/NA#

NOTES:

1. These pins are undefined when the CPU is configured as a Dual processor.
2. BF0 has an internal pulldown on the Pentium® processor with MMX™ technology and an internal pullup on the Pentium processor (75/90/100/120/133/150/166/200).
3. FRCMC# is defined only for the Pentium procesor (75/90/100/120/133/150/166/200).

Table 4-6. Input/Output Pins[1]

Name	Active Level	When Floated	Qualified (when an input)	Internal Resistor
A31-A3	n/a	Address Hold, Bus Hold, BOFF#	EADS#	
AP	n/a	Address Hold, Bus Hold, BOFF#	EADS#	
BE3#-BE0# [3]	Low	Address Hold, Bus Hold, BOFF#	RESET	Pulldown[3]
D63-D0	n/a	Bus Hold, BOFF#	BRDY#	
DP7-DP0	n/a	Bus Hold, BOFF#	BRDY#	
DPEN#	low		RESET	Pullup
PICD0	n/a			Pullup
PICD1	n/a			Pulldown

NOTES:

1. All output and input/output pins are floated during tristate test mode (except TDO) and checker mode (except IERR#, PICD0, PICD1 and TDO).
2. BE4# is an input/output pin on the Pentium® processor (75/90/100/120/133/150/166/200).
3. BE4#-BE0# have pulldowns during RESET only.

Table 4-7. Inter-Processor Input/Output Pins

Name	Active Level	Internal Resistor
PHIT#	Low	Pullup
PHITM#	Low	Pullup
PBGNT#	Low	Pullup
PBREQ#	Low	Pullup

NOTE:

For proper inter-processor operation, the system cannot load these signals.

4.3.2. Pin Grouping According to Function

Table 4-8 organizes the pins with respect to their function.

Table 4-8. Pin Functional Grouping

Function	Pins
Clock	CLK
Initialization	RESET, INIT, BF1–BF0
Address Bus	A31-A3, BE7#–BE0#
Address Mask	A20M#
Data Bus	D63-D0
Address Parity	AP, APCHK#
APIC Support	PICCLK, PICD0-1
Data Parity	DP7-DP0, PCHK#, PEN#
Internal Parity Error	IERR#
System Error	BUSCHK#
Bus Cycle Definition	M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK#
Bus Control	ADS#, ADSC#, BRDY#, BRDYC#, NA#
Page Cacheability	PCD, PWT
Cache Control	KEN#, WB/WT#
Cache Snooping/Consistency	AHOLD, EADS#, HIT#, HITM#, INV
Cache Flush	FLUSH#
Write Ordering	EWBE#
Bus Arbitration	BOFF#, BREQ, HOLD, HLDA
Dual Processing Private Bus Control	PBGNT#, PBREQ#, PHIT#, PHITM#
Interrupts	INTR, NMI
Floating Point Error Reporting	FERR#, IGNNE#
System Management Mode	SMI#, SMIACT#
Functional Redundancy Checking[1]	FRCMC#, (IERR#)
TAP Port	TCK, TMS, TDI, TDO, TRST#
Breakpoint/Performance Monitoring	PM0/BP0, PM1/BP1, BP3-2
Power Management	STPCLK#
Miscellaneous Dual Processing	CPUTYP, D/P#
Debugging	R/S#, PRDY
Voltage Detection	VCC2DET#[2]

NOTES:

1. Functional Redundancy Checking is not supported on the Pentium® processor with MMX™ technology. The FRCMC# pin is defined only for the Pentium processor (75/90/100/120/133/150/166/200). This pin should be left as a “NC” or tied to $V_{CC}3$ via an external pullup resistor on the Pentium processor with MMX technology.
2. The VCC2DET# pin is defined only for the Pentium processor with MMX technology. This pin is an INC on the Pentium processor (75/90/100/120/133/150/166/200).

5

Hardware Interface

CHAPTER 5
HARDWARE INTERFACE

5.1. DETAILED PIN DESCRIPTIONS

This chapter describes the pins of the Pentium processor that interface to the system. Both the Pentium processor (75/90/100/120/133/150/166/200) and the Pentium processor with MMX technology have the same logical hardware interface. The Pentium processor with MMX technology has one extra signal, VCC2DET#.

The Pentium processor, when operating in dual processing mode, modifies the functionality of the following signals:

- A20M#, ADS#, BE4#-BE0#, CACHE#, D/C#, FERR#, FLUSH#, HIT#, HITM#, HLDA, IGNNE#, LOCK#, M/IO#, PCHK#, RESET, SCYC, SMIACT#, W/R#

5.1.1. A20M#

A20M#	Address 20 Mask
	Used to emulate the 1 Mbyte address wraparound on the 8086
	Asynchronous Input

Signal Description

When the address 20 mask input is asserted, the Pentium processor masks physical address bit 20 (A20) before performing a lookup to the internal caches or driving a memory cycle on the bus. A20M# is provided to emulate the address wraparound at one Mbyte which occurs on the 8086.

A20M# must only be asserted when the processor is in real mode. **The effect of asserting A20M# in protected mode is undefined** and may be implemented differently in future processors.

Inquire cycles and writebacks caused by inquire cycles are not affected by this input. Address bit A20 is not masked when an external address is driven into the Pentium processor for an inquire cycle. Note that if an OUT instruction is used to modify A20M# this will not affect previously prefetched instructions. A serializing instruction must be executed to guarantee recognition of A20M# before a specific instruction.

The Pentium processor, when configured as a Dual processor, will ignore the A20M# input.

When Sampled/Driven

A20M# is sampled on every rising clock edge. A20M# is level sensitive and active low. This pin is asynchronous, but must meet setup and hold times for recognition in any specific clock. To guarantee that A20M# will be recognized before the first ADS# after RESET, A20M# must be asserted within two clocks after the falling edge of RESET

NOTE

As the performance of Pentium processors continues to improve, a given code sequence is executed faster. As a result, some code sequences that rely upon hardware timing may fail. Specifically when a keyboard controller is used to toggle the A20M# pin and if the keyboard controller is slow in response, then at some point in a code sequence, data or code may be read from a wrong address. Therefore, it should be ensured that the keyboard controller switches the A20M# signal fast enough to match the execution speed of the processor. Software should be written to synchronize between code execution and the event of A20M# toggling.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A20	When asserted, A20M# will mask the value of address pin A20.
CPUTYP	When strapped to V_{CC}, the processor will ignore the A20M# input.

5.1.2. A31-A3

A31-A3	Address Lines
	Defines the physical area of memory or I/O accessed.
	Input/Output

Signal Description

As outputs, the Address Lines (A31-A3) along with the byte enable signals (BE7#-BE0#) form the address bus and define the physical area of memory or I/O accessed.

The Pentium processor is capable of addressing 4 gigabytes of physical memory space and 64K bytes of I/O address space.

As inputs, the address bus lines A31-A5 are used to drive addresses back into the processor to perform inquire cycles. Since inquire cycles affect an entire 32-byte line, the logic values of A4 and A3 are not used for the hit/miss decision, however A4 and A3 must be at valid logic level and meet setup and hold times during inquire cycles.

When Sampled/Driven

When an output, the address is driven in the same clock as ADS#. The address remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#, or until AHOLD is asserted.

When an input, the address must be returned to the processor to meet setup and hold times in the clock EADS# is sampled asserted.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A20M#	Causes address pin A20 to be masked.
ADS#	A31-A3 are driven with ADS# (except when a external inquire cycle causes a writeback before AHOLD is deasserted, see the Bus Functional Description chapter).
AHOLD	A31-A3 are floated one clock after AHOLD is asserted.
AP	Even address parity is driven/sampled with the address bus on AP.
APCHK#	The status of the address parity check is driven on the APCHK# pin.
BE7#-BE0#	Completes the definition of the physical area of memory or I/O accessed.
BOFF#	A31-A3 are floated one clock after BOFF# is asserted.
EADS#	A31-A5 are sampled with EADS# during inquire cycles.
HIT#	HIT# is driven to indicate whether the inquire address driven on A31-A5 is valid in an internal cache.
HITM#	HITM# is driven to indicate whether the inquire address driven on A31-A5 is in the modified state in the data cache.
HLDA	A31-A3 are floated when HLDA is asserted.
INV	INV determines if the inquire address driven to the processor on A31-A5 should be invalidated or marked as shared if it is valid in an internal cache.

5.1.3. ADS#

ADS#	Address Strobe
	Indication that a new valid bus cycle is currently being driven by the processor.
	Synchronous Input/Output

Signal Description

The Address Strobe output indicates that a new valid bus cycle is currently being driven by the Pentium processor. The following pins are driven to their valid level in the clock ADS# is asserted: A31-A3, AP, BE7#-0#, CACHE#, LOCK#, M/IO#, W/R#, D/C#, SCYC, PWT, PCD.

ADS# is used by external bus circuitry as the indication that the processor has started a bus cycle. The external system may sample the bus cycle definition pins on the next rising edge of the clock after ADS# is driven active.

ADS# floats during bus HOLD and BOFF#. ADS# is not driven low to begin a bus cycle while AHOLD is asserted unless the cycle is a writeback due to an external invalidation. An active (floating low) ADS# in the clock after BOFF# is asserted should be ignored by the system.

This signal is normally identical to the ADSC# output. When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

ADS# is driven active in the first clock of a bus cycle and is driven inactive in the second and subsequent clocks of the cycle. ADS# is driven inactive when the bus is idle.

This signal becomes an Input/Output when two Pentium processors are operating together in Dual Processing Mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADSC#	ADS# is identical to the ADSC# output.
APCHK#	When operating in dual processing mode, APCHK# is driven in response to ADS# for a private snoop.
D/P#	When operating in dual processing mode, D/P# should be sampled with an active ADS#.
SMIACT#	When operating in dual processing mode, SMIACT# should be sampled with an active ADS# and qualified by D/P#.

5.1.4. ADSC#

ADSC#	Additional Address Strobe
	Indicates that a new valid bus cycle is currently being driven by the processor.
	Synchronous Output

Signal Description

This signal is identical to the ADS# output. This signal can be used to relieve tight board timings by easing the load on the Address Strobe signal.

When Sampled/Driven

Refer to the ADS# signal description.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	ADSC# is identical to the ADS# output.

5.1.5. AHOLD

AHOLD	Address Hold
	Floats the address bus so an inquire cycle can be driven to the Pentium® processor.
	Synchronous Input

Signal Description

In response to the Address Hold request input the Pentium processor will stop driving A31-A3 and AP in the next clock. This pin is intended to be used for running inquire cycles to the Pentium processor. AHOLD allows another bus master to drive the Pentium processor address bus with the address for an inquire cycle. Since inquire cycles affect the entire cache line, although A31-A3 are floated during AHOLD, only A31-A5 are used by the Pentium processor for inquire cycles (and parity checking). Address pins 3 and 4 are logically ignored during inquire cycles but must be at a valid logic level when sampled.

While AHOLD is active, the address bus will be floated, but the remainder of the bus can remain active. For example, data can be returned for a previously driven bus cycle when AHOLD is active. In general, the Pentium processor will not issue a bus cycle (ADS#) while AHOLD is active; the *only* exception to this is that writeback cycles due to an external snoop will be driven while AHOLD is asserted.

Since the Pentium processor floats its bus immediately (in the next clock) in response to AHOLD, an address hold acknowledge is not required.

When AHOLD is deasserted, the Pentium processor will drive the address bus in the next clock. It is the responsibility of the system designer to prevent address bus contention. This can be accomplished by ensuring that other bus masters have stopped driving the address bus before AHOLD is deasserted. Note the restrictions to the deassertion of AHOLD discussed in the inquire cycle section of the Bus Functional Description chapter (Chapter 6).

AHOLD is recognized during RESET and INIT. Note that the internal caches are flushed as a result of RESET, so invalidation cycles run during RESET are unnecessary.

When Sampled

AHOLD is sampled on every rising clock edge, including during RESET and INIT.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A31-A3	A31-A3 are floated as a result of the assertion of AHOLD.
ADS#	ADS# will not be driven if AHOLD is asserted (except when a external inquire cycle causes a writeback before AHOLD is deasserted, see the Bus Functional Description chapter (Chapter 6)).
AP	AP is floated as a result of the assertion of AHOLD.
EADS#	EADS# is recognized while AHOLD is asserted.

5.1.6. AP

AP	Address Parity
	Bi-directional address parity pin for the address lines of processor.
	Input/Output

Signal Description

This is the bi-directional Address Parity pin for the address lines of processor. There is one address parity pin for the address lines A31-A5. Note A4 and A3 are not included in the parity determination.

When an output, AP is driven by the Pentium processor with even parity information on all Pentium processor generated cycles in the same clock as the address driven. (Even address parity means that there are an even number of HIGH outputs on A31-A5 and the AP pins.)

When an input, even parity information must be returned to the Pentium processor on this pin during inquire cycles in the same clock that EADS# is sampled asserted to insure that the correct parity check status is driven on the APCHK# output.

The value read on the AP pin does not affect program execution. The value returned on the AP pin is used only to determine even parity and drive the APCHK# output with the proper value. It is the responsibility of the system to take appropriate actions if a parity error occurs. If parity checks are not implemented in the system, AP may be connected to V_{CC} through a pull-up resistor and the APCHK# pin may be ignored.

When Sampled/Driven

When an output, AP is driven by the Pentium processor with even parity information on all Pentium processor generated cycles in the same clock as the address driven. The AP output remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#, or until AHOLD is asserted.

When an input, even parity information must be returned to the Pentium processor on this pin during inquire cycles in the same clock that EADS# is sampled asserted to guarantee that the proper value is driven on APCHK#. The AP input must be at a valid level and meet setup and hold times when sampled.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A31-A5	The AP pin is used to create even parity with the A31-A5 pins.
ADS#	AP is driven with ADS# (except when a external inquire cycle causes a write-back before AHOLD is deasserted, see the Bus Functional Description chapter).
AHOLD	AP is floated one clock after AHOLD is asserted.
APCHK#	The status of the address parity check is driven on the APCHK# output.
BOFF#	AP is floated one clock after BOFF# is asserted.
EADS#	AP is sampled with EADS# during inquire cycles.
HLDA	AP is floated when HLDA is asserted.

5.1.7. APCHK#

APCHK#	Address Parity Check
	The status of the address parity check is driven on this output.
	Asynchronous Output

Signal Description

APCHK# is asserted two clocks after EADS# is sampled active if the Pentium processor has detected a parity error on the A31-A5 during inquire cycles.

Driving APCHK# is the only effect that bad address parity has on the Pentium processor. It is the responsibility of the system to take appropriate action if a parity error occurs. If parity checks are not implemented in the system, the APCHK# pin may be ignored.

Address parity is checked during every private snoop between the Primary and Dual processors. Therefore, APCHK# may be asserted due to an address parity error during this private snoop. If an error is detected, APCHK# will be asserted 2 clocks after ADS# for one processor clock period. The system can choose to acknowledge this parity error indication at this time or do nothing.

When Sampled/Driven

APCHK# is valid for one clock and should be sampled two clocks following ADS# and EADS# assertion. At all other times it is inactive (high). APCHK# is not floated with AHOLD, HOLD, or BOFF#. The APCHK# signal is glitch free.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	When operating in dual processing mode, APCHK# is driven in response to a private snoop.
AP	Even address parity with the A31-A5 should be returned to the Pentium® processor on the AP pin. If even parity is not driven, the APCHK# pin is asserted.
A31-A5	The AP pin is used to create even parity with A31-A5. If even parity is not driven to the Pentium processor, the APCHK# pin is asserted.
EADS#	APCHK# is driven in response to an external snoop.

5.1.8. APICEN

APICEN	APIC Enable
	This pin enables the APIC on the processor.
	Synchronous Configuration Input
	Needs external pull-up resistors.

Signal Description

APICEN, if sampled high at the falling edge of RESET, enables the on-chip APIC. If it is sampled low, then the on-chip APIC is not enabled and the processor uses the interrupts as if the APIC was not present (Bypass mode).

APICEN must be driven by the system. This pin has an internal pulldown resistor and is sampled at the falling edge of RESET. When using an active circuit to override the internal pulldown resistor, the driver should have an internal effective pullup resistance of 1K ohms or less.

When Sampled/Driven

APICEN should be valid and stable two clocks before and after the falling edge of RESET.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BE3#-BE0#	When APICEN is sampled active, BE3#-BE0# are used to sample the APIC ID.
INTR/LINT0	When APICEN is sampled active, this input becomes the APIC local interrupt 0.
NMI/LINT1	When APICEN is sampled active, this input becomes the APIC local interrupt 1.
PICCLK	PICCLK must be tied or driven high when APICEN is sampled low at the falling edge of RESET.
PICD1	APICEN shares a pin with PICD1.
RESET	APICEN is sampled at the falling edge of RESET.

5.1.9. BE7#-BE0#

BE7#-BE0#	Byte Enable Outputs / APIC ID Inputs
	When operating in dual processing mode, BE4# is used to transfer information between the Dual and Primary processors during the atomic Flush operation.
	At RESET, the BE3#-BE0# pins read the APIC ID bits for the Pentium® processor.
	After RESET, these pins are byte enables and help define the physical area of memory to I/O accessed.
	BE4#: Synchronous Input/Output, Dual Processing Mode. BE3#-BE0#: Synchronous Configuration Inputs, during RESET. BE3#-BE0#: Synchronous Outputs, following RESET.

Signal Description

As outputs, the byte enable signals are used in conjunction with the address lines to provide physical memory and I/O port addresses. The byte enables are used to determine which bytes of data must be written to external memory, or which bytes were requested by the CPU for the current cycle.

- BE7# applies to D63-D56
- BE6# applies to D55-D48
- BE5# applies to D47-D40
- BE4# applies to D39-D32
- BE3# applies to D31-D24
- BE2# applies to D23-D16
- BE1# applies to D15-D8
- BE0# applies to D7-D0

In the case of cacheable reads (line fill cycles), all 8 bytes of data must be driven to the Pentium processor regardless of the state of the byte enables. If the requested read cycle is a single transfer cycle, valid data must be returned on the data lines corresponding to the active byte enables. Data lines corresponding to inactive byte enables need not be driven with valid logic levels. Even data parity is checked and driven only on the data bytes that are enabled by the byte enables.

The local APIC module on the Pentium processor loads its 4-bit APIC ID value from the four least significant byte-enable pins at the falling edge of RESET. The following table shows the four pins that comprise the APIC ID.

APIC ID Register Bit	Pin Latched at RESET
bit 24	BE0#

bit 25	BE1#
bit 26	BE2#
bit 27	BE3#

Loading the APIC ID should be done with external logic that drives the proper address at reset. If the BE3#-BE0# signals are not driven, the APIC ID value will default to 0000 for the Pentium processor and 0001 for the Dual processor.

BE[0:3]# pins establish the APIC ID for the processor and are input/output pins. These pins have strong internal pull down resistors and typically high external capacitive loading. A strong pullup on BE[0:3]# is needed to make sure that the pins reach the correct value. In addition, since these pins are also outputs, a large resistive load would degrade the signal output during normal operation. A 50 Ohm tristate driver is recommended to drive these pins during RESET only.

WARNING

An APIC ID of all 1s is an APIC special case (i.e., a broadcast) and must not be used. Since the Dual processor inverts the lowest order bit of the APIC ID placed on the lowest four BE pins, the value "1110" must not be used when operating in Dual Processing mode.

In a dual-processor configuration, the OEM socket and Socket 5/Socket 7 should have the four byte enable pairs tied together. The Primary processor will load the value seen on these four pins at RESET. The Dual processor will load the value seen on these pins and automatically invert bit 24 of the APIC ID Register. Thus, the two processors will have unique APIC ID values.

The Primary and Dual processors incorporate a mechanism to present an atomic view of the cache flush operation to the system when in dual processing mode. The Dual processor performs the cache flush operation and grants the bus to the Primary processor by PBREQ#/PBGNT# arbitration exchange. The Primary processor then flushes both of its internal caches and runs a cache flush acknowledge special cycle by asserting BE4#, to indicate to the external system that the cache line entries have been invalidated. The Dual processor halts all code execution while the Pentium processor is flushing its caches, and does not begin executing code until it recognizes the flush acknowledge special cycle. Please refer to the Bus Functional Description chapter of this volume for more details (Chapter 6).

When Sampled/Driven

As outputs, the byte enables are driven in the same clock as ADS#. The byte enables are driven with the same timing as the address bus (A31-3). The byte enables remain valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#. The byte enables do not float with AHOLD

The four least significant byte-enable bits are sampled for APIC ID at the falling edge of RESET. These pins should be valid and stable two clocks before and after the falling edge of RESET.

NOTE

Asserting the APIC ID is not specified for the rising edge of RESET. In a FRC system, the BE3#-BE0# pins must not be driven for the 2 clocks following the rising edge of RESET. The system design should drive these signals on the third clock or later.

There are strong pull down resistors on the byte enable pins internally that make it impractical to use pullup circuits to drive the APIC ID (on BE3#-BE0#) or enter Lock Step operation (with BE4#) at the falling edge of RESET. When not using the internal defaults on these pins, the value of the external pullup resistors would have to be 50 Ohms or less. For this reason it is suggested to use active drivers on these lines that would drive the byte enable pins during the falling edge of RESET; passive pullups should be avoided.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A31-A3	A31-3 and BE7#-BE0# together define the physical area of memory or I/O accessed.
ADS#	BE7#-BE0# are driven with ADS#.
APICEN	When APICEN is sampled active, BE3#-BE0# are used to sample the APIC ID.
BOFF#	BE7#-BE0# are floated one clock after BOFF# is asserted.
D63-D0	BE7#-BE0# indicate which data bytes are being requested or driven by the Pentium® processor.
DP7-DP0	Even data parity is checked/driven only on the data bytes enabled by BE7#-BE0#.
HLDA	BE7#-BE0# are floated when HLDA is asserted.
RESET	During reset the BE3#-BE0# pins are sampled to determine the APIC ID. Following RESET, they function as byte-enable outputs.

5.1.10. BF1-0

BF1-0	Bus to Core Frequency Ratio
	Used to configure processor bus-to-core frequency ratio.
	Asynchronous Input

Signal Description

The BF[1:0] pins determine whether the processor will operate at a 1/2, 2/3, 2/5, or 1/3 I/O bus to core frequency ratio. These pins have internal pullup/pulldown resistors; therefore, they can be left floating when the default value is desired. However, external pulldowns of 500 Ohms or less must be used between the pins and ground to effectively override default (internal) pullups, while external pullups of 2.2K Ohms or less should be used to override default pulldowns on BF[1:0].

Each Pentium processor is specified to operate within a single bus-to-core ratio and a specific minimum to maximum bus frequency range (corresponding to a minimum to maximum core frequency range). Operation in other bus-to-core ratios or outside the specified operating frequency range is not supported. For example, the 150 MHz Pentium processor does not operate beyond the 60 MHz bus frequency and only supports the 2/5 bus-to-core ratio; it does not support the 1/3, 1/2, or 2/3 bus-to-core ratios. The table below clarifies and summarizes these specifications.

Bus to Core Frequency Ratios for the Pentium® Processor

BF1	BF0	Pentium® Processor (75/90/100/120/ 133/150/166/200) Bus/Core Ratio	Pentium Processor with MMX™ Technology Bus/Core Ratio[4]	Max Bus/Core Frequency (MHz)	Min Bus/Core Frequency (MHz)
0	1	1/3	1/3	66/200	33/100
0	0	2/5	2/5	66/166	33/83
0	0	2/5	2/5	60/150	30/75
1	0	1/2	1/2 [2]	66/133	33/66
1	0	1/2	1/2 [2]	60/120	30/60
1	0	1/2	1/2 [2]	50/100[3]	25/50
1	1	2/3 [1]	reserved	66/100[3]	33/50
1	1	2/3 [1]	reserved	60/90	30/45
1	1	2/3 [1]	reserved	50/75	25/37.5

NOTES:

1. This is the default bus fraction for the Pentium® processor (75/90/100/120/133/150/166/200). If the BF pins are left floating, the processor will be configured for the 2/3 bus to core frequency ratio.
2. This is the default bus fraction for the Pentium processor with MMX™ technology. If the BF pins are left floating, the processor will be configured for the 1/2 bus to core frequency ratio.
3. The 100 MHz (Max Core Frequency) Pentium processors can be operated in both 1/2 and 2/3 Bus/Core Ratios.
4. Currently, the desktop Pentium processor with MMX technology supports 66/200 and 66/166 operation.

If BF[1:0] are left unconnected on the Pentium processor with MMX technology, the bus-to-core ratio defaults to 2/5. If BF[1:0] are left unconnected on the Pentium processor (75/90/100/120/133/150/166/200) the bus-to-core ratio defaults to 2/3.

When Sampled/Driven

BF[0:1] are sampled at RESET and cannot be changed until another non-warm (1 ms) assertion of RESET. BF[1:0] must meet a 1 ms setup time to the falling edge of RESET.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
RESET	BF[1:0] are sampled at the falling edge of RESET.

5.1.11. BOFF#

BOFF#	Backoff
	The back off input is used to force the Pentium® processor off the bus in the next clock.
	Synchronous Input

Signal Description

In response to BOFF#, the Pentium processor will abort all outstanding bus cycles that have not yet completed and float the Pentium processor bus in the next clock. The processor floats all pins normally floated during bus hold. Note that since the bus is floated in the clock after BOFF# is asserted, an acknowledge is not necessary (HLDA is not asserted in response to BOFF#).

The processor remains in bus hold until BOFF# is negated, at which time the Pentium processor restarts any aborted bus cycle(s) in their entirety by driving out the address and status and asserting ADS#.

This pin can be used to resolve a deadlock situation between two bus masters.

Any data with BRDY# returned to the processor while BOFF# is asserted is ignored.

BOFF# has higher priority than BRDY#. If both BOFF# and BRDY# occur in the same clock, BOFF# takes effect.

BOFF# also has precedence over BUSCHK#. If BOFF# and BUSCHK# are both asserted during a bus cycle, BOFF# causes the BUSCHK# to be forgotten.

When Sampled

BOFF# is sampled on every rising clock edge, including when RESET and INIT are asserted.

NOTE

If a read cycle is running on the bus, and an internal snoop of that read cycle hits a modified line in the data cache, and the system asserts BOFF#, then the sequence of bus cycles is as follows. Upon negation of BOFF#, the Pentium processor will drive out a writeback resulting from the internal snoop hit. After completion of the writeback, the processor will then restart the original read cycle. Thus, like external snoop writebacks, internal snoop writebacks may also be reordered in front of cycles that encounter a BOFF#. Also note that, although the original read encountered both an external BOFF# and an internal snoop hit to an M-state line, it is restarted only once.

This circumstance can occur during accesses to the page tables/directories and during prefetch cycles (these accesses cause a bus cycle to be generated before the internal snoop to the data cache is performed).

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A3-A31 ADS# AP BE7#-BE3# CACHE# D/C# D63-D0 DP7-DP0 LOCK# M/IO# PCD PWT SCYC W/R#	These signals float in response to BOFF#.
BRDY#	If BRDY# and BOFF# are asserted simultaneously, BOFF# takes priority and BRDY# is ignored.
EADS#	EADS# is recognized when BOFF# is asserted.
HLDA	The same pins are floated when HLDA or BOFF# is asserted.
BUSCHK#	If BUSCHK# and BOFF# are both asserted during a bus cycle, BOFF# takes priority and BUSCHK# is forgotten.
NA#	If NA# and BOFF# are asserted simultaneously, BOFF# takes priority and NA# is ignored.

5.1.12. BP3-BP0

BP3-BP0	Breakpoint signals
	BP3-BP0 externally indicate a breakpoint match.
	Synchronous Output

Signal Description

The Breakpoint pins (BP3-0) correspond to the debug registers, DR3-DR0. These pins externally indicate a breakpoint match when the debug registers are programmed to test for breakpoint matches. BP1 and BP0 are multiplexed with the performance monitoring pins (PM1 and PM0). The PB1 and PB0 bits in the debug mode control register determine if the pins are configured as breakpoint or performance monitoring pins. The pins come out of RESET configured for performance monitoring.

Because of the fractional-speed bus, each assertion of a Pentium processor BP pin indicates that one or more BP matches occurred. The maximum number of matches per assertion is two when using the 2/3 or 1/2 bus-to-core ratios. Similarly, the maximum number of matches per assertion is three when using the 2/5 or 1/3 bus-to-core ratios.

When Sampled/Driven

The BP3-BP0 pins are driven in every clock and are not floated during bus HOLD of BOFF#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
PM1-PM0	BP1 and BP0 share pins with PM1 and PM0, respectively.

5.1.13. BRDY#

BRDY#	Burst Ready
	Transfer complete indication.
	Synchronous Input

Signal Description

The Burst Ready input indicates that the external system has presented valid data on the data pins in response to a read, or that the external system has accepted the Pentium processor data in response to a write request.

Each cycle generated by the Pentium processor will either be a single transfer read or write, or a burst cache line fill or writeback. For single data transfer cycles, one BRDY# is expected to be returned to the Pentium processor. Once this BRDY# is returned, the cycle is complete. For burst transfers, four data transfers are expected by the Pentium processor. The cycle is ended when the fourth BRDY# is returned.

When Sampled

This signal is sampled in the T2, T12 and T2P bus states.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BOFF#	If BOFF# and BRDY# are asserted simultaneously, BOFF# takes priority and BRDY# is ignored.
BUSCHK#	BUSCHK# is sampled with BRDY#.
CACHE#	In conjunction with the KEN# input, CACHE# determines whether the bus cycle will consist of 1 or 4 transfers.
D63-D0	During reads, the D63-D0 pins are sampled by the Pentium® processor with BRDY#. During writes, BRDY# indicates that the system has accepted D63-D0.
DP7-0	During reads, the DP7-0 pins are sampled by the Pentium processor with BRDY#. During writes, BRDY# indicates that the system has accepted DP7-0.
EWBE#	EWBE# is sampled with each BRDY# of a write cycle.
KEN#	KEN# is sampled and latched by the Pentium processor with the earlier of the first BRDY# or NA#. Also, in conjunction with the CACHE# input, KEN# determines whether the bus cycle will consist of 1 or 4 transfers (assertions of BRDY#).
LOCK#	LOCK# is deasserted after the last BRDY# of the locked sequence.
PCHK#	PCHK# indicates the results of the parity check two clocks after BRDY# is returned for reads.
PEN#	PEN# is sampled with BRDY# for read cycles.
WB/WT#	WB/WT# is sampled and latched by the Pentium processor with the earlier of the first BRDY# or NA#.

5.1.14. BRDYC#

BRDYC#	Burst Ready
	Transfer complete indication.
	Synchronous Input

Signal Description

This signal is identical to the BRDY# input. This signal can be used to relieve tight board timings by easing the load on the Burst Ready signal.

In addition to its normal functionality, BRDYC# is sampled with BUSCHK# at RESET to select the buffer strength for some pins. BRDYC# has an internal pullup resistor. To override the default settings for the buffer strengths, this pin should be driven and not permanently strapped to ground since this will interfere with the normal operation of this pin. The driver should have an internal resistance of 1K Ohms or less. This is only a function of BRDYC#. The BRDY# signal is not sampled to select buffer sizes.

When Sampled/Driven

Refer to the BRDY# signal description.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BRDY#	BRDYC# is identical to the BRDY# input.
RESET	BRDYC# and BUSCHK# are sampled at RESET to select the buffer strength for some pins

5.1.15. BREQ

BREQ	Bus Request
	Indicates externally when a bus cycle is pending internally.
	Output

Signal Description

The Pentium processor asserts the BREQ output whenever a bus cycle is pending internally. BREQ is always asserted in the first clock of a bus cycle with ADS#. Furthermore, if the Pentium processor is not currently driving the bus (due to AHOLD, HOLD, or BOFF#), BREQ is asserted in the same clock that ADS# would have been asserted if the Pentium processor were driving the bus. After the first clock of the bus cycle, BREQ may change state. Every assertion of BREQ is not guaranteed to have a corresponding assertion of ADS#.

External logic can use the BREQ signal to arbitrate between multiple processors. This signal is always driven regardless of the state of AHOLD, HOLD or BOFF#.

When Driven

BREQ is always driven by the Pentium processor, and is not floated during bus HOLD or BOFF#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	BREQ is always asserted in the clock that ADS# is asserted.

5.1.16. BUSCHK#

BUSCHK#	Bus Check
	Allows the system to signal an unsuccessful completion of a bus cycle.
	Synchronous Input

Signal Description

The Bus Check input pin allows the system to signal an unsuccessful completion of a bus cycle. If this pin is sampled active, the Pentium processor will latch the address and control signals of the failing cycle in the machine check registers. If in addition, the MCE bit in CR4 is set, the Pentium processor will vector to the machine check exception upon completion of the current instruction.

If BUSCHK# is asserted in the middle of a cycle, the system must return all expected BRDY#s to the Pentium processor. BUSCHK# is remembered by the processor if asserted during a bus cycle. The processor decides after the last BRDY# whether to take the machine check exception or not.

BOFF# has precedence over BUSCHK#. If BOFF# and BUSCHK# are both asserted during a bus cycle, the BOFF# causes the BUSCHK# to be forgotten.

In addition to its normal functionality, BUSCHK# is sampled with BRDYC# at RESET to select the buffer strength for some pins. BUSCHK# has an internal pullup resistor. To override the default settings for the buffer strengths, this pin should be driven and not permanently strapped to ground since this will interfere with the normal operation of this pin. The driver should have an internal resistance of 1K Ohms or less.

When Sampled

BUSCHK# is sampled when BRDY# is returned to the Pentium processor.

NOTE

The Pentium processor can remember only one machine check exception at a time. This exception is recognized on an instruction boundary. If BUSCHK# is sampled active while servicing the machine check exception for a previous BUSCHK#, it will be remembered by the processor until the original machine check exception is completed. It is then that the processor will service the machine check exception for the second BUSCHK#. Note that only one BUSCHK# will be remembered by the processor while the machine exception for the previous one is being serviced.

When the BUSCHK# is sampled active by the processor, the cycle address and cycle type information for the failing bus cycle is latched upon assertion of the last BRDY# of the bus cycle. The information is latched into the Machine Check Address (MCA) and Machine Check Type (MCT) registers respectively. However, if the BUSCHK# input is not deasserted before the

first BRDY# of the next bus cycle, and the machine check exception for the first bus cycle has not occurred, then new information will be latched into the MCA and MCT registers, over-writing the previous information at the completion of this new bus cycle. Therefore, in order for the MCA and MCT registers to report the correct information for the failing bus cycle when the machine check exception for this cycle is taken at the next instruction boundary, the system must deassert the BUSCHK# input immediately after the completion of the failing bus cycle (i.e., before the first BRDY# of the next bus cycle is returned).

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BOFF#	If BOFF# and BUSCHK# are both asserted during a bus cycle, the BOFF# signal causes the BUSCHK# to be forgotten.
BRDY#	BUSCHK# is sampled with BRDY#.
BRDYC#	BUSCHK# is sampled with BRDYC# at RESET to select the buffer strength for some pins.
RESET	BUSCHK# and BRDYC# are sampled at RESET to select the buffer strength for some pins

5.1.17. CACHE#

CACHE#	Cacheability
	External indication of internal cacheability.
	Synchronous Input/Output

Signal Description

The Cacheability output is a cycle definition pin. For Pentium processor initiated cycles this pin indicates internal cacheability of the cycle (if a read), and indicates a burst writeback (if a write). CACHE# is asserted for cycles coming from the cache (writebacks) and for cycles that will go into the cache if KEN# is asserted (linefills). More specifically, CACHE# is asserted for cacheable reads, cacheable code fetches, and writebacks. It is driven inactive for non-cacheable reads, TLB replacements, locked cycles (except writeback cycles from an external snoop that interrupt a locked read/modify/write sequence), I/O cycles, special cycles and writethroughs.

For read cycles, the CACHE# pin indicates whether the Pentium processor will allow the cycle to be cached. If CACHE# is asserted for a read cycle, the cycle will be turned into a cache line fill if KEN# is returned active to the Pentium processor. If this pin is driven inactive during a read cycle, Pentium processor will not cache the returned data, regardless of the state of the KEN#.

If this pin is asserted for a write cycle, it indicates that the cycle is a burst writeback cycle. Writethroughs cause a non-burst write cycle to be driven to the bus. The Pentium processor does not support write allocations (cache line fills as a result of a write miss).

When operating in dual processing mode, the Pentium processors uses this signal for private snooping.

When Sampled/Driven

CACHE# is driven to its valid level in the same clock as the assertion of ADS# and remains valid until the earlier of the last BRDY# or the clock after NA#.

This signal becomes an Input/Output when two Pentium processor are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	CACHE# is driven to its valid level with ADS#.
BOFF#	CACHE# floats one clock after BOFF# is asserted.
BRDY#	In conjunction with the KEN# input, CACHE# determines whether the bus cycle will consist of 1 or 4 transfers (assertions of BRDY#).
HLDA	CACHE# floats when HLDA is asserted.
KEN#	KEN# and CACHE# are used together to determine if a read will be turned into a linefill.

5.1.18. CLK

CLK	Clock
	Fundamental timing for the Pentium processor.
	Input

Signal Description

The Clock input provides the fundamental timing for the Pentium processor. Its frequency is proportional to the internal operating frequency of the Pentium processor (as selected by the BF[1:0] pins) and requires TTL levels. All external timing parameters except TDI, TDO, TMS, and TRST# are specified with respect to the rising edge of CLK.

Note that the CLK signal on the Pentium processor with MMX technology is 3.3V tolerant, while on the Pentium processor (75/90/100/120/133/150/166/200) the CLK input is 5.0V tolerant.

When Sampled

CLK is used as a reference for sampling other signals. It is recommended that CLK begin toggling within 150 ms after V_{CC} reaches its proper operating level. This recommendation is only to ensure long term reliability of the device. V_{CC} specifications and clock duty cycle, stability and frequency specifications must be met for 1 millisecond before the negation of RESET. If at any time during normal operation one of these specifications is violated, the power on RESET sequence must be repeated. This requirement is to insure proper operation of the phase locked loop circuitry on the clock input.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
All except TCK, TDI, TDO, TMS, TRST#	External timing parameters are measured from the rising edge of CLK for all signals except TDI, TDO, TMS, TCK, and TRST#.

5.1.19. CPUTYP

CPUTYP	CPU Type Definition Pin
	Used to configure the Pentium® processor as a Dual processor.
	Asynchronous Input

Signal Description

The CPUTYP pin is used to determine whether the Pentium processor will function as a Primary or Dual processor. CPUTYP must be strapped to either V_{CC} or V_{SS}. When CPUTYP is strapped to V_{CC}, the Pentium processor will function as a Dual processor. When CPUTYP is strapped to V_{SS} (or left unconnected), the Pentium processor will function as a Primary processor. In a single socket system design, CPUTYP pin must be strapped to V_{SS} (or left unconnected).

When Sampled/Driven

CPUTYP is sampled at RESET and cannot be changed until another non-warm (1 ms) assertion of RESET. CPUTYP must meet a 1 ms setup time to the falling edge of RESET. It is recommended that CPUTYP be strapped to V_{CC} or V_{SS}.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A20M#	When CPUTYP is strapped to V_{CC}, the processor will ignore the A20M# input.
BE4#-BE0#	The BE3#-BE0# input values are sampled during RESET to determine the APIC ID. The Dual processor uses BE4# to indicate to the Primary processor that it has completed it's cache flush operation. Refer to the BE4#-BE0# pin description.
D/P#	D/P# is driven by the Pentium processor only when the CPUTYP signal is strapped to V_{SS}.
DPEN#	When CPUTYP is strapped to V_{CC}, DPEN# is driven active to indicate that the second socket is occupied.
FERR#	When CPUTYP is strapped to V_{CC}, the FERR# output is undefined.
FLUSH#	When operating in dual processing mode, the FLUSH# inputs become Synchronous to the CPU clock.
IGNNE#	When CPUTYP is strapped to V_{CC}, the processor will ignore the IGNNE# input.
RESET	CPUTYP is sampled at the falling edge of RESET. When operating in dual processing mode, the RESET inputs become synchronous to the CPU clock.

NOTE

It is common practice to put either a pullup or pulldown resistor on a net. If a pullup resistor is connected to the CPUTYP pin in order to operate in a Dual Processing mode, the value of this resistor must be 100 ohms or less to override the internal pulldown. In the absence of an external pullup, the internal pulldown will sufficiently pulldown the CPUTYP pin, therefore the pin can be left floating.

5.1.20. D/C#

D/C#	Data/Code
	Distinguishes a data access from a code access.
	Synchronous Input/Output

Signal Description

The Data/Code signal is one of the primary bus cycle definition pins. D/C# distinguishes between data (D/C# = 1) and code/special cycles (D/C# = 0).

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

The D/C# pin is driven valid in the same clock as ADS# and the cycle address. It remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#.

This signal becomes an Input/Output when two Pentium processors are operating together in Dual Processing Mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	D/C# is driven with ADS#.
BOFF#	D/C# floats one clock after BOFF# is asserted.
HLDA	D/C# floats when HLDA is asserted.

5.1.21. D63-D0

D63-D0	Data Lines
	Forms the 64-bit data bus.
	Input/Output

Signal Description

The bi-directional lines, D63-D0 form the 64 data bus lines for the Pentium processor. Lines D7-D0 define the least significant byte of the data bus; lines D63-D56 define the most significant byte of the data bus.

When Sampled/Driven

When the CPU is driving the data lines (during writes), they are driven during the T2, T12, or T2P clocks for that cycle.

During reads, the CPU samples the data bus when BRDY# is returned.

D63-D0 are floated during T1, TD, and Ti states.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BE7#-BE0#	BE7#-BE0# indicate which data bytes are being requested or driven by the Pentium® processor.
BOFF#	D63-D0 float one clock after BOFF# is asserted.
BRDY#	BRDY# indicates that the data bus transfer is complete.
DP7-DP0	Even data parity is driven/sampled with the data bus on DP7-DP0.
HLDA	D63-D0 float when HLDA is asserted.
PCHK#	The status of the data bus parity check is driven on PCHK#.
PEN#	Even data parity with D63-D0 should be returned on to the Pentium processor on the DP pin. If a data parity error occurs, and PEN# is enabled, the cycle will be latched and a machine check exception will be taken if CR4.MCE = 1.

5.1.22. D/P#

D/P#	Dual Processor / Primary Processor
	Indicates whether the Dual processor or the Primary processor is driving the bus.
	Synchronous Output

Signal Description

The D/P# pin is driven LOW when the Primary processor is driving the bus. Otherwise, the Primary processor drives this pin high to indicate that the Dual processor owns the bus. The D/P# pin can be sampled for the current cycle with ADS#. This pin is defined only on the Primary processor. In a single socket system design, D/P# pin should be left NC.

When Sampled/Driven

The D/P# pin is always driven by the Primary processor and should be sampled with ADS# of the current cycle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	D/P# is valid for the current cycle with ADS# (like a status pin).
CPUTYP	D/P# is driven by the Pentium® processor when the CPUTYP signal is strapped to V_{SS} (or left unconnected).
SMIACT#	When operating in dual processing mode, D/P# qualifies the SMIACT# SMM indicator.

5.1.23. DP7-DP0

DP7-DP0	Data Parity
	Bi-directional data parity pins for the data bus.
	Input/Output

Signal Description

These are the bi-directional Data Parity pins for the processor. There is one parity pin for each byte of the data bus. DP7 applies to D63-D56, DP0 applies to D7-D0.

As outputs, the data parity pins are driven by the Pentium processor with even parity information for writes in the same clock as write data. Even parity means that there are an even number of HIGH logic values on the eight corresponding data bus pins and the parity pin.

As inputs, even parity information must be driven back to the Pentium processor on these pins in the same clock as the data to ensure that the correct parity check status is indicated by the Pentium processor.

The value read on the data parity pins does not affect program execution unless PEN# is also asserted. If PEN# is not asserted, the value returned on the DP pins is used only to determine even parity and drive the PCHK# output with the proper value. If PEN# is asserted when a parity error occurs, the cycle address and type will be latched in the MCA and MCT registers. If in addition, the MCE bit in CR4 is set, a machine check exception will be taken.

It is the responsibility of the system to take appropriate actions if a parity error occurs. If parity checks are not implemented in the system, the DP[7:0] and PEN# pins should be tied to V_{CC} through a pullup resistor and the PCHK# pin may be ignored.

When Sampled/Driven

As outputs, the data parity pins are driven by the Pentium processor with even parity information in the same clock as write data. The parity remains valid until sampled by the assertion of BRDY# by the system.

As inputs, even parity information must be driven back to the Pentium processor on these pins in the same clock as the data to ensure that the correct parity check status is indicated by the Pentium processor. The data parity pins must be at a valid logic level and meet setup and hold times when sampled.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BE7#-BE0#	Even data parity is checked/driven only on the data bytes enabled by BE7#-BE0#.
BOFF#	DP7-DP0 are floated one clock after BOFF# is asserted.
BRDY#	DP7-DP0 are sampled with BRDY# for reads.
D63-D0	The DP7-0 pins are used to create even parity with D63-D0 on a byte by byte basis. DP7-DP0 are driven with D63-D0 for writes.
HLDA	DP7-DP0 are floated when HLDA is asserted.
PCHK#	The status of the data parity check is driven on the PCHK# output.
PEN#	The DP7-DP0 pins are used to create even parity with D63-D0. If even parity is not detected, and PEN# is enabled, the cycle address and type will be latched. If in addition CR4.MCE = 1, the machine check exception will be taken.

5.1.24. DPEN#

DPEN#	Second Socket Occupied
	Configuration signal which indicates that the second socket in a dual socket system is occupied.
	Synchronous Input (to the Pentium® processor) Synchronous Output (from the Pentium processor, when configured as a Dual processor)

Signal Description

DPEN# is driven during RESET by the Pentium processor when configured as a Dual processor to indicate to the Primary processor in the first socket that there is a Dual processor present in the system.

This pin has an internal pullup resistor and is sampled at the falling edge of RESET. When using an active circuit to override the internal pullup resistor, the driver should have an internal effective pulldown resistance of 1K Ohms or less.

When Sampled/Driven

DPEN# is driven during RESET by the Dual processor, and sampled at the falling edge of RESET by the Primary processor. This pin becomes PICD0 following the falling edge of RESET. This pin should be valid and stable two clocks before and after the falling edge of RESET.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
CPUTYP	When CPUTYP is strapped to V_{CC}, DPEN# is driven active to indicate that the second socket is occupied.
RESET	DPEN# is valid during the falling edge of RESET.
PICD0	DPEN# shares a pin with PICD0.

5.1.25. EADS#

EADS#	External Address Strobe
	Signals the Pentium® processor to run an inquire cycle with the address on the bus.
	Synchronous Input

Signal Description

The EADS# input indicates that a valid external address has been driven onto the Pentium processor address pins to be used for an inquire cycle. The address driven to the Pentium processor when EADS# is sampled asserted will be checked with the current cache contents. The HIT# and HITM# signals will be driven to indicate the result of the comparison. If the INV pin is returned active (high) to the Pentium processor in the same clock as EADS# is sampled asserted, an inquire hit will result in that line being invalidated. If the INV pin is returned inactive (low), an inquire hit will result in that line being marked Shared (S).

When Sampled

To guarantee recognition, EADS# should be asserted two clocks after an assertion of AHOLD or BOFF#, or one clock after an assertion of HLDA. In addition, the Pentium processor will ignore an assertion of EADS# if the processor is driving the address bus, or if HITM# is active, or in the clock after ADS# or EADS# is asserted.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A31-A5	The inquire cycle address must be valid on A31-A5 when EADS# is sampled asserted.
A4-A3	These signals must be at a valid logic level when EADS# is sampled asserted.
AHOLD	EADS# is recognized while AHOLD is asserted.
AP	AP is sampled when EADS# is sampled asserted.
APCHK#	APCHK# is driven to its valid level two clocks after EADS# is sampled asserted.
BOFF#	EADS# is recognized while BOFF# is asserted.
HIT#	HIT# is driven to its valid level two clocks after EADS# is sampled asserted.
HITM#	HITM# is driven to its valid level two clocks after EADS# is sampled asserted.
HLDA	EADS# is recognized while HLDA is asserted.
INV	INV is sampled with EADS# to determine the final state of the cache line in the case of an inquire hit.

5.1.26. EWBE#

EWBE#	External Write Buffer Empty
	Provides the option of strong write ordering to the memory system.
	Synchronous Input

Signal Description

The External write Buffer Empty input, when inactive (high), indicates that a writethrough cycle is pending in the external system. When the Pentium processor generates a write (memory or I/O), and EWBE# is sampled inactive, the Pentium processor will hold off all subsequent writes to all E or M-state lines until all writethrough cycles have completed, as indicated by EWBE# being active. In addition, if the Pentium processor has a write pending in a write buffer, the Pentium processor will also hold off all subsequent writes to E- or M-state lines. This insures that writes are visible from outside the Pentium processor in the same order as they were generated by software.

When the Pentium processor serializes instruction execution through the use of a serializing instruction, it waits for the EWBE# pin to go active before fetching and executing the next instruction.

After the OUT or OUTS instructions are executed, the Pentium processor ensures that EWBE# has been sampled active before beginning to execute the next instruction. Note that the instruction may be prefetched if EWBE# is not active, but it will not execute until EWBE# is sampled active.

When Sampled

EWBE# is sampled with each BRDY# of a write cycle. If sampled deasserted, the Pentium processor repeatedly samples EWBE# in each clock until it is asserted. Once sampled asserted, the Pentium processor ignores EWBE# until the next BRDY# of a write cycle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BRDY#	EWBE# is sampled with each BRDY# of a write cycle.
SMIACT#	SMIACT# is not asserted until EWBE# is asserted.

5.1.27. FERR#

FERR#	Floating-Point Error
	The floating-point error output is driven active when an unmasked floating-point error occurs.
	Synchronous Output

Signal Description

The Floating-Point Error output is driven active when an unmasked floating-point error occurs. FERR# is similar to the ERROR# pin on the Intel387 math coprocessor. FERR# is included for compatibility with systems using DOS type floating-point error reporting.

In some cases, FERR# is asserted when the next floating-point instruction is encountered and in other cases it is asserted before the next floating-point instruction is encountered depending upon the execution state of the instruction causing the exception.

The following class of floating-point exceptions drive FERR# at the time the exception occurs (i.e., before encountering the next floating-point instruction):

1. Stack fault, all invalid operation exceptions and denormal exceptions on: all transcendental instructions, FSCALE, FXTRACT, FPREM, FPREM(1), FBLD, FLD_extended, FRNDINT, and stack fault and invalid operation exceptions on Floating-Point arithmetic instructions with an integer operand (FIADD/FIMUL/FISUB/FIDIV, etc.).
2. All real stores (FST/FSTP), Floating-Point integer stores (FIST/FISTP) and BCD store (FBSTP) (true for all exception on stores except Precision Exception).

The following class of floating-point exceptions drive FERR# only after encountering the next floating-point instruction. Note that the Pentium processor with MMX technology will report a pending floating-point exception (assert FERR#) upon encountering the next floating-point or MMX instruction.

1. Numeric underflow, overflow and precision exception on: Transcendental instructions, FSCALE, FXTRACT, FPREM, FPREM(1), FRNDINT, and Precision Exception on all types of stores to memory.
2. All exceptions on basic arithmetic instructions (FADD/FSUB/FMUL/FDIV/ FSQRT/FCOM/FUCOM...)

FERR# is deasserted when the FCLEX, FINIT, FSTENV, or FSAVE instructions are executed. In the event of a pending unmasked floating-point exception the FNINIT, FNCLEX, FNSTENV, FNSAVE, FNSTSW, FNSTCW, FNENI, FNDISI, and FNSETPM instructions assert the FERR# pin. Shortly after the assertion of the pin, an interrupt window is opened during which the processor samples and services interrupts, if any. If no interrupts are sampled within this window, the processor will then execute these instructions with the pending unmasked exception. However, for the FNCLEX, FNINIT, FNSTENV, and FNSAVE instructions, the FERR# pin is deasserted to enable the execution of these instructions. For

details please refer to the *Intel Architecture Software Developer's Manual*, Volume 1 (Chapter 7 and Appendix D)..

This signal is undefined when the Pentium processor is configured as a Dual processor.

When Sampled/Driven

FERR# is driven in every clock and is not floated during bus HOLD or BOFF#. The FERR# signal is glitch free.

The Pentium processor, when configured as a Dual processor, will not drive this signal to valid levels.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
CPUTYP	When CPUTYP is strapped to V_{CC}, the FERR# output is undefined.

5.1.28. FLUSH#

FLUSH#	Cache Flush
	Writes all modified lines in the data cache back and flushes the code and data caches.
	Asynchronous Input (Normal, Uni-processor mode) Synchronous Input (Dual processor mode)

Signal Description

When asserted, the Cache Flush input forces the Pentium processor to writeback all modified lines in the data cache and invalidate both internal caches. A Flush Acknowledge special cycle will be generated by the Pentium processor indicating completion of the invalidation and writeback.

FLUSH# is implemented in the Pentium processor as an interrupt, so it is recognized on instruction boundaries. External interrupts are ignored while FLUSH# is being serviced. Once FLUSH# is sampled active, it is ignored until the flush acknowledge special cycle is driven.

If FLUSH# is sampled low when RESET transitions from high to low, tristate test mode is entered.

The Pentium processor, when operating with a second Pentium processor in dual processing mode, incorporates a mechanism to present an atomic cache flush operation to the system. The Dual processor performs the cache flush operation first, then grants the bus to the Primary processor. The Primary processor flushes its internal caches, and then runs the cache flush special cycle. This could cause the total flush latency of two Pentium processor in dual processor mode to be up to twice that of the Pentium processor in uni-processor mode.

The flush latency of the Pentium processor with MMX technology and the future Pentium OverDrive processor may also be up to twice that of the Pentium processor (75/90/100/120/133/150/166/200) due to the implementation of larger on-chip caches.

When Sampled/Driven

FLUSH# is sampled on every rising clock edge. FLUSH# is falling edge sensitive and recognized on instruction boundaries. Recognition of FLUSH# is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. If it meets setup and hold times, FLUSH# need only be asserted for one clock. To guarantee recognition if FLUSH# is asserted asynchronously, it must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor and remain asserted for a minimum pulse width of two clocks.

If the processor is in the HALT or Shutdown state, FLUSH# is still recognized. The processor will return to the HALT or Shutdown state after servicing the FLUSH#.

If FLUSH# is sampled low when RESET transitions from high to low, tristate test mode is entered. If RESET is negated synchronously, FLUSH# must be at its valid level and meet setup and hold times on the clock before the falling edge of RESET. If RESET is negated

asynchronously, FLUSH# must be at its valid level two clocks before and after RESET transitions from high to low.

When operating in a dual processing system, FLUSH# must be sampled synchronously to the rising CLK edge to ensure both processors recognize an active FLUSH# signal in the same clock.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS# and cycle definition pins.	Writeback cycles are driven as a result of FLUSH# assertion. The Flush Special Cycle is driven as a result of FLUSH# assertion.
RESET	If FLUSH# is sampled low when RESET transitions from high to low, tristate test mode is entered.
CPUTYP	When operating in dual processing mode, the FLUSH# inputs become synchronous to the CPU clock.

5.1.29. FRCMC#

FRCMC#	Functional Redundancy Checking Master/Checker Configuration
	Determines whether the Pentium® processor is configured as a Master or Checker.
	Asynchronous Input

NOTE: Functional Redundancy Checking is not supported on the Pentium processor with MMX technology. The FRCMC# pin is defined only for the Pentium processor (75/90/100/120/133/150/166/200). This pin should be left as a "NC" or tied to $V_{CC}3$ via an external pullup resistor on the Pentium processor with MMX technology.

Signal Description

The Functional Redundancy Checking Master/Checker Configuration input is sampled in every clock that RESET is asserted to determine whether the Pentium processor is configured in master mode (FRCMC# high) or checker mode (FRCMC# low). When configured as a master, the Pentium processor drives its output pins as required by the bus protocol. When configured as a checker, the Pentium processor tristates all outputs (except IERR# and TDO) and samples the output pins that would normally be driven in master mode. If the sampled value differs from the value computed internally, the Checker Pentium processor asserts IERR# to indicate an error.

Note that the final configuration as a master or checker is set after RESET and may not be changed other than by a subsequent RESET. FRCMC# is sampled in every clock that RESET is asserted to prevent bus contention before the final mode of the processor is determined.

When Sampled

This pin is sampled in any clock in which RESET is asserted. FRCMC# is sampled in the clock before RESET transitions from high to low to determine the final mode of the processor. If RESET is negated synchronously, FRCMC# must be at its valid level and meet setup and hold times on the clock before the falling edge of RESET. If RESET is negated asynchronously, FRCMC# must be at its valid level two clocks before and after RESET transitions from high to low.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
IERR#	IERR# is asserted by the Checker Pentium® processor in the event of an FRC error.
RESET	FRCMC# is sampled when RESET is asserted to determine if the Pentium processor is in Master or Checker mode.

5.1.30. HIT#

HIT#	Inquire Cycle Hit/Miss
	Externally indicates whether an inquire cycle resulted in a hit or miss.
	Synchronous Input/Output

Signal Description

The HIT# output is driven to reflect the outcome of an inquire cycle. If an inquire cycle hits a valid line (M, E, or S) in either the Pentium processor data or instruction cache, HIT# is asserted two clocks after EADS# has been sampled asserted by the processor. If the inquire cycle misses the Pentium processor cache, HIT# is negated two clocks after EADS# is sampled asserted. This pin changes its value only as a result of an inquire cycle and retains its value between cycles.

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

HIT# reflects the hit or miss outcome of the inquire cycle 2 clocks after EADS# is sampled asserted. After RESET, this pin is driven high. It changes it value only as a result of an inquire cycle. This pin is always driven. It is not floated during bus HOLD or BOFF#.

This signal becomes an Input/Output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relative to Other Signals
A31-A5	HIT# is driven to indicate whether the inquire address driven on A31-A5 is valid in an internal cache.
EADS#	HIT# is driven two clocks after EADS# is sampled asserted to indicate the outcome of the inquire cycle.
HITM#	HITM# is never asserted without HIT# also being asserted.

5.1.31. HITM#

HITM#	Inquire Cycle Hit/Miss to a Modified Line
	Externally indicates whether an inquire cycle hit a modified line in the data cache.
	Synchronous Input/Output

Signal Description

The HITM# output is driven to reflect the outcome of an inquire cycle. If an inquire cycle hits a modified line in the Pentium processor data cache, HITM# is asserted two clocks after EADS# has been sampled asserted by the processor and a writeback cycle is scheduled to be driven to the bus. If the inquire cycle misses the Pentium processor cache, HITM# is negated two clocks after EADS# is sampled asserted.

HITM# can be used to inhibit another bus master from accessing the data until the line is completely written back.

HITM# is asserted two clocks after an inquire cycle hits a modified line in the Pentium processor cache. ADS# for the writeback cycle will be asserted no earlier than two clocks after the assertion of HITM#. ADS# for the writeback cycle will be driven even if AHOLD for the inquire cycle is not yet deasserted. ADS# for a writeback of an external snoop cycle is the only ADS# that will be driven while AHOLD is asserted.

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

HITM# is driven two clocks after EADS# is sampled asserted to reflect the outcome of the inquire cycle. HITM# remains asserted until two clocks after the last BRDY# of writeback is returned. This pin is always driven. It is not floated during bus HOLD or BOFF#.

This signal becomes an input/output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A31-A5	HITM# is driven to indicate whether the inquire address driven on A31-A5 is in the modified state in the data cache.
EADS#	HITM# is driven two clocks after EADS# is sampled asserted.
HIT#	HITM# is never asserted without HIT# also being asserted.

5.1.32. HLDA

HLDA	Bus Hold Acknowledge
	External indication that the Pentium processor outputs are floated.
	Synchronous Input/Output

Signal Description

The Bus Hold Acknowledge output goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the Pentium processor has given the bus to another local bus master. Internal instruction execution will continue from the internal caches during bus HOLD/HLDA.

When leaving bus hold, HLDA will be driven inactive and the Pentium processor will resume driving the bus. A pending bus cycle will be driven in the same clock that HLDA is deasserted by the Pentium processor (75/90/100/120/133/150/166/200) and one clock after HLDA is deasserted by the Pentium processor with MMX technology.

The operation of HLDA is not affected by the assertion of BOFF#. If HOLD is asserted while BOFF# is asserted, HLDA will be asserted two clocks later. If HOLD goes inactive while BOFF# is asserted, HLDA is deasserted two clocks later.

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

When the Pentium processor bus is idle, HLDA is driven high two clocks after HOLD is asserted, otherwise, HLDA is driven high two clocks after the last BRDY# of the current cycle is returned. It is driven active in the same clock that the Pentium processor floats its bus. When leaving bus hold, HLDA will be driven inactive 2 clocks after HOLD is deasserted and the Pentium processor will resume driving the bus. The HLDA signal is glitch free.

This signal becomes an input/output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A3-A31 ADS# AP BE7#-BE3# CACHE# D/C# D63-D0 DP7-DP0 LOCK# M/IO# PCD PWT SCYC W/R#	These signals float in response to HLDA.
BOFF#	The same pins are floated when HLDA or BOFF# is asserted.
EADS#	EADS# is recognized while HLDA is asserted.
HOLD	The assertion of HOLD causes HLDA to be asserted when all outstanding cycles are complete.

5.1.33. HOLD

HOLD	Bus Hold
	The bus hold request input allows another bus master complete control of the Pentium® processor bus.
	Synchronous Input

Signal Description

The Bus Hold request input allows another bus master complete control of the Pentium processor bus. In response to HOLD, after completing all outstanding bus cycles the Pentium processor will float most of its output and input/output pins and assert HLDA. The Pentium processor will maintain its bus in this state until HOLD is deasserted. Cycles that are locked together will not be interrupted by bus HOLD. HOLD is recognized during RESET.

When Sampled

HOLD is sampled on every rising clock edge including during RESET and INIT.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A3-A31 ADS# AP BE7#-BE3# CACHE# D/C# D63-D0 DP7-DP0 LOCK# M/IO# PCD PWT SCYC W/R#	These are the signals floated in response to HOLD.
HLDA	HLDA is asserted when the Pentium® processor relinquishes the bus in response to the HOLD request.

5.1.34. IERR#

IERR#	Internal or Functional Redundancy Check[1] Error
	Alerts the system of internal parity errors and functional redundancy errors.
	Output

NOTE:

1. Functional Redundancy Checking is not supported on the Pentium processor with MMX technology

Signal Description

The Internal Error output is used to alert the system of two types of errors, internal parity errors and functional redundancy errors.

If a parity error occurs on a read from an internal array (reads during normal instruction execution, reads during a flush operation, reads during BIST and testability cycles, and reads during inquire cycles), the Pentium processor will assert the IERR# pin for one clock and then shutdown. Shutdown will occur provided the processor is not prevented from doing so by the error.

If the Pentium processor is configured as a checker (by FRCMC# being sampled low while RESET is asserted) and a mismatch occurs between the value sampled on the pins and the value computed internally, the Pentium processor will assert IERR# two clocks after the mismatched value is returned. Shutdown is not entered as a result of a function redundancy error.

It is the responsibility of the system to take appropriate action if an internal parity or FRC error occurs.

When Driven

IERR# is driven in every clock. While RESET is active IERR# is driven high. After RESET is deasserted, IERR# will not be asserted due to an FRC mismatch until after the first clock of the first bus cycle. Note however that IERR# may be asserted due to an internal parity error before the first bus cycle. IERR# is asserted for 1 clock for each detected FRC or internal parity error, two clocks after the error is detected. IERR# is asserted for each detected mismatch, so IERR# may be asserted for more than one consecutive clock.

IERR# is not floated with HOLD or BOFF#. IERR# is a glitch free signal.

NOTE

When paging is turned on, an additional parity check occurs to page 0 for all TLB misses. If this access is a valid entry in the cache and this entry also has a parity error, then IERR# will be asserted and shutdown will occur even though the pipeline is frozen to service the TLB miss.

During a TLB miss, a cache lookup occurs (to the data cache for a data TLB miss, or the code cache for a code TLB miss) to a default page 0 physical address until the correct page translation becomes available. At this time, if a

valid cache entry is found at the page 0 address, then parity will be checked on the data read out of the cache. However, the data is not used until after the correct page address becomes available. If this valid line contains a true parity error, then the error will be reported. This will not cause an unexpected parity error. It can cause a parity error and shutdown at a time when the data is not being used because the pipeline is frozen to service the TLB miss. However, it still remains that a true parity error must exist within the cache in order for IERR# assertion and shutdown to occur. For more details on TLB, refer to Section 3.7 of the *Intel Architecture Software Developer's Manual*, Volume 1.

Relation to Other Signals

Pin Symbol	Relative to Other Signals
FRCMC#	If the Pentium® processor is configured as a Checker, IERR# will be asserted in the event of an FRC error.

5.1.35. IGNNE#

IGNNE#	Ignore Numeric Exception
	Determines whether or not numeric exceptions should be ignored.
	Asynchronous Input

Signal Description

This is the Ignore Numeric Exception input. This pin has no effect when the NE bit in CR0 is set to 1. When the CR0.NE bit is 0, this pin functions as follows:

When the IGNNE# pin is asserted, the Pentium processor will ignore any pending unmasked numeric exception and continue executing floating-point instructions for the entire duration that this pin is asserted.

When IGNNE# is not asserted and a pending unmasked numeric exception exists, (SW.ES = 1), the Pentium processor will behave as follows:

On encountering a floating-point instruction that is one of FNINIT, FNCLEX, FNSTENV, FNSAVE, FNSTSW, FNSTCW, FNENI, FNDISI, or FNSETPM, the Pentium processor will assert the FERR# pin. Subsequently, the processor opens an interrupt sampling window. The interrupts are checked and serviced during this window. If no interrupts are sampled within this window, the processor will then execute these instructions in spite of the pending unmasked exception. For further details please refer to the *Intel Architecture s Software Developer's Manual*, Volume1 (Chapter 7 and Appendix D).

On encountering any floating-point instruction other than FINIT, FCLEX, FSTENV, FSAVE, FSTSW, FSTCW, FENI, FDISI, or FSETPM, the Pentium processor will stop execution and wait for an external interrupt.

The Pentium processor, when configured as a Dual processor, will ignore the IGNNE# input.

When Sampled/Driven

IGNNE# is sampled on every rising clock edge. Recognition of IGNNE# is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. To guarantee recognition if IGNNE# is asserted asynchronously, it must have been deasserted for a minimum of two clocks before being returned active to the Pentium processor and remain asserted for a minimum pulse width of two clocks.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
CPUTYP	When strapped to V_{CC}, the processor will ignore the IGNNE# input.

5.1.36. INIT

INIT	Initialization
	Forces the Pentium® processor to begin execution in a known state without flushing the caches or affecting the floating-point state.
	Asynchronous Input

Signal Description

The Initialization input forces the Pentium processor to begin execution in a known state. The processor state after INIT is the same as the state after RESET except that the internal caches, write buffers, model specific registers, and floating-point registers retain the values they had prior to INIT. The Pentium processor starts execution at physical address FFFFFFF0H.

INIT can be used to help performance for DOS extenders written for the 80286. INIT provides a method to switch from protected to real mode while maintaining the contents of the internal caches and floating-point state. INIT may not be used instead of RESET after power-up.

Once INIT is sampled active, the INIT sequence will begin on the next instruction boundary (unless a higher priority interrupt is requested before the next instruction boundary). The INIT sequence will continue to completion and then normal processor execution will resume, independent of the deassertion of INIT. ADS# will be asserted to drive bus cycles even if INIT is not deasserted.

If INIT is sampled high when RESET transitions from high to low the Pentium processor will perform built-in self test (BIST) prior to the start of program execution.

When Sampled

INIT is sampled on every rising clock edge. INIT is an edge sensitive interrupt. Recognition of INIT is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. To guarantee recognition if INIT is asserted asynchronously, it must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor and remain asserted for a minimum pulse width of two clocks. INIT must remain active for three clocks prior to the BRDY# of an I/O write cycle to guarantee that the Pentium processor recognizes and processes INIT right after an I/O write instruction.

If INIT is sampled high when RESET transitions from high to low the Pentium processor will perform built-in self test. If RESET is driven synchronously, INIT must be at its valid level the clock before the falling edge of RESET. If RESET is driven asynchronously, INIT must be at its valid level two clocks before and after RESET transitions from high to low.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
RESET	If INIT is sampled high when RESET transitions from high to low, BIST will be performed.

5.1.37. INTR

INTR	External Interrupt
	Indicates that an external interrupt has been generated.
	Asynchronous Input

Signal Description

The INTR input indicates that an external interrupt has been generated. The interrupt is maskable by the IF bit in the EFLAGS register. If the IF bit is set, the Pentium processor will vector to an interrupt handler after the current instruction execution is completed. Upon recognizing the interrupt request, the Pentium processor will generate two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the first interrupt acknowledge cycle is completed to assure that the interrupt is recognized.

When the local APIC is hardware disabled, this pin is the INTR input for the processor. It bypasses the local APIC in that case.

When the local APIC is hardware enabled, this pin becomes the programmable interrupt LINT0. It can be programmed in software in any of the interrupt modes. Since this pin is the INTR input when the APIC is disabled, it is logical to program the vector table entry for this pin as ExtINT (i.e. through local mode). In this mode, the interrupt signal is passed on to the processor through the local APIC. The processor generates the interrupt acknowledge, INTA, cycle in response to this interrupt and receives the vector on the processor data bus.

When Sampled/Driven

INTR is sampled on every rising clock edge. INTR is an asynchronous input, but recognition of INTR is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. To guarantee recognition if INTR is asserted asynchronously it must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor.

NOTE

This applies only when using the APIC in the through local (virtual wire) mode. Once INTR has been asserted (by a rising edge), it must not be asserted again until after the end of the first resulting interrupt acknowledge cycle. Otherwise, the new interrupt may not be recognized. The end of an interrupt acknowledge cycle is defined by the end of the system's BRDY# response to the CPU cycle. Note that the APIC through local mode was designed to match the protocol of an 8259A PIC, and an 8259A will always satisfy this requirement.

To ensure INTR is not recognized inadvertantly a second time, deassert INTR no later than the BRDY# of the second INTA cycle and no earlier than the BRDY# of the first INTA cycle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS# and cycle definition pins	An interrupt acknowledge cycle is driven as a result of the INTR pin assertion.
APICEN	When the APICEN configuration input is sampled inactive, this input becomes the INTR interrupt.
LINT0	INTR shares a pin with LINT0.
LOCK#	LOCK# is asserted for interrupt acknowledge cycles.

5.1.38. INV

INV	Invalidation Request
	Determines final state of a cache line as a result of an inquire hit.
	Synchronous Input

Signal Description

The INV input is driven to the Pentium processor during an inquire cycle to determine the final cache line state (S or I) in case of an inquire cycle hit. If INV is returned active (high) to the Pentium processor in the same clock as EADS# is sampled asserted, an inquire hit will result in that line being invalidated. If the INV pin is returned inactive (low), an inquire hit will result in that line being marked Shared (S). If the inquire cycle is a miss in the cache, the INV input has no effect.

If an inquire cycle hits a modified line in the data cache, the line will be written back regardless of the state of INV.

When Sampled

The INV input is sampled with the EADS# of the inquire cycle.

Relation to Other Signals

Pin Symbol	Relative to Other Signals
A31-A5	INV determines if the inquire address driven to the processor on A31-A5 should be invalidated or marked as shared if it is valid in an internal cache.
EADS#	INV is sampled with EADS#.

5.1.39. KEN#

KEN#	Cache Enable
	Indicates to the Pentium® processor whether or not the system can support a cache line fill for the current cycle.
	Synchronous Input

Signal Description

KEN# is the cache enable input. It is used to determine whether the current cycle is cacheable or not and consequently is used to determine cycle length.

When the Pentium processor generates a read cycle that can be cached (CACHE# asserted) and KEN# is active, the cycle will be transformed into a burst cache linefill. During a cache line fill the byte enable outputs should be ignored and valid data must be returned on all 64 data lines. The Pentium processor will expect 32 bytes of valid data to be returned in four BRDY# transfers.

If KEN# is not sampled active, a linefill will not be performed (regardless of the state of CACHE#) and the cycle will be a single transfer read.

Once KEN# is sampled active for a cycle, the cacheability cannot be changed. If a cycle is restarted for any reason after the cacheability of the cycle has been determined, the same cacheability attribute on KEN# must be returned to the processor when the cycle is redriven.

When Sampled

KEN# is sampled once in a cycle to determine cacheability. It is sampled and latched with the earlier of the first BRDY# or NA# of a cycle, however it must meet setup and hold times on every clock edge.

Relation to Other Signals

Pin Symbol	Relative to Other Signals
BRDY#	KEN# is sampled with the earlier of the first BRDY# or NA# for that cycle. Also, in conjunction with the CACHE# input, KEN# determines whether the bus cycle will consist of 1 or 4 transfers (assertions of BRDY#).
CACHE#	KEN# determines cacheability only if the CACHE# pin is asserted.
NA#	KEN# is sampled with the earlier of the first BRDY# or NA# for that cycle.
W/R#	KEN# determines cacheability only if W/R# indicates a read.

5.1.40. LINT1-LINT0

LINT1-LINT0	Local Interrupts 1 and 0
	APIC Programmable Interrupts.
	Asynchronous Inputs

Signal Description

When the local APIC is hardware enabled, these pins become the programmable interrupts (LINT1-LINT0). They can be programmed in software in any of the interrupt modes. Since these pins are the INTR and NMI inputs when the APIC is disabled, it is logical to program the vector table entry for them as ExtINT (i.e. through local mode) and NMI, respectively. In this mode, the interrupt signals are passed on to the processor through the local APIC.

When the local APIC is hardware disabled, these pins are the INTR and NMI inputs for the processor. They bypass the APIC in that case.

When Sampled

LINT1-LINT0 are sampled on every rising clock edge. LINT1-LINT0 are asynchronous inputs, but recognition of LINT1-LINT0 are guaranteed in a specific clock if they are asserted synchronously and meets the setup and hold times. To guarantee recognition if LINT1-LINT0 are asserted asynchronously they must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
APICEN	When the APICEN configuration input is sampled inactive, these inputs become the INTR and NMI interrupts.
INTR	INTR shares a pin with LINT0.
NMI	NMI shares a pin with LINT1.

5.1.41. LOCK#

LOCK#	Bus Lock
	Indicates to the system that the current sequence of bus cycles should not be interrupted.
	Synchronous Input/Output

Signal Description

The bus lock output indicates that the Pentium processor is running a read-modify-write cycle where the external bus must not be relinquished between the read and write cycles. Read-modify-write cycles of this type are used to implement memory based semaphores. Interrupt Acknowledge cycles are also locked.

If a cycle is split due to a misaligned memory operand, two reads followed by two writes may be locked together. When LOCK# is asserted, the current bus master should be allowed exclusive access to the system bus.

The Pentium processor will not allow a bus hold when LOCK# is asserted, but address holds (AHOLD) and BOFF# are allowed. LOCK# is floated during bus hold.

All locked cycles will be driven to the external bus. If a locked address hits a valid location in one of the internal caches, the cache location is invalidated (if the line is in the modified state, it is written back before it is invalidated). Locked read cycles will not be transformed into cache line fill cycles regardless of the state of KEN#.

LOCK# is guaranteed to be deasserted for at least one clock between back to back locked cycles.

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

LOCK# goes active with the ADS# of the first locked bus cycle and goes inactive after the BRDY# is returned for the last locked bus cycle. The LOCK# signal is glitch free.

This signal becomes an input/output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	LOCK# is driven with the ADS# of the first locked cycle.
BOFF#	LOCK# floats one clock after BOFF# is asserted.
BRDY#	LOCK# is deasserted after the last BRDY# of the locked sequence.
HLDA	LOCK# floats when HLDA is asserted.
NA#	ADS# is not asserted to pipeline an additional cycle if LOCK# is asserted, regardless of the state of NA#.
INTR	LOCK# is asserted for interrupt acknowledge cycles.
SCYC	SCYC is driven active if the locked cycle is misaligned.

5.1.42. M/IO#

M/IO#	Memory Input/Output
	Distinguishes a memory access from an I/O access.
	Synchronous Input/Output

Signal Description

The Memory/Input-Output signal is one of the primary bus cycle definition pins. M/IO# distinguishes between memory (M/IO# =1) and I/O (M/IO# =0) cycles.

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

M/IO# is driven valid in the same clock as ADS# and the cycle address. It remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#.

This signal becomes an input/output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	M/IO# is driven to its valid state with ADS#.
BOFF#	M/IO# floats one clock after BOFF# is asserted.
HLDA	M/IO# floats when HLDA is asserted.

5.1.43. NA#

NA#	Next Address
	Indicates that external memory is prepared for a pipelined cycle.
	Synchronous Input

Signal Description

The Next Address input, when active, indicates that external memory is ready to accept a new bus cycle although all data transfers for the current cycle have not yet completed. This is referred to as bus cycle pipelining.

The Pentium processor will drive out a pending cycle in response to NA# no sooner than two clocks after NA# is asserted. The Pentium processor supports up to 2 outstanding bus cycles. ADS# is not asserted to pipeline an additional cycle if LOCK# is asserted, or during a writeback cycle. In addition, ADS# will not be asserted to pipeline a locked cycle or a writeback cycle into the current cycle.

NA# is latched internally, so once it is sampled active during a cycle, it need not be held active to be recognized. The KEN#, and WB/WT# inputs for the current cycle are sampled with the first NA#, if NA# is asserted before the first BRDY# of the current cycle.

When Sampled

NA# is sampled in all T2, TD and T2P clocks.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	If NA# is sampled asserted and an internal bus request is pending, the Pentium® processor drives out the next bus cycle and asserts ADS#.
KEN#	KEN# is sampled with the earlier of the first BRDY# or NA# for that cycle.
WB/WT#	WB/WT# is sampled with the earlier of the first BRDY# or NA# for that cycle.
LOCK#	ADS# is not asserted to pipeline an additional cycle if LOCK# is asserted, regardless of the state of NA#.
BOFF#	If NA# and BOFF# are asserted simultaneously, BOFF# takes priority and NA# is ignored.

5.1.44. NMI

NMI	Non-Maskable Interrupt
	Indicates that an external non-maskable interrupt has been generated.
	Asynchronous Input

Signal Description

The Non-Maskable interrupt request input indicates that an external non-maskable interrupt has been generated. Asserting NMI causes an interrupt with an internally supplied vector value of 2. External interrupt acknowledge cycles are not generated.

If NMI is asserted during the execution of the NMI service routine, it will remain pending and will be recognized after the IRET is executed by the NMI service routine. At most, one assertion of NMI will be held pending. If NMI is reasserted prior to the NMI service routine entry, it will be ignored.

When the local APIC is hardware enabled, this pin becomes the programmable interrupt LINT1. It can be programmed in software in any of the interrupt modes. Since this pin is the NMI input when the APIC is disabled, it is logical to program the vector table entry for this pin as NMI. In this mode, the interrupt signal is passed on to the processor through the local APIC.

When the local APIC is hardware disabled, this pin is the NMI input for the processor. It bypasses the APIC in that case.

When Sampled

NMI is sampled on every rising clock edge. NMI is rising edge sensitive. Recognition of NMI is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. To guarantee recognition if NMI is asserted asynchronously, it must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor and remain asserted for a minimum pulse width of two clocks.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
APICEN	When the APICEN configuration input is sampled inactive, this input becomes the NMI interrupt.
LINT1	NMI shares a pin with LINT1.

5.1.45. PBGNT#

PBGNT#	Dual Processor Bus Grant
	Indicates to the LRM processor that it will become the MRM in the next clock.
	Synchronous Input (to the Least Recent Master, LRM, processor) Synchronous Output (of the Most Recent Master, MRM, processor)

Signal Description

Two Pentium processors, when configured as dual processors, will arbitrate for the system bus via two private arbitration pins (PBREQ# and PBGNT#). The processor that currently owns the system bus is referred to as the MRM processor. The processor that does not own the bus is referred to as the LRM processor.

PBGNT# is used by the dual processing private arbitration mechanism to indicate that bus ownership will change in the next clock. The LRM processor will request ownership of the processor bus by asserting the private arbitration request pin, PBREQ#. The processor that is currently the MRM and owns the bus, will grant the bus to the LRM as soon as any pending bus transactions have completed. The MRM will notify that the LRM can assume ownership by asserting the private arbitration grant pin, PBGNT#. The PBGNT# pin is always the output of the MRM and an input to the LRM.

NOTE

In a single socket system design, PBGNT# pin should be left NC. For proper operation, PBGNT# must not be loaded by the system.

When Sampled/Driven

PBGNT# is driven by the MRM processor in response to the PBREQ# signal from the LRM processor. It is asserted following the completion of the current cycle on the processor bus, or in the clock following the request if the bus is idle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
PBREQ#	PBGNT# is asserted in response to a bus request, PBREQ#, from the LRM processor.
A[31:3], AP, BE[7:0]#, CACHE#, D/C#, M/IO#, PCD, PWT, SCYC, W/R#	These signals are tristated for one CLK in response to PBGNT# (when the MRM becomes the LRM).

5.1.46. PBREQ#

PBREQ#	Dual Processor Bus Request
	Indicates to the MRM processor that the LRM processor requires ownership of the bus.
	Synchronous Input (to the Most Recent Master, MRM, processor) Synchronous Output (of the Least Recent Master, LRM, processor)

Signal Description

Two Pentium processors, when configured as dual processors, will arbitrate for the system bus via two private arbitration pins (PBREQ# and PBGNT#). The processor that currently owns the system bus is referred to as the MRM processor. The processor that does not own the bus is referred to as the LRM processor.

PBREQ# is used by the dual processing private arbitration mechanism to indicate that the LRM processor requests bus ownership. The processor that is currently the MRM and owns the bus, will grant the bus to the LRM as soon as any pending bus transactions have completed. The MRM will notify that the LRM can assume ownership by asserting the private arbitration grant pin, PBGNT#. The PBREQ**#** pin is always the output of the LRM and an input to the MRM.

NOTE

In a single socket system design, PBREQ# pin should be left NC. For proper operation, PBREQ# must not be loaded by the system.

When Sampled/Driven

PBREQ# is driven by the LRM processor, and sampled by the MRM processor.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
PBGNT#	PBGNT# is asserted in response to a bus request, PBREQ#, from the LRM processor.

5.1.47. PCD

PCD	Page Cacheability Disable
	Externally reflects the cacheability paging attribute bit in CR3, PDE, or PTE.
	Output

Signal Description

PCD is driven to externally reflect the cache disable paging attribute bit for the current cycle. PCD corresponds to bit 4 of CR3, the Page Directory Entry, or the Page Table Entry. For cycles that are not paged when paging is enabled (for example I/O cycles), PCD corresponds to bit 4 in CR3. In real mode or when paging is disabled, the PCD pin reflects the cache disable bit in control register 0 (CR0.CD).

PCD is masked by the CD (cache disable) bit in CR0. When CD=1, the Pentium processor forces PCD high. When CD=0, PCD is driven with the value of the Page Table Entry/Directory.

The purpose of PCD is to provide an external cacheability indication on a page by page basis.

When Driven

The PCD pin is driven valid in the same clock as ADS# and the cycle address. It remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	PCD is driven valid with ADS#.
BOFF#	PCD floats one clock after BOFF# is asserted.
HLDA	PCD floats when HLDA is asserted.

5.1.48. PCHK#

PCHK#	Data Parity Check
	Indicates the result of a parity check on a data read.
	Synchronous Output

Signal Description

The data parity check pin indicates the result of a parity check on a data read. Data parity is checked during code reads, memory reads, and I/O reads. Data parity is not checked during the first Interrupt Acknowledge cycle. PCHK# indicates the parity status only for the bytes on which valid data is expected. Parity is checked for all data bytes for which a byte enable is asserted. In addition, during a cache linefill, parity is checked on the entire data bus regardless of the state of the byte enables.

PCHK# is driven low two clocks after BRDY# is returned if incorrect parity was returned.

Driving PCHK# is the only effect that bad data parity has on the Pentium processor unless PEN# is also asserted. The data returned to the processor is not discarded.

If PEN# is asserted when a parity error occurs, the cycle address and type will be latched in the MCA and MCT registers. If in addition, the MCE bit in CR4 is set, a machine check exception will be taken.

It is the responsibility of the system to take appropriate actions if a parity error occurs. If parity checks are not implemented in the system, the PCHK# pin may be ignored, and PEN# pulled high (or CR4.MCE cleared).

When operating in dual processing mode, the PCHK# signal can be asserted either 2 OR 3 CLKs following incorrect parity being detected on the data bus. When operating in Dual Processing mode, the PCHK# pin circuit is implemented as a weak driving high output that operates similar to an open drain output. This implementation allows connection of the two processor PCHK# pins together in a dual processing system with no ill effects. Nominally, this circuit acts like a 360 Ohm resistor tied to V_{CC}.

When Sampled/Driven

PCHK# is driven low two clocks after BRDY# is returned if incorrect parity was returned. PCHK# remains low one clock for each clock in which a parity error was detected. At all other times PCHK# is inactive (high). PCHK# is not floated during bus HOLD or BOFF#. PCHK# is a glitch free signal.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BRDY#	PCHK# is driven to its valid level two clocks after the assertion of BRDY#.
D63-D0	The DP7-DP0 pins are used to create even parity with D63-D0. If even parity is not returned, the PCHK# pin is asserted.
DP7-DP0	Even data parity with D63-D0 should be returned on to the Pentium® processor on the dual processor pin. If even parity is not returned, the PCHK# pin is asserted.

5.1.49. PHIT#

PHIT#	Private Inquire Cycle Hit/Miss Indication
	Indicates whether a private, dual processor, inquire cycle resulted in a hit or miss.
	Synchronous Input (to the Most Recent Master, MRM, processor) Synchronous Output (of the Least Recent Master, LRM, processor)

Signal Description

A private snoop interface has been added to the Pentium processor for use in dual processing. The interface consists of two pins (PHIT# and PHITM#).

The LRM processor will initiate a snoop sequence for all ADS# cycles that are initiated by the MRM. The LRM processor will assert the private hit indication (PHIT#) if the data requested by the MRM matches a valid cache line in the LRM. In addition, if the data requested by the MRM matches a valid cache line in the LRM that is in the modified state, the LRM will also assert the PHITM# signal. The system snooping indication signals (HIT#, HITM#) will not change state as a result of a private snoop.

The MRM will use an assertion of the PHIT# signal as an indication that the requested data is being shared with the LRM. Independent of the WB/WT# pin, a cache line will be placed in the shared state if PHIT# is asserted. This will make all subsequent writes to that line externally visible until the state of the line becomes exclusive (E or M states). In a uni-processor system, the line may have been placed in the cache in the E state. In this situation, all subsequent writes to that line will not be visible on the bus until the state is changed to I.

PHIT# will also be driven by the LRM during external snoop operations (e.g., following EADS#) to indicate the private snoop results.

NOTE

In a single socket system, PHIT# pin should be left NC. For proper operation, PHIT# must not be loaded by the system.

When Sampled/Driven

PHIT# is driven by the LRM processor, and sampled by the MRM processor. It is asserted within two clocks following an assertion of ADS# or EADS#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A[31:5]	PHIT# is driven to indicate whether the private inquire address driven on A[31:5] is valid in the LRM's on-chip cache.
ADS#	PHIT# is driven within 2 clocks after ADS# is sampled asserted to indicate the outcome of the private inquire cycle.
EADS#	PHIT# is driven within 2 clocks after EADS# is sampled asserted to indicate the outcome of the external inquire cycle.
PHITM#	PHITM# is never asserted without PHIT# also being asserted.
WB/WT#	The state of the WB/WT# pin will be ignored by the MRM if the PHIT# pin is sampled active, and the cache line placed in the shared state.

5.1.50. PHITM#

PHITM#	Private Inquire Cycle Hit/Miss to a Modified Line Indication
	Indicates whether a private, dual processor, inquire cycle resulted in a hit or miss to a Modified line.
	Synchronous Input (to the Most Recent Master, MRM, processor) Synchronous Output (of the Least Recent Master, LRM, processor)

Signal Description

A private snoop interface has been added to the Pentium processor for use in dual processing. The interface consists of two pins (PHIT# and PHITM#).

The LRM processor will initiate a snoop sequence for all ADS# cycles that are initiated by the MRM. The LRM processor will assert the private hit indication (PHIT#) if the data requested by the MRM matches a valid cache line in the LRM. In addition, if the data requested by the MRM matches a valid cache line in the LRM that is in the modified state, the LRM will also assert the PHITM# signal. The system snooping indication signals (HIT#, HITM#) will not change state as a result of a private snoop.

PHITM# will also be driven by the LRM during external snoop operations (e.g. following EADS#) to indicate the private snoop results.

NOTE

In a single socket system, PHITM# pin should be left NC. For proper operation, PHITM# must not be loaded by the system.

When Sampled/Driven

PHITM# is driven by the LRM processor, and sampled by the MRM processor. It is asserted within two clocks following an assertion of ADS# or EADS#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A[31:5]	PHITM# is driven to indicate whether the private inquire address driven on A[31:5] is modified in the LRM's on-chip cache.
ADS#	PHITM# is driven within 2 clocks after ADS# is sampled asserted to indicate the outcome of the private inquire cycle.
EADS#	PHITM# is driven within 2 clocks after EADS# is sampled asserted to indicate the outcome of the external inquire cycle.
PHIT#	PHITM# is never asserted without PHIT# also being asserted.

5.1.51. PICCLK

PICCLK	Processor Interrupt Controller Clock
	This pin drives the clock for the APIC serial data bus operation.
	Input

Signal Description

This pin provides the clock timings for the on-chip APIC unit of the processor. This clock input controls the frequency for the APIC operation and data transmission on the 2-wire APIC serial data bus. All the timings on APIC bus are referenced to this clock.

When hardware disabled, PICCLK must be tied high.

Note that the PICCLK signal on the Pentium processor with MMX technology is 3.3V tolerant, while on the Pentium processor (75/90/100/120/133/150/166/200) the PICCLK input is 5.0V tolerant.

When Sampled

PICCLK is a clock signal and is used as a reference for sampling the APIC data signals.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
APICEN	PICCLK must be tied or driven high when APICEN is sampled low at the falling edge of RESET.
PICD0-1	External timing parameters for the PICD0-1 pins are measured with respect to this clock.

5.1.52. PICD1-PICD0

PICD1-PICD0	Processor Interrupt Controller Data
	These are the data pins for the 3-wire APIC bus.
	Synchronous Input/Output to PICCLK
	Needs external pull-up resistors.

Signal Description

The PICD1-PICD0 are bi-directional pins which comprise the data portion of the 3-wire APIC bus.

When Sampled/Driven

These signals are sampled with the rising edge of PICCLK.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
APICEN	PICD1 shares a pin with APICEN.
DPEN#	PICD0 shares a pin with DPEN#.

5.1.53. PEN#

PEN#	Parity Enable
	Indicates to the Pentium® processor that the correct data parity is being returned by the system. Determines if a Machine Check Exception should be taken if a data parity error is detected.
	Synchronous Input

Signal Description

The PEN# input (along with CR4.MCE) determines whether a machine check exception will be taken as a result of a data parity error on a read cycle. If this pin is sampled active in the clock a data parity error is detected, the Pentium processor will latch the address and control signals of the cycle with the parity error in the machine check registers. If, in addition, the machine check enable bit in CR4 is set to "1," the Pentium processor will vector to the machine check exception before the beginning of the next instruction. If this pin is sampled inactive, it does not prevent PCHK# from being asserted in response to a bus parity error. If systems are using PCHK#, they should be aware of this usage of PEN#.

This pin may be tied to V_{SS}.

When Sampled

This signal is sampled when BRDY# is asserted for memory and I/O read cycles and the second interrupt acknowledge cycle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BRDY#	PEN# is sampled with BRDY# for read cycles.
D63-D0	The DP7-DP0 pins are used to create even parity with D63-D0. If even parity is not returned, and PEN# is enabled, the cycle will be latched and an MCE will be taken if CR4.MCE = 1.
DP7-DP0	Even data parity with D63-D0 should be returned to the Pentium® processor on the dual-processor pins. If even parity is not returned, and PEN# is enabled, the cycle will be latched and a MCE will be taken if CR4.MCE = 1.

5.1.54. PM[1:0]

PM/BP1-0	Performance Monitoring
	PM1-0 externally indicate the status of the performance monitor counter.
	Output pins

Signal Description

The performance monitoring pins can be individually configured to externally indicate either that the associated performance monitoring counter has incremented or that it has overflowed. PM1 indicates the status of CTR1; PM0 indicates the status of CTR0.

BP1 and BP0 are multiplexed with the Performance Monitoring pins (PM1 and PM0). The PB1 and PB0 bits in the Debug Mode Control Register determine if the pins are configured as breakpoint or performance monitoring pins. The pins come out of reset configured for performance monitoring .

When Driven

The BP[3:2], PM/BP[1:0] pins are driven in every clock and are not floated during bus HOLD or BOFF#.

NOTE

The PM1/PM0 pins externally indicate the status of the performance monitoring counters on the Pentium processor. These counters are undefined after RESET, and must be cleared or pre-set (using the WRMSR instruction) before they are assigned to specific events.

However, it is possible for these pins to toggle even during RESET. This may occur ONLY if the RESET pin was asserted while the Pentium processor was in the process of counting a particular performance monitoring event. Since the event counters continue functioning until the CESR (Control and Event Select Register) is cleared by RESET, it is possible for the event counters to increment even during RESET. Externally, the state of the event counters would also be reflected on the PM1/PM0 pins. Any assertion of the PM1/PM0 pins during RESET should be ignored until after the start of the first bus cycle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BP1-BP0	PM1 and PM0 are share pins with BP1 and BP0.

5.1.55. PRDY

PRDY	PRDY
	For use with the Intel debug port.
	Output

Signal Description

The PRDY pin is provided for use with the Intel debug port described in the “Debugging” chapter.

When Driven

This output is always driven by the Pentium processor. It is not floated during bus HOLD or BOFF#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
R/S#	R/S# is also used with the Intel debug port.

5.1.56. PWT

PWT	Page Writethrough
	Externally reflects the writethrough paging attribute bit in CR3, PDE, or PTE.
	Output

Signal Description

PWT is driven to externally reflect the cache writethrough paging attribute bit for the current cycle. PWT corresponds to bit 3 of CR3, the Page Directory Entry, or the Page Table Entry. For cycles that are not paged when paging is enabled (for example I/O cycles), PWT corresponds to bit 3 in CR3. In real mode or when paging is disabled, the Pentium processor drives PWT low.

PWT can override the effect of the WB/WT# pin. If PWT is asserted for either reads or writes, the line is saved in, or remains in, the Shared (S) state.

When Driven

The PWT pin is driven valid in the same clock as ADS# and the cycle address. It remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	PWT is driven valid with ADS#.
BOFF#	PWT floats one clock after BOFF# is asserted.
HLDA	PWT floats when HLDA is asserted.
WB/WT#	PWT is used in conjunction with the WB/WT# pin to determine the MESI state of cache lines.

5.1.57. R/S#

R/S#	R/S#
	For use with the Intel debug port.
	Asynchronous Input

Signal Description

The R/S# pin is provided for use with the Intel debug port described in the “Debugging” chapter.

When Sampled

This pin should not be driven except in conjunction with the Intel debug port.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
PRDY	PRDY is also used with the Intel debug port.

5.1.58. RESET

RESET	Reset
	Forces the Pentium® processor to begin execution at a known state.
	Asynchronous Input (Normal, Uni-processor, mode) Synchronous Input (Dual processor mode)

Signal Description

The RESET input forces the Pentium processor to begin execution at a known state. All the Pentium processor internal caches (code and data caches, the translation lookaside buffers, branch target buffer and segment descriptor cache) will be invalidated upon the RESET. Modified lines in the data cache are not written back. When RESET is asserted, the Pentium processor will immediately abort all bus activity and perform the RESET sequence. The Pentium processor starts execution at FFFFFFF0H.

When RESET transitions from high to low, FLUSH# is sampled to determine if tristate test mode will be entered, FRCMC# is sampled to determine if the Pentium processor will be configured as a master or a checker (only on the Pentium processor (75/90/100/120/133/150/166/200), and INIT is sampled to determine if BIST will be run.

When Sampled/Driven

RESET is sampled on every rising clock edge. RESET must remain asserted for a minimum of 1 millisecond after V_{CC} and CLK have reached their AC/DC specifications for the "cold" or "power on" reset. During power up, RESET should be asserted while V_{CC} is approaching nominal operating voltage (the simplest way to insure this is to place a pullup resistor on RESET). RESET must remain active for at least 15 clocks while V_{CC} and CLK are within their operating limits for a "warm reset." Recognition of RESET is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. To guarantee recognition if RESET is asserted asynchronously, it must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor.

FLUSH#, FRCMC# and INIT are sampled when RESET transitions from high to low to determine if tristate test mode or checker mode will be entered, or if BIST will be run. If RESET is driven synchronously, these signals must be at their valid level and meet setup and hold times on the clock before the falling edge of RESET. If RESET is driven asynchronously, these signals must be at their valid level two clocks before and after RESET transitions from high to low.

When operating in a dual processing system, RESET is sampled synchronously to the rising CLK edge to ensure both processors recognize the falling edge in the same clock.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
APICEN	APICEN is sampled at the falling edge of RESET.
BE3#-BE0#	During reset the BE3#-BE0# pins are sampled to determine the APIC ID. Following RESET, they function as Byte Enable outputs.
BF[1:0]	BF[1:0] are sampled at the falling edge of RESET.
CPUTYP	CPUTYP is sampled at the falling edge of RESET.
DPEN#	DPEN# is valid during RESET.
FLUSH#	If FLUSH# is sampled low when RESET transitions from high to low, tristate test mode will be entered.
FRCMC#	FRCMC# is sampled when RESET transitions from high to low to determine if the Pentium processor is in Master or Checker mode.
INIT	If INIT is sampled high when RESET transitions from high to low, BIST will be performed.

5.1.59. SCYC

SCYC	Split Cycle Indication
	Indicates that a misaligned locked transfer is on the bus.
	Synchronous Input/Output

Signal Description

The Split Cycle output is activated during misaligned locked transfers. It is asserted to indicate that more than two cycles will be locked together. This signal is defined for locked cycles only. It is undefined for cycles which are not locked.

The Pentium processor defines misaligned transfers as a 16-bit or 32-bit transfer which crosses a 4-byte boundary, or a 64-bit transfer which crosses an 8-byte boundary.

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

SCYC is only driven during the length of the locked cycle that is split. SCYC is asserted with the first ADS# of a misaligned split cycle and remains valid until the earlier of the last BRDY# of the last split cycle or the clock after NA# of the last split cycle.

This signal becomes an input/output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	SCYC is driven valid in the same clock as ADS#.
BOFF#	SCYC is floated one clock after BOFF# is asserted.
HLDA	SCYC is floated when HLDA is asserted.
LOCK#	SCYC is defined for locked cycles only.

5.1.60. SMI#

SMI#	System Management Interrupt
	Latches a System Management Interrupt request.
	Asynchronous Input
	Internal Pullup Resistor

Signal Description

The System Management Interrupt input latches a System Management Interrupt request. After SMI# is recognized on an instruction boundary, the Pentium processor waits for all writes to complete and EWBE# to be asserted, then asserts the SMIACT# output. The processor will then save its register state to SMRAM space and begin to execute the SMM handler. The RSM instruction restores the registers and returns to the user program.

SMI# has greater priority than debug exceptions and external interrupts. This means that if more than one of these conditions occur at an instruction boundary, only the SMI# processing occurs, not a debug exception or external interrupt. Subsequent SMI# requests are not acknowledged while the processor is in system management mode (SMM). The first SMI# interrupt request that occurs while the processor is in SMM is latched, and serviced when the processor exits SMM with the RSM instruction. Only one SMI# will be latched by the CPU while it is in SMM.

When Sampled

SMI# is sampled on every rising clock edge. SMI# is a falling edge sensitive input. Recognition of SMI# is guaranteed in a specific clock if it is asserted synchronously and meets the setup and hold times. To guarantee recognition if SMI# is asserted asynchronously, it must have been deasserted for a minimum of 2 clocks before being returned active to the Pentium processor and remain asserted for a minimum pulse width of two clocks.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
SMIACT#	When the SMI# input is recognized, the Pentium® processor asserts SMIACT#.

5.1.61. SMIACT#

SMIACT#	System Management Interrupt Active
	Indicates that the processor is operating in SMM.
	Synchronous Output

Signal Description

The System Management Interrupt Active output is asserted in response to the assertion of SMI#. It indicates that the processor is operating in System Management Mode (SMM). It will remain active (low) until the processor executes the RSM instruction to leave SMM.

When the system is operating in dual processing mode, the D/P# signal toggles based upon whether the Primary or Dual processor owns the bus (MRM). The SMIACT# pins may be tied together or be used separately to insure SMRAM access by the correct processor.

CAUTION

If SMIACT# is used separately, note that the SMIACT# signal is only driven by the processor when it is the MRM (so this signal must be qualified with the D/P# signal).

Connecting the SMIACT# signals on the Primary and Dual processors together is strongly recommended for operation with the Dual processor and upgradability with the future Pentium OverDrive processor.

In dual processing systems, SMIACT# may not remain low (e.g., may toggle) if both processors are not in SMM mode. The SMIACT# signal is asserted by either the Primary or Dual processor based on two conditions: the processor is in SMM mode and is the bus master (MRM). If one processor is executing in normal address space, the SMIACT# signal will go inactive when that processor is MRM. The LRM processor, even if in SMM mode, will not drive the SMIACT# signal low.

When Sampled/Driven

SMIACT# is driven active in response to the assertion of SMI# after all internally pending writes are complete and the EWBE# pin is active (low). It will remain active (low) until the processor executes the RSM instruction to leave SMM. This signal is always driven. It does not float during bus HOLD or BOFF#.

When operating in dual processing mode, the SMIACT# output must be sampled with an active ADS# and qualified with the D/P# signal to determine which Pentium processor (i.e., the Primary or Dual) is driving the SMM cycle.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	SMIACT# should be sampled with an active ADS# during dual processing operation.
D/P#	When operating in dual processing mode, D/P# qualifies the SMIACT# SMM indicator.
EWBE#	SMIACT# is not asserted until EWBE# is active.
SMI#	SMIACT# is asserted when the SMI# is recognized.

5.1.62. STPCLK#

STPCLK#	Stop Clock Pin
	Used to stop the internal processor clock and consume less power.
	Asynchronous Input

Signal Description

Assertion of STPCLK# causes the Pentium processor to stop its internal clock and consume less power while still responding to interprocessor and external snoop requests. This low-power state is called the stop grant state. When the CPU recognizes a STPCLK# interrupt, the CPU will do the following:

1. Wait for all instructions being executed to complete.
2. Flush the instruction pipeline of any instructions waiting to be executed.
3. Wait for all pending bus cycles to complete and EWBE# to go active.
4. Drive a special bus cycle (stop grant bus cycle) to indicate that the clock is being stopped.
5. Enter low power mode.

The stop grant bus cycle consists of the following signal states: M/IO# = 0, D/C# = 0, W/R# = 1, Address Bus = 0000 0010H (A_4 = 1), BE7#-BE0# = 1111 1011, Data bus = undefined.

STPCLK# must be driven high (not floated) to exit the stop grant state. The rising edge of STPCLK# will tell the CPU that it can return to program execution at the instruction following the interrupted instruction.

When Sampled/Driven

STPCLK# is treated as a level triggered interrupt to the Pentium processor and is prioritized below all of the external interrupts. When the Pentium processor recognizes the STPCLK# interrupt, the processor will stop execution on the instruction boundary following the STPCLK# assertion.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
A4, Cycle Control signals (M/IO#, D/C#, W/R#, BE7#-BE0#, D/P#)	The Stop Grant Special Bus Cycle is driven on these pins in response to an assertion of the STPCLK# signal. M/IO# = 0, D/C# = 0, W/R# = 1. Address Bus = 0000 0010H (A4 = 1), BE7#-BE0# = 1111 1011.
EWBE#	After STPCLK# has been recognized, all pending cycles must be completed and EWBE# must go active before the internal clock will be disabled.
External Interrupt signals (FLUSH#, INIT, INTR, NMI, R/S#, SMI#)	While in the Stop Grant state, the CPU will latch transitions on the external interrupt signals. All of these interrupts are taken after the de-assertion of STPCLK#. The CPU requires that INTR be held active until the CPU issues an interrupt acknowledge cycle in order to guarantee recognition.
HLDA	The CPU will not respond to a STPCLK# request from a HLDA state because it cannot generate a Stop Grant cycle.

5.1.63. TCK

TCK	Test Clock Input
	Provides Boundary Scan clocking function.
	Input

Signal Description

This is the Testability Clock input that provides the clocking function for the Pentium processor boundary scan in accordance with the boundary scan interface (IEEE Std 1149.1). It is used to clock state information and data into and out of the Pentium processor during boundary scan. State select information and data are clocked into the Pentium processor on the rising edge of TCK on TMS and TDI inputs respectively. Data is clocked out of the Pentium processor on the falling edge of TCK on TDO.

When TCK is stopped in a low state, the boundary scan latches retain their state indefinitely. When boundary scan is not used, TCK should be tied high or left as a no-connect.

When Sampled

TCK is a clock signal and is used as a reference for sampling other boundary scan signals.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
TDI	Serial data is clocked into the processor on the rising edge of TCK.
TDO	Serial data is clocked out of the processor on the falling edge of TCK.
TMS	TAP controller state transitions occur on the rising edge of TCK.

5.1.64. TDI

TDI	Test Data Input
	Input to receive serial test data and instructions.
	Synchronous Input to TCK

Signal Description

This is the serial input for the Boundary Scan test logic. TAP instructions and data are shifted into the Pentium processor on the TDI pin on the rising edge of TCK when the TAP controller is in the SHIFT-IR and SHIFT-DR states. During all other states, TDI is a “don’t care.”

An internal pull-up resistor is provided on TDI to ensure a known logic state if an open circuit occurs on the TDI path. Note that when “1” is continuously shifted into the instruction register, the BYPASS instruction is selected.

When Sampled

TDI is sampled on the rising edge of TCK during the SHIFT-IR and SHIFT-DR states. During all other states, TDI is a “don’t care.”

Relation to Other Signals

Pin Symbol	Relation to Other Signals
TCK	TDI is sampled on the rising edge of TCK.
TDO	In the SHIFT-IR and SHIFT-DR TAP controller states, TDO contains the output data of the register being shifted, and TDI provides the input.
TMS	TDI is sampled only in the SHIFT-IR and SHIFT DR states (controlled by TMS).

5.1.65. TDO

TDO	Test Data Output
	Outputs serial test data and instructions.
	Output

Signal Description

This is the serial output of the Boundary Scan test logic. TAP instructions and data are shifted out of the Pentium processor on the TDO pin on the falling edge of TCK when the TAP controller is in the SHIFT-IR and SHIFT-DR states. During all other states, the TDO pin is driven to the high impedance state to allow connecting TDO of different devices in parallel.

When Driven

TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times, TDO is driven to the high impedance state. TDO does not float during bus HOLD or BOFF#.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
TCK	TDO is driven on the falling edge of TCK.
TDI	In the SHIFT-IR and SHIFT-DR TAP controller states, TDI provides the input data to the register being shifted, and TDO provides the output.
TMS	TDO is driven only in the SHIFT-IR and SHIFT DR states (controlled by TMS).

5.1.66. TMS

TMS	Test Mode Select
	Controls TAP controller state transitions.
	Synchronous Input to TCK

Signal Description

This a Boundary Scan test logic control input. The value of this input signal sampled at the rising edge of TCK controls the sequence of TAP controller state changes.

To ensure deterministic behavior of the TAP controller, TMS is provided with an internal pullup resistor. If boundary scan is not used, TMS may be tied to V_{CC} or left unconnected.

When Sampled

TMS is sampled on every rising edge of TCK.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
TCK	TMS is sampled on every rising edge of TCK.
TDI	TDI is sampled only in the SHIFT-IR and SHIFT DR states (controlled by TMS).
TDO	TDO is driven only in the SHIFT-IR and SHIFT DR states (controlled by TMS).

5.1.67. TRST#

TRST#	Test Reset
	Allows the TAP controller to be asynchronously initialized.
	Asynchronous Input

Signal Description

This is a Boundary Scan test logic reset or initialization pin. When asserted, it allows the TAP controller to be asynchronously initialized. When asserted, TRST# will force the TAP controller into the Test Logic Reset State. When in this state, the test logic is disabled so that normal operation of the device can continue unhindered. During initialization, the Pentium processor initializes the instruction register with the IDCODE instruction.

An alternate method of initializing the TAP controller is to Drive TMS high for at least 5 TCK cycles. In addition, the Pentium processor implements a power on TAP controller reset function. When the Pentium processor is put through its normal power on/RESET function, the TAP controller is automatically reset by the processor. The user does not have to assert the TRST# pin or drive TMS high after the falling edge of RESET.

When Sampled

TRST# is an asynchronous input.

Relation to Other Signals

None

5.1.68. V_{CC}

V_{CC}	Supply Voltage for Pentium® processor (75/90/100/120/133/150/166/200)
	V_{CC} is used to supply power to the Pentium processor.
	Power Input

Signal Description

The Pentium processor (75/90/100/120/133/150/166/200) and the future Pentium OverDrive processor require 3.3V V_{CC} inputs.

5.1.69. $V_{CC}2$

$V_{CC}2$	Core Supply Voltage for Pentium® processor with MMX™ technology
	$V_{CC}2$ is used to supply the core of the Pentium processor with MMX technology
	Power Input

Signal Description

The Pentium processor with MMX technology requires a 2.8V $V_{CC}2$ (core) voltage.

5.1.70. $V_{CC}3$

$V_{CC}3$	I/O Supply Voltage for Pentium® processor with MMX™ technology
	$V_{CC}3$ is used to supply the I/O of the Pentium processor with MMX technology
	Power Input

Signal Description

The Pentium processor with MMX technology requires a 3.3V $V_{CC}3$ (I/O) voltage. This enables compatibility with Pentium processor (75/90/100/120/133/150/166/200) system components.

5.1.71. VCC2DET#

VCC2DET#	$V_{CC}2$ Detect
	VCC2DET# can be used in flexible motherboard implementations to configure the voltage regulator output set-point appropriately for the $V_{CC}2$ inputs of the Pentium® processor with MMX™ technology.
	Output

NOTES: This pin is defined only on the Pentium processor with MMX technology. This pin is an INC on the Pentium processor (75/90/100/120/133/150/166/200).

Signal Description

The Pentium processor with MMX technology requires 2.8V on the $V_{CC}2$ pins and 3.3V on the $V_{CC}3$ pins. By using the VCC2DET# signal the system can adjust the core voltage to the processor when a Pentium processor with MMX technology is inserted into Socket 7.

VCC2DET# is driven active (low) to indicate that a Pentium processor with MMX technology is installed in the system and can be used in flexible motherboard designs to configure the voltage regulator output set-point appropriately for the $V_{CC}2$ inputs of the Pentium processor with MMX technology.

When Sampled/Driven

This pin is internally strapped to V_{SS}.

5.1.72. W/R#

W/R#	Write/Read
	Distinguishes a Write cycle from a Read cycle.
	Synchronous Input/Output

Signal Description

The Write/Read signal is one of the primary bus cycle definition pins. W/R# distinguishes between write (W/R# = 1) and read cycles (W/R# = 0).

When operating in dual processing mode, the Pentium processor uses this signal for private snooping.

When Sampled/Driven

W/R# is driven valid in the same clock as ADS# and the cycle address. It remains valid from the clock in which ADS# is asserted until the earlier of the last BRDY# or the clock after NA#.

This signal becomes an input/output when two Pentium processors are operating together in dual processing mode.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
ADS#	W/R# is driven to its valid state with ADS#.
BOFF#	W/R# floats one clock after BOFF# is asserted.
HLDA	W/R# floats when HLDA is asserted.
KEN#	KEN# determines cacheability only if W/R# indicates a read.

5.1.73. WB/WT#

WB/WT#	Writeback/Writethrough
	This pin allows a cache line to be defined as writeback or writethrough on a line by line basis.
	Synchronous Input

Signal Description

This pin allows a cache line to be defined as writeback or writethrough on a line by line basis. As a result, in conjunction with the PWT pin, it controls the MESI state in which the line is saved.

If WB/WT# is sampled high during a memory read cycle and the PWT pin is low, the line is saved in the Exclusive (E) state in the cache. If WB/WT# is sampled low during a memory read cycle, the line is saved in the Shared (S) state in the cache.

If WB/WT# is sampled high during a write to a shared line in the cache and the PWT pin is low, the line transitions to the E state. If WB/WT# is sampled low during a write to a shared line in the cache, the line remains in the S state.

If for either reads or writes the PWT pin is high, the line is saved in, or remains in, the Shared (S) state.

When Sampled

This pin is sampled with KEN# on the clock in which NA# or the first BRDY# is returned, however it must meet setup and hold times on every clock edge.

Relation to Other Signals

Pin Symbol	Relation to Other Signals
BRDY# NA#	WB/WT# is sampled with the earlier of the first BRDY# or NA# for that cycle.
PWT	If PWT is high, WB/WT# is a “don’t care.”

6

Bus Functional Description

CHAPTER 6
BUS FUNCTIONAL DESCRIPTION

Both the Pentium processor (75/90/100/120/133/130/166/200) and the Pentium processor with MMX technology support the same bus functionality. The Pentium processor bus is designed to support a 528-Mbyte/sec data transfer rate at 66 MHz. All data transfers occur as a result of one or more bus cycles. This chapter describes the Pentium processor bus cycles and the Pentium processor data transfer mechanism.

6.1. PHYSICAL MEMORY AND I/O INTERFACE

Pentium processor memory is accessible in 8-, 16-, 32-, and 64-bit quantities. Pentium processor I/O is accessible in 8-, 16-, and 32-bit quantities. The Pentium processor can directly address up to 4 Gbytes of physical memory, and up to 64 Kbytes of I/O.

In hardware, memory space is organized as a sequence of 64-bit quantities. Each 64-bit location has eight individually addressable bytes at consecutive memory addresses (see Figure 6-1).

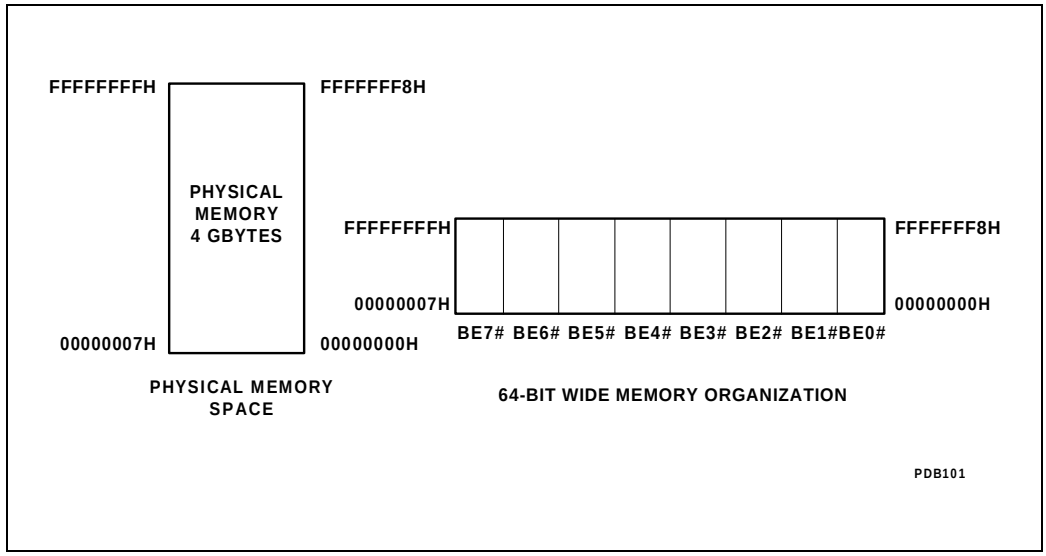


Figure 6-1. Memory Organization

The I/O space is organized as a sequence of 32-bit quantities. Each 32-bit quantity has four individually addressable bytes at consecutive memory addresses. See Figure 6-2 for a conceptual diagram of the I/O space.

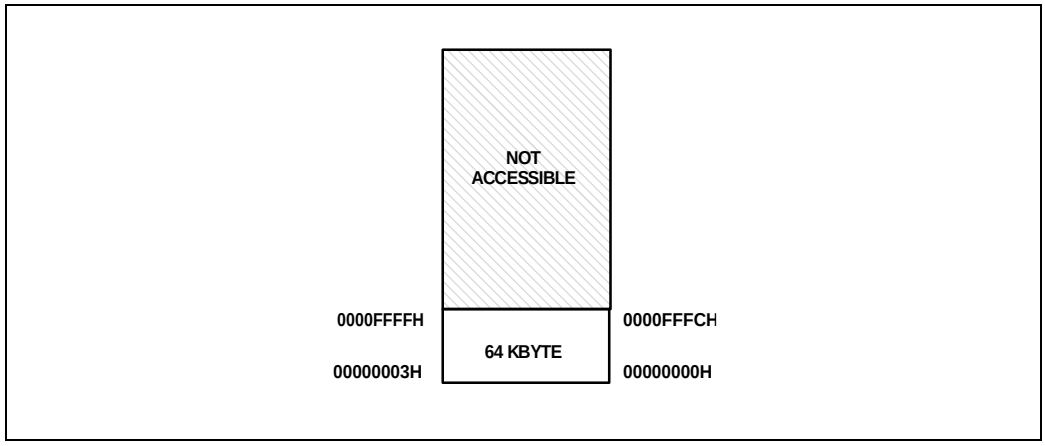


Figure 6-2. I/O Space Organization

Sixty-four-bit memories are organized as arrays of physical quadwords (8-byte words). Physical quadwords begin at addresses evenly divisible by 8. The quadwords are addressable by physical address lines A31-A3.

Thirty-two-bit memories are organized as arrays of physical dwords (4-byte words). Physical dwords begin at addresses evenly divisible by 4. The dwords are addressable by physical address lines A31-A3 and A2. A2 can be decoded from the byte enables according to Table 6-2.

Sixteen-bit memories are organized as arrays of physical words (2-byte words). Physical words begin at addresses evenly divisible by 2. The words are addressable by physical address lines A31-A3, A2-A1, BHE#, and BLE#. A2 and A1 can be decoded from the byte enables according to Table 6-2, BHE# and BLE# can be decoded from the byte enables according to Table 6-3 and Table 6-4.

To address 8-bit memories, the lower 3 address lines (A2-A0) must be decoded from the byte enables as indicated in Table 6-2.

6.2. DATA TRANSFER MECHANISM

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte, word, dword, and quadword lengths may be transferred. Data may be accessed at any byte boundary, but two cycles may be required for misaligned data transfers. The Pentium processor considers a 2-byte or 4-byte operand that crosses a 4-byte boundary to be misaligned. In addition, an 8-byte operand that crosses an 8-byte boundary is misaligned.

Like the Intel486 CPU, the Pentium processor address signals are split into two components. High-order address bits are provided by the address lines A31-A3. The byte enables BE7#-BE0# form the low-order address and select the appropriate byte of the 8-byte data bus.

The byte enable outputs are asserted when their associated data bus bytes are involved with the present bus cycle as shown in Table 6-1. For both memory and I/O accesses, the byte enable outputs indicate which of the associated data bus bytes are driven valid for write cycles and on which bytes data is expected back for read cycles. Non-contiguous byte enable patterns will never occur.

Address bits A2-A0 of the physical address can be decoded from the byte enables according to Table 6-2. The byte enables can also be decoded to generate BLE# (byte low enable) and BHE# (byte high enable) to address 16-bit memory systems (see Table 6-3 and Table 6-4).

Table 6-1. Pentium® Processor Byte Enables and Associated Data Bytes

Byte Enable Signal	Associated Data Bus Signals
BE0#	D0-D7 (byte 0 — least significant)
BE1#	D8-D15 (byte 1)
BE2#	D16-D23 (byte 2)
BE3#	D24-D31 (byte 3)
BE4#	D32-D39 (byte 4)
BE5#	D40-D47 (byte 5)
BE6#	D48-D55 (byte 6)
BE7#	D56-D63 (byte 7 — most significant)

Address bits A2-A0 of the physical address can be decoded from the byte enables according to Table 6-2. The byte enables can also be decoded to generate BLE# (byte low enable) and BHE# (byte high enable) to address 16-bit memory systems (see Table 6-3 and Table 6-4).

Table 6-2. Generating A2-A0 from BE7-0#

A2	A1	A0	BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#
0	0	0	X	X	X	X	X	X	X	Low
0	0	1	X	X	X	X	X	X	Low	High
0	1	0	X	X	X	X	X	Low	High	High
0	1	1	X	X	X	X	Low	High	High	High
1	0	0	X	X	X	Low	High	High	High	High
1	0	1	X	X	Low	High	High	High	High	High
1	1	0	X	Low	High	High	High	High	High	High
1	1	1	Low	High	High	High	High	High	High	High

Table 6-3. When BLE# is Active

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	BLE#
X	X	X	X	X	X	X	Low	Low
X	X	X	X	X	Low	High	High	Low
X	X	X	Low	High	High	High	High	Low
X	Low	High	High	High	High	High	High	Low

Table 6-4. When BHE# is Active

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	BHE#
X	X	X	X	X	X	Low	X	Low
X	X	X	X	Low	X	High	High	Low
X	X	Low	X	High	High	High	High	Low
Low	X	High	High	High	High	High	High	Low

Table 6-5. When BE3'# is Active

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	BE3'#
Low	X	X	X	Low	X	X	X	Low

Table 6-6. When BE2'# is Active

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	BE2'#
X	Low	X	X	X	Low	X	X	Low

Table 6-7. When BE1'# is Active

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	BE1'#
X	X	Low	X	X	X	Low	X	Low

Table 6-8. When BE0'# is Active

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	BE0'#
X	X	X	Low	X	X	X	Low	Low

6.2.1. Interfacing With 8-, 16-, 32-, and 64-Bit Memories

In 64-bit physical memories such as Figure 6-3, each 8-byte quadword begins at a byte address that is a multiple of eight. A31-A3 are used as an 8-byte quadword select and BE7#-BE0# select individual bytes within the word.

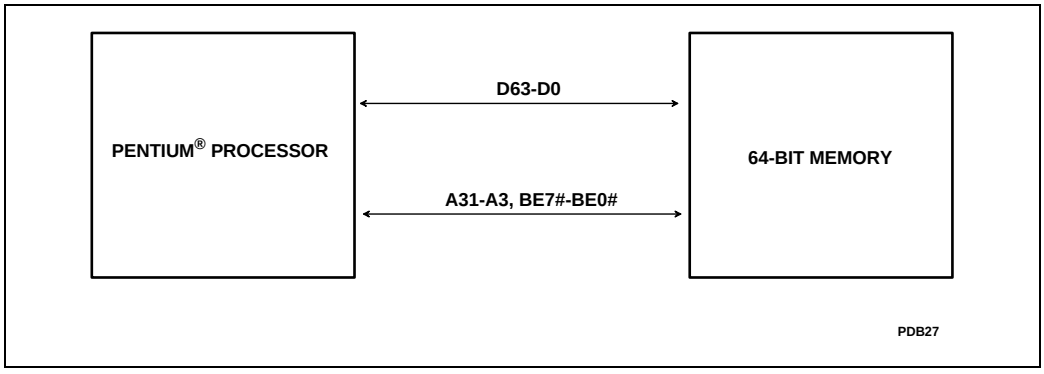


Figure 6-3. Pentium® Processor with 64-Bit Memory

Memories that are 32 bits wide require external logic for generating A2 and BE3'#-BE0'#. Memories that are 16 bits wide require external logic for generating A2, A1, BHE# and BLE#. Memories that are 8 bits wide require external logic for generating A2, A1, and A0. All memory systems that are less than 64 bits wide require external byte swapping logic for routing data to the appropriate data lines.

The Pentium processor expects all the data requested by the byte enables to be returned as one transfer (with one BRDY#), so byte assembly logic is required to return all requested bytes to the Pentium processor at one time. Note that the Pentium processor does not support BS8#, BS16# or BS32#, so this logic must be implemented externally if necessary.

Figure 6-4 shows the Pentium processor address bus interface to 64, 32, 16 and 8-bit memories. Address bits A2, A1, and A0 and BHE#, BLE#, and BE3'#-BE0'# are decoded as shown in Table 6-2 through Table 6-8.

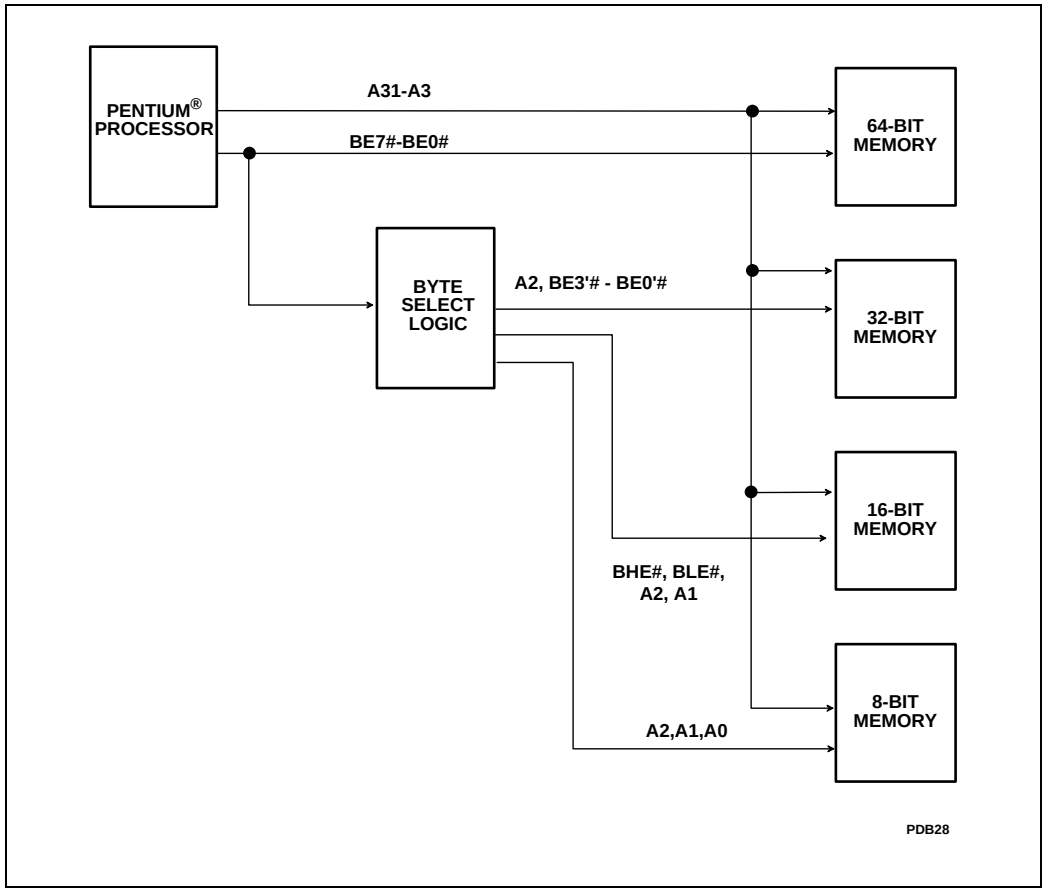


Figure 6-4. Addressing 32-, 16- and 8-Bit Memories

Figure 6-5 shows the Pentium processor data bus interface to 32-, 16- and 8-bit wide memories. External byte swapping logic is needed on the data lines so that data is supplied to and received from the Pentium processor on the correct data pins (see Table 6-1). For memory widths smaller than 64 bits, byte assembly logic is needed to return all bytes of data requested by the Pentium processor in one cycle.

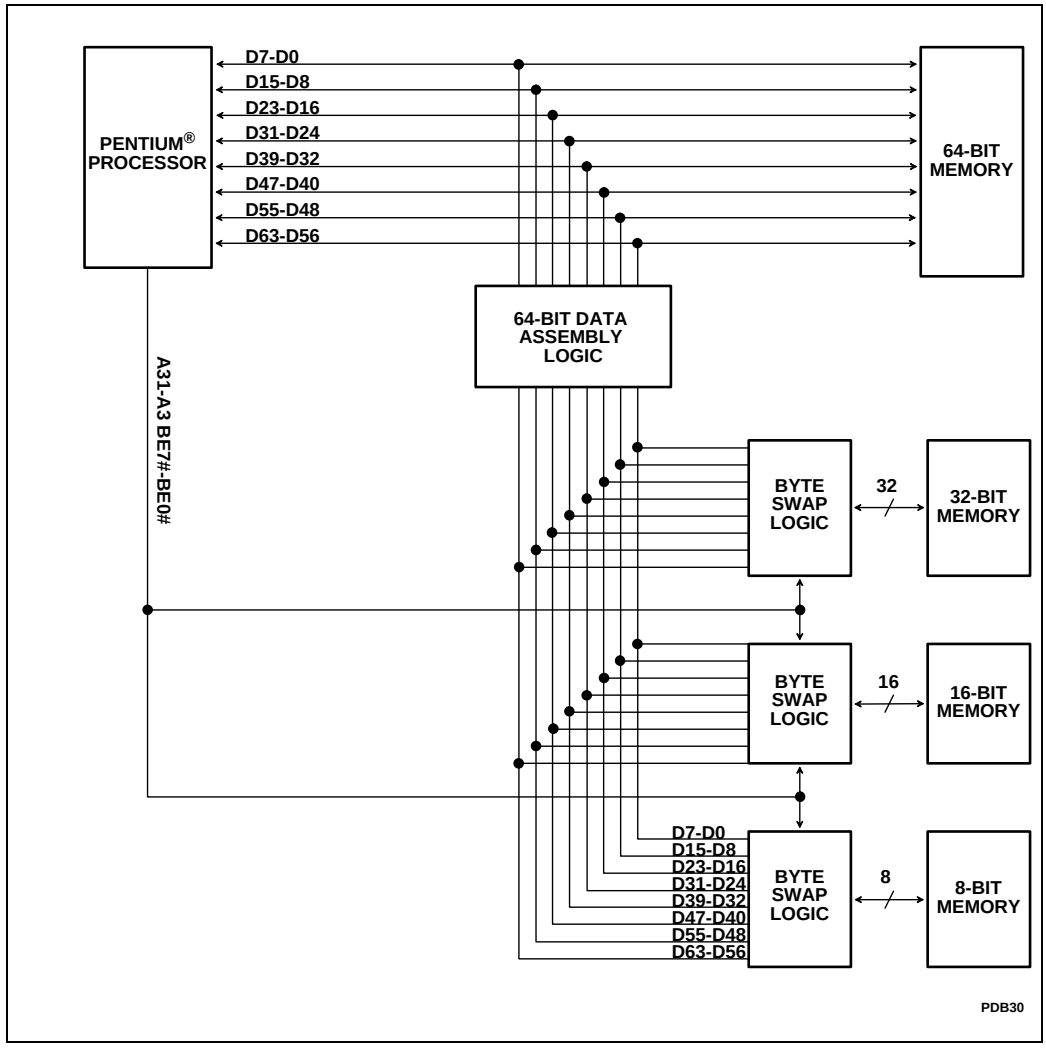


Figure 6-5. Data Bus Interface to 32-, 16- and 8-Bit Memories

Operand alignment and size dictate when two cycles are required for a data transfer. Table 6-9 shows the transfer cycles generated by the Pentium processor for all combinations of logical operand lengths and alignment and applies to both locked and unlocked transfers. When multiple cycles are required to transfer a multi-byte logical operand, the highest order bytes are transferred first.

Table 6-9. Transfer Bus Cycles for Bytes, Words, Dwords and Quadwords

Length of Transfer	**1 Byte**	**2 Bytes**							
Low Order Address	**xxx**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
1st transfer	b	w	w	w	hb	w	w	w	hb
Byte enables driven	0	BE0-1#	BE1-2#	BE2-3#	BE4#	BE4-5#	BE5-6#	BE6-7#	BE0#
Value driven on A3		0	0	0	0	0	0	0	1
2nd transfer (if needed)					lb				lb
Byte enables driven					BE3#				BE7#
Value driven on A3					0				0

Length of Transfer	**4 Bytes**							
Low Order Address	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
1st transfer	d	hb	hw	h3	d	hb	hw	h3
Byte enables driven	BE0-3#	BE4#	BE4-5#	BE4-6#	BE4-7#	BE0#	BE0-1#	BE0-2#
Low order address	0	0	0	0	0	1	1	1
2nd transfer (if needed)		l3	lw	lb		l3	lw	lb
Byte enables driven		BE1-3#	BE2-3#	BE3#		BE5-7#	BE6-7#	BE7#
Value driven on A3		0	0	0		0	0	0

Length of Transfer	**8 Bytes**							
Low Order Address	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
1st transfer	q	hb	hw	h3	hd	h5	h6	h7
Byte enables driven	BE0-7#	BE0#	BE0-1#	BE0-2#	BE0-3#	BE0-4#	BE0-5#	BE0-6#
Value driven on A3	0	1	1	1	1	1	1	1
2nd transfer (if needed)		l7	l6	l5	ld	l3	lw	lb
Byte enables driven		BE1-7#	BE2-7#	BE3-7#	BE4-7#	BE5-7#	BE6-7#	BE7#
Value driven on A3		0	0	0	0	0	0	0

Key:

b = byte transfer w = 2-byte transfer 3 = 3-byte transfer d = 4-byte transfer

5 = 5-byte transfer 6 = 6-byte transfer 7 = 7-byte transfer q = 8-byte transfer

h = high order l = low order

8-byte operand:

high order byte	byte 7	byte 6	byte 5	byte 4	byte 3	byte 2	low order byte

↑ byte with highest address

↑ byte with lowest address

6.3. BUS STATE DEFINITION

This section describes the Pentium processor bus states in detail. See Figure 6-6 for the bus state diagram.

Ti: This is the bus idle state. In this state, no bus cycles are being run. The Pentium processor may or may not be driving the address and status pins, depending on the state of the HLDA, AHOLD, and BOFF# inputs. An asserted BOFF# or RESET will always force the state machine back to this state. HLDA will only be driven in this state.

T1: This is the first clock of a bus cycle. Valid address and status are driven out and ADS# is asserted. There is one outstanding bus cycle.

T2: This is the second and subsequent clock of the first outstanding bus cycle. In state T2, data is driven out (if the cycle is a write), or data is expected (if the cycle is a read), and the BRDY# pin is sampled. There is one outstanding bus cycle.

T12: This state indicates there are two outstanding bus cycles, and that the Pentium processor is starting the second bus cycle at the same time that data is being transferred for the first. In T12, the Pentium processor drives the address and status and asserts ADS# for the second outstanding bus cycle, while data is transferred and BRDY# is sampled for the first outstanding cycle.

T2P: This state indicates there are two outstanding bus cycles, and that both are in their second and subsequent clocks. In T2P, data is being transferred and BRDY# is sampled for the first outstanding cycle. The address, status and ADS# for the second outstanding cycle were driven sometime in the past (in state T12).

TD: This state indicates there is one outstanding bus cycle, that its address, status and ADS# have already been driven sometime in the past (in state T12), and that the data and BRDY# pins are not being sampled because the data bus requires one dead clock to turn around between consecutive reads and writes, or writes and reads. The Pentium processor enters TD if in the previous clock there were two outstanding cycles, the last BRDY# was returned, and a dead clock is needed. The timing diagrams in the next section give examples when a dead clock is needed.

Table 6-10 gives a brief summary of bus activity during each bus state. Figure 6-6 shows the Pentium processor bus state diagram.

Table 6-10. Pentium® Processor Bus Activity

Bus State	Cycles Outstanding	ADS# Asserted New Address Driven	BRDY# Sampled Data Transferred
Ti	0	No	No
T1	1	Yes	No
T2	1	No	Yes
T12	2	Yes	Yes
T2P	2	No	Yes
TD	1	No	No

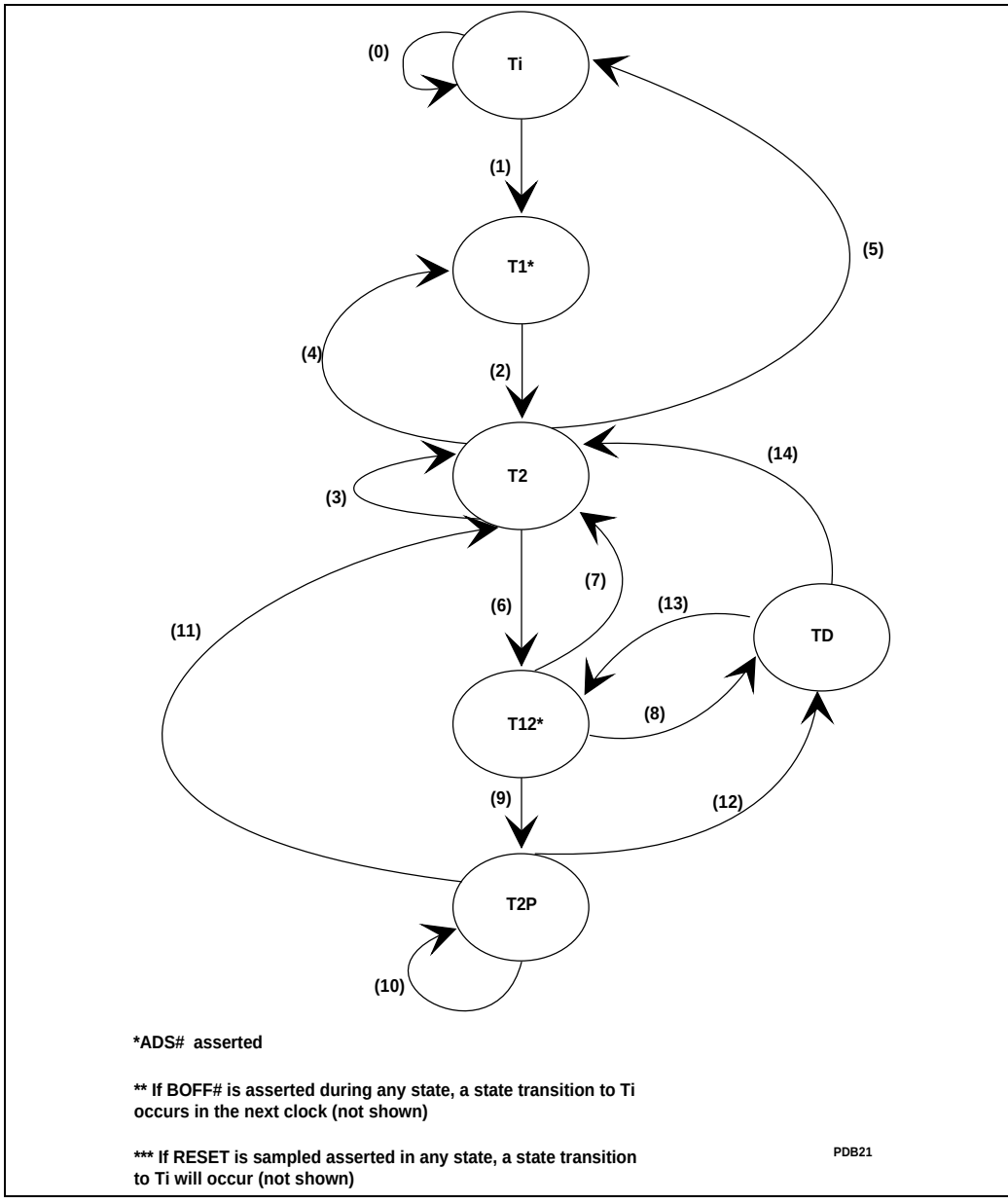


Figure 6-6. Pentium® Processor Bus Control State Machine

6.3.1. State Transitions

The state transition equations with descriptions are listed below. In the equations, "&" means logical AND, "+" means logical OR, and "#" placed after label means active low. The NA# used here is actually a delayed version of the external NA# pin (delayed by one clock). The definition of request pending is:

> The Pentium processor has generated a new bus cycle internally & HOLD (delayed by one clock) negated & BOFF# negated & (AHOLD negated + HITM# asserted).

Note that once NA# is sampled asserted the Pentium processor latches NA# and will pipeline a cycle when one becomes pending even if NA# is subsequently deasserted.

(0) No Request Pending

(1) Request Pending;:

> The Pentium processor starts a new bus cycle & ADS# is asserted in the T1 state.

(2) Always:

> With BOFF# negated, and a cycle outstanding the Pentium processor always moves to T2 to process the data transfer.

(3) Not Last BRDY# & (No Request Pending + NA# Negated):

> The Pentium processor stays in T2 until the transfer is over if no new request becomes pending or if NA# is not asserted.

(4) Last BRDY# & Request Pending & NA# Sampled Asserted:

> If there is a new request pending when the current cycle is complete, and if NA# was sampled asserted, the Pentium processor begins from T1.

(5) Last BRDY# & (No Request Pending + NA# Negated):

> If no cycle is pending when the Pentium processor finishes the current cycle or NA# is not asserted, the Pentium processor goes back to the idle state.

(6) Not Last BRDY# & Request Pending & NA# Sampled Asserted:

> While the Pentium processor is processing the current cycle (one outstanding cycle), if another cycle becomes pending and NA# is asserted, the Pentium processor moves to T12 indicating that the Pentium processor now has two outstanding cycles. ADS# is asserted for the second cycle.

(7) Last BRDY# & No dead clock:

> When the Pentium processor finishes the current cycle, and no dead clock is needed, it goes to the T2 state.

(8) Last BRDY# & Need a dead clock:

> When the Pentium processor finishes the current cycle and a dead clock is needed, it goes to the TD state.

(9) Not Last BRDY#:

With BOFF# negated, and the current cycle not complete, the Pentium processor always moves to T2P to process the data transfer.

(10) Not Last BRDY#:

The Pentium processor stays in T2P until the first cycle transfer is over.

(11) Last BRDY# & No dead clock:

When the Pentium processor finishes the first cycle and no dead clock is needed, it goes to T2 state.

(12) Last BRDY# & Need a dead clock:

When the first cycle is complete, and a dead clock is needed, it goes to TD state.

(13) Request Pending & NA# sampled asserted:

If NA# was sampled asserted and there is a new request pending, it goes to T12 state.

(14) No Request Pending + NA# Negated:

If there is no new request pending, or NA# was not asserted, it goes to T2 state.

6.4. BUS CYCLES

The following terminology is used in this document to describe the Pentium processor bus functions. The Pentium processor requests data transfer cycles, bus cycles, and bus operations. A *data transfer cycle* is one data item, up to 8 bytes in width, being returned to the Pentium processor or accepted from the Pentium processor with BRDY# asserted. A *bus cycle* begins with the Pentium processor driving an address and status and asserting ADS#, and ends when the last BRDY# is returned. A bus cycle may have 1 or 4 data transfers. A *burst cycle* is a bus cycle with 4 data transfers. A *bus operation* is a sequence of bus cycles to carry out a specific function, such as a locked read-modify-write or an interrupt acknowledge.

The section titled “Bus State Definition” describes each of the bus states, and shows the bus state diagram.

Table 6-11 lists all of the bus cycles that will be generated by the Pentium processor. Note that inquire cycles (initiated by EADS#) may be generated from the system to the Pentium processor.

Table 6-11. Pentium® Processor Initiated Bus Cycles

M/IO#	D/C#	W/R#	CACHE#*	KEN#	Cycle Description	# of Transfers
0	0	0	1	x	Interrupt Acknowledge (2 locked cycles)	1 transfer each cycle
0	0	1	1	x	Special Cycle (Table 6-13)	1
0	1	0	1	x	I/O Read, 32-bits or less, non-cacheable	1
0	1	1	1	x	I/O Write, 32-bits or less, non-cacheable	1
1	0	0	1	x	Code Read, 64-bits, non-cacheable	1
1	0	0	x	1	Code Read, 64-bits, non-cacheable	1
1	0	0	0	0	Code Read, 256-bit burst line fill	4
1	0	1	x	x	Intel Reserved (will not be driven by the Pentium processor)	n/a
1	1	0	1	x	Memory Read, 64 bits or less, non-cacheable	1
1	1	0	x	1	Memory Read, 64 bits or less, non-cacheable	1
1	1	0	0	0	Memory Read, 256-bit burst line fill	4
1	1	1	1	x	Memory Write, 64 bits or less, non-cacheable	1
1	1	1	0	x	256-bit Burst Writeback	4

* CACHE# will not be asserted for any cycle in which M/IO# is driven low or for any cycle in which PCD is driven high.

Note that all burst reads are cacheable, and all cacheable read cycles are bursted. There are no non-cacheable burst reads or non-burst cacheable reads.

The remainder of this chapter describes all of the above bus cycles in detail. In addition, locked operations and bus cycle pipelining will be discussed.

6.4.1. Single-Transfer Cycle

The Pentium processor supports a number of different types of bus cycles. The simplest type of bus cycle is a single-transfer non-cacheable 64-bit cycle, either with or without wait states. Non-pipelined read and write cycles with 0 wait states are shown in Figure 6-7.

The Pentium processor initiates a cycle by asserting the address status signal (ADS#) in the first clock. The clock in which ADS# is asserted is by definition the first clock in the bus cycle. The ADS# output indicates that a valid bus cycle definition and address is available on the cycle definition pins and the address bus. The CACHE# output is deasserted (high) to indicate that the cycle will be a single transfer cycle.

For a zero wait state transfer, BRDY# is returned by the external system in the second clock of the bus cycle. BRDY# indicates that the external system has presented valid data on the data pins in response to a read or the external system has accepted data in response to a write. The Pentium processor samples the BRDY# input in the second and subsequent clocks of a bus cycle (the T2, T12 and T2P bus states; see the Bus State Definition section of this chapter for more information).

The timing of the data parity input, DP, and the parity check output, PCHK#, is also shown in Figure 6-7. DP is driven by the Pentium processor and returned to the Pentium processor in the same clock as the data. PCHK# is driven two clocks after BRDY# is returned for reads with the results of the parity check.

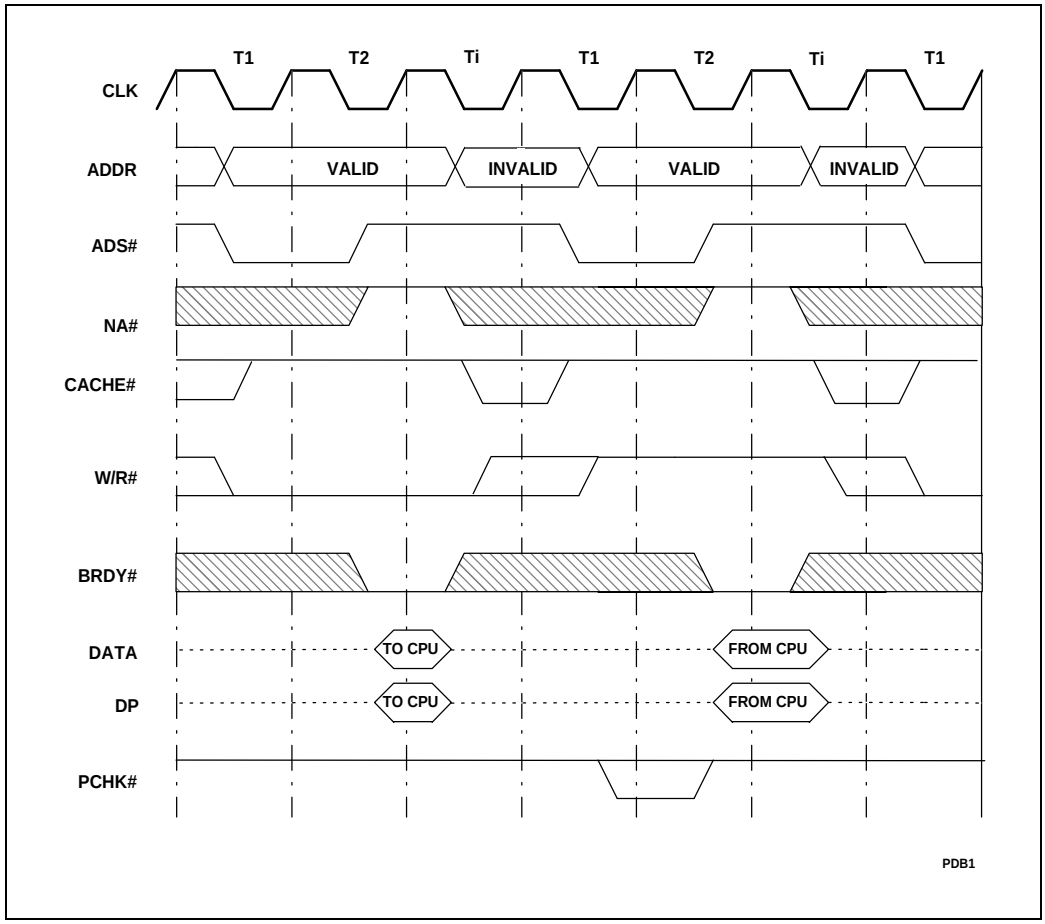


Figure 6-7. Non-Pipelined Read and Write

If the system is not ready to drive or accept data, wait states can be added to these cycles by not returning BRDY# to the processor at the end of the second clock. Cycles of this type, with one and two wait states added are shown in Figure 6-8. Note that BRDY# must be driven inactive at the end of the second clock. Any number of wait states can be added to Pentium processor bus cycles by maintaining BRDY# inactive.

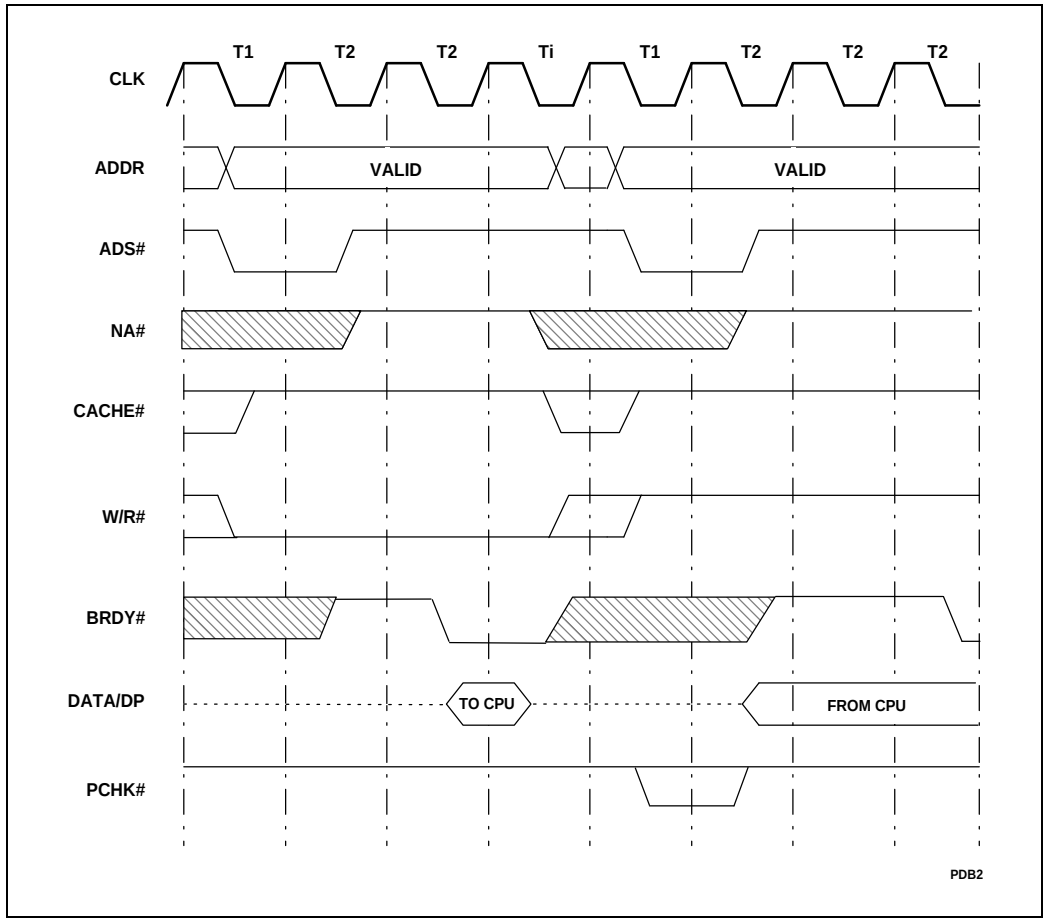


Figure 6-8. Non-Pipelined Read and Write with Wait States

6.4.2. Burst Cycles

For bus cycles that require more than a single data transfer (cacheable cycles and writeback cycles), the Pentium processor uses the burst data transfer. In burst transfers, a new data item can be sampled or driven by the Pentium processor in consecutive clocks. In addition the addresses of the data items in burst cycles all fall within the same 32-byte aligned area (corresponding to an internal Pentium processor cache line).

The implementation of burst cycles is via the BRDY# pin. While running a bus cycle of more than one data transfer, the Pentium processor requires that the memory system perform a burst transfer and follow the burst order (see Table 6-12). Given the first address in the burst sequence, the address of subsequent transfers must be calculated by external hardware. This

requirement exists because the Pentium processor address and byte-enables are asserted for the first transfer and are not re-driven for each transfer. The burst sequence is optimized for two bank memory subsystems and is shown in Table 6-12.

Table 6-12. Pentium® Processor Burst Order

1st Address	2nd Address	3rd Address	4th Address
0	8	10	18
8	0	18	10
10	18	0	8
18	10	8	0

NOTES: The addresses are represented in hexadecimal format.

The cycle length is driven by the Pentium processor together with cycle specification (see Table 6-11), and the system should latch this information and terminate the cycle on time with the appropriate number of transfers. The fastest burst cycle possible requires 2 clocks for the first data item to be returned/driven with subsequent data items returned/driven every clock.

6.4.2.1. BURST READ CYCLES

When initiating any read, the Pentium processor will present the address and byte enables for the data item requested. When the cycle is converted into a cache linefill, the first data item returned should correspond to the address sent out by the Pentium processor; however, the byte enables should be ignored, and valid data must be returned on all 64 data lines. In addition, the address of the subsequent transfers in the burst sequence must be calculated by external hardware since the address and byte enables are not re-driven for each transfer.

Figure 6-9 shows a cacheable burst read cycle. Note that in this case the initial cycle generated by the Pentium processor might have been satisfied by a single data transfer, but was transformed into a multiple-transfer cache fill by KEN# being returned active on the clock that the first BRDY# is returned. In this case KEN# has such an effect because the cycle is internally cacheable in the Pentium processor (CACHE# pin is driven active). KEN# is only sampled once during a cycle to determine cacheability.

PCHK# is driven with the parity check status two clocks after each BRDY#.

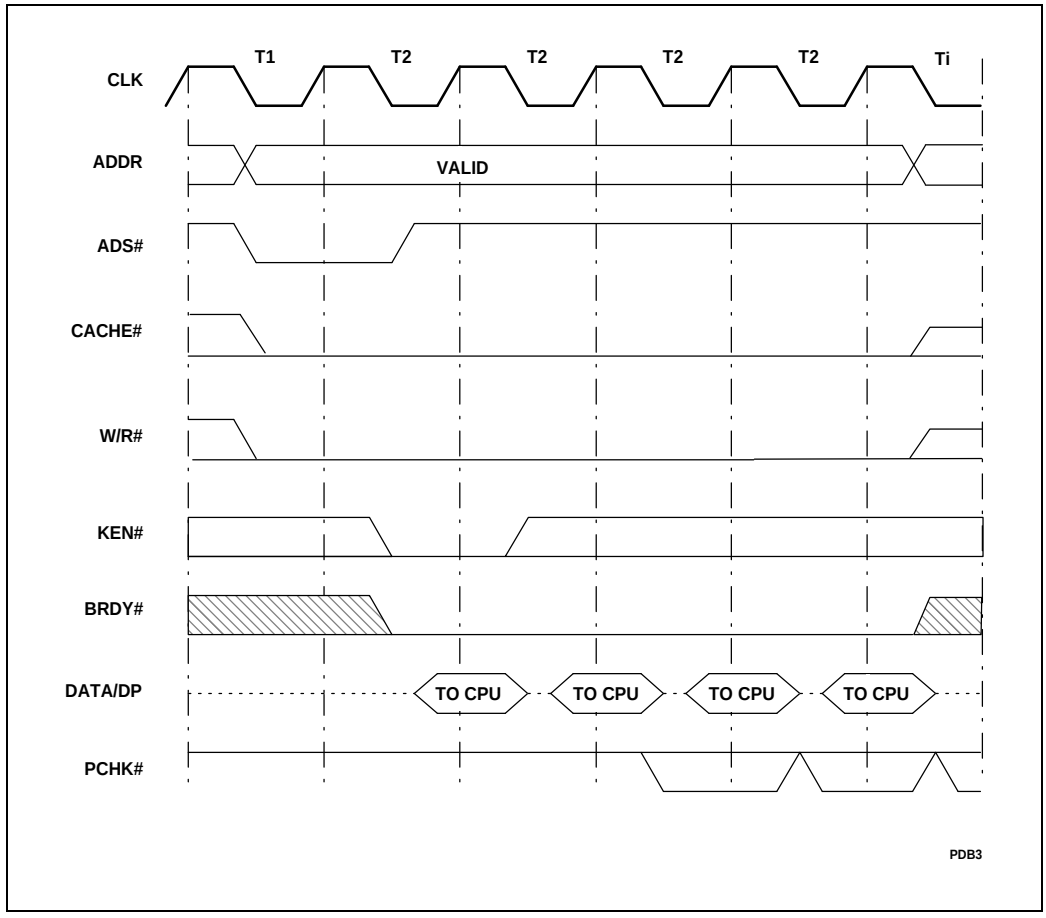


Figure 6-9. Basic Burst Read Cycle

Data will be sampled only in the clock that BRDY# is returned, which means that data need not be sent to Pentium processor every clock in the burst cycle. An example burst cycle where two clocks are required for every burst item is shown in Figure 6-10.

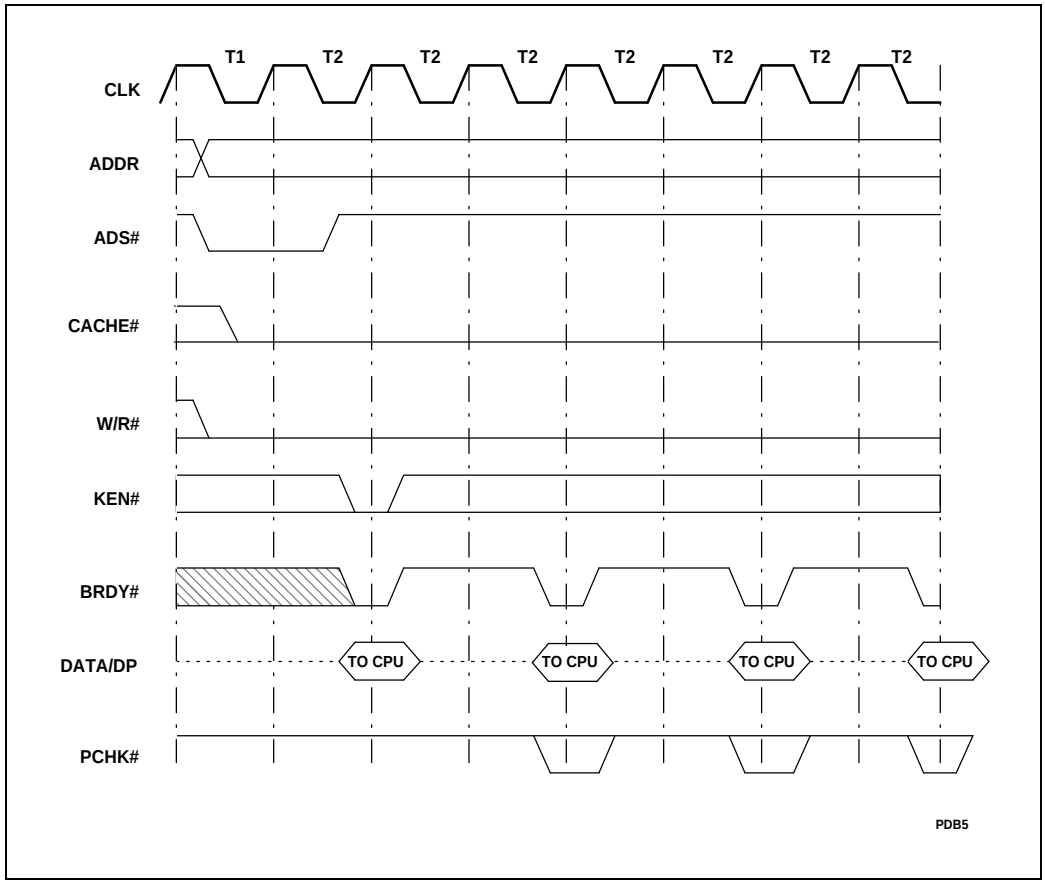


Figure 6-10. Slow Burst Read Cycle

6.4.2.2. BURST WRITE CYCLES

Figure 6-11 shows the timing diagram of basic burst write cycle. KEN# is ignored in burst write cycle. If the CACHE# pin is active (low) during a write cycle, it indicates that the cycle will be a burst writeback cycle. Burst write cycles are always writebacks of modified lines in the data cache. Writeback cycles have several causes:

1. Writeback due to replacement of a modified line in the data cache.
2. Writeback due to an inquire cycle that hits a modified line in the data cache.
3. Writeback due to an internal snoop that hits a modified line in the data cache.
4. Writebacks caused by asserting the FLUSH# pin.
5. Writebacks caused by executing the WBINVD instruction.

Writeback cycles are described in more detail in the Inquire Cycle section of this chapter.

The only write cycles that are burstable by the Pentium processor are writeback cycles. All other write cycles will be 64 bits or less, single transfer bus cycles.

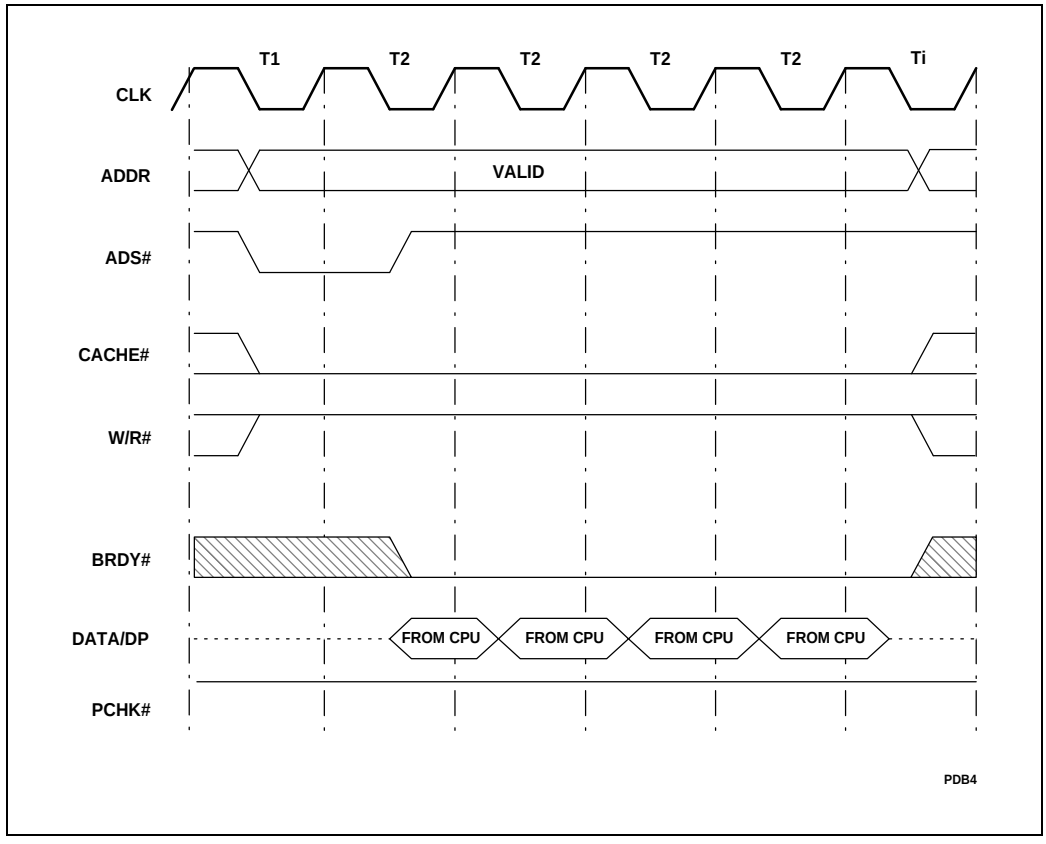


Figure 6-11. Basic Burst Write Cycle

For writeback cycles, the lower five bits of the first burst address always starts at zero; therefore, the burst order becomes 0, 8h, 10h, and 18h. Again, note that the address of the subsequent transfers in the burst sequence must be calculated by external hardware since the Pentium processor does not drive the address and byte enables for each transfer.

6.4.3. Locked Operations

The Pentium processor architecture provides a facility to perform atomic accesses of memory. For example, a programmer can change the contents of a memory-based variable and be assured that the variable was not accessed by another bus master between the read of the variable and the update of that variable. This functionality is provided for select instructions

using a LOCK prefix, and also for instructions which implicitly perform locked read modify write cycles such as the XCHG (exchange) instruction when one of its operands is memory based. Locked cycles are also generated when a segment descriptor or page table entry is updated and during interrupt acknowledge cycles.

In hardware, the LOCK functionality is implemented through the LOCK# pin, which indicates to the outside world that the Pentium processor is performing a read-modify-write sequence of cycles, and that the Pentium processor should be allowed atomic access for the location that was accessed with the first locked cycle. Locked operations begin with a read cycle and end with a write cycle. Note that the data width read is not necessarily the data width written. For example, for descriptor access bit updates the Pentium processor fetches eight bytes and writes one byte.

A locked operation is a combination of one or multiple read cycles followed by one or multiple write cycles. Programmer generated locked cycles and locked page table/directory accesses are treated differently and are described in the following sections.

6.4.3.1. PROGRAMMER GENERATED LOCKS AND SEGMENT DESCRIPTOR UPDATES

For programmer generated locked operations and for segment descriptor updates, the sequence of events is determined by whether or not the accessed line is in the internal cache and what state that line is in.

6.4.3.1.1. Cached Lines in the Modified (M) State

Before a programmer initiated locked cycle or a segment descriptor update is generated, the Pentium processor first checks if the line is in the Modified (M) state. If it is, the Pentium processor drives an unlocked writeback first, leaving the line in the Invalid (I) state, and then runs the locked read on the external bus. Since the operand may be misaligned, it is possible that the Pentium processor may do two writeback cycles before starting the first locked read. In the misaligned scenario the sequence of bus cycles is: writeback, writeback, locked read, locked read, locked write, then the last locked write. Note that although a total of six cycles are generated, the LOCK# pin is active only during the last four cycles. In addition, the SCYC pin is asserted during the last four cycles to indicate that a misaligned lock cycle is occurring. In the aligned scenario the sequence of cycles is: writeback, locked read, locked write. The LOCK# pin is asserted for the last two cycles (SCYC is not asserted, indicating that the locked cycle is aligned). The cache line is left in the Invalid state after the locked operation.

6.4.3.1.2. Non-Cached (I-State), S-State and E-State Lines

A programmer initiated locked cycle or a segment descriptor update to an I, S, or E -state line is always forced out to the bus and the line is transitioned to the Invalid state. Since the line is not in the M-State, no writeback is necessary. Because the line is transitioned to the Invalid state, the locked write is forced out to the bus also. The cache line is left in the Invalid state after the locked operation.

6.4.3.2. PAGE TABLE/DIRECTORY LOCKED CYCLES

In addition to programmer generated locked operations, the Pentium processor performs locked operations to set the dirty and accessed bits in page tables/page directories. The Pentium processor runs the following sequence of bus cycles to set the dirty/accessed bit.

6.4.3.2.1. Cached Lines in the Modified (M) State

If there is a TLB miss, the Pentium processor issues an (unlocked) read cycle to determine if the dirty or accessed bits need to be set. If the line is modified in the internal data cache, the line is written back to memory (lock not asserted). If the dirty or accessed bits need to be set, the Pentium processor then issues a locked read-modify-write operation. The sequence of bus cycles to set the dirty or accessed bits in a page table/directory when the line is in the M-state is: unlocked read, unlocked writeback, locked read, then locked write. The line is left in the Invalid state after the locked operation. Note that accesses to the page tables/directories will not be misaligned.

6.4.3.2.2. Non-Cached (I-State), S-State and E-State Lines

If the line is in the I, S or E-state, the locked cycle is always forced out to the bus and the line is transitioned to the Invalid state. The sequence of bus cycles for an internally generated locked operation is locked read, locked write. The line is left in the Invalid state. Note that accesses to the page tables/directories will not be misaligned.

6.4.3.3. LOCK# OPERATION DURING AHOLD/HOLD/BOFF#

LOCK# is not deasserted if AHOLD is asserted in the middle of a locked cycle.

LOCK# is floated during bus HOLD, but if HOLD is asserted during a sequence of locked cycles, HLDA will not be asserted until the locked sequence is complete.

LOCK# will float if BOFF# is asserted in the middle of a locked cycle, and is driven low again when the cycle is restarted. If BOFF# is asserted during the read cycle of a locked read-modify write, the locked cycle is redriven from the read when BOFF# is deasserted. If BOFF# is asserted during the write cycle of a locked read-modify-write, only the write cycle is redriven when BOFF# is deasserted. The system is responsible for ensuring that other bus masters do not access the operand being locked if BOFF# is asserted during a LOCKed cycle.

6.4.3.4. INQUIRE CYCLES DURING LOCK#

This section describes the Pentium processor bus cycles that will occur if an inquire cycle is driven while LOCK# is asserted. Note that inquire cycles are only recognized if AHOLD, BOFF# or HLDA is asserted and the external system returns an external snoop address to the Pentium processor. If AHOLD, BOFF# or HLDA is not asserted when EADS# is driven, EADS# is ignored. Note also that an inquire cycle can not hit the “locked line” because the LOCK cycle invalidated it.

Because HOLD is not acknowledged when LOCK# is asserted, inquire cycles run in conjunction with the assertion of HOLD can not be driven until LOCK# is deasserted and HLDA is asserted.

BOFF# takes priority over LOCK#. Inquire cycles are permitted while BOFF# is asserted. If an inquire cycle hits a modified line in the data cache, the writeback due to the snoop hit will be driven before the locked cycle is re-driven. LOCK# will be asserted for the writeback.

An inquire cycle with AHOLD may be run concurrently with a locked cycle. If the inquire cycle hits a modified line in the data cache, the writeback may be driven between the locked read and the locked write. If the writeback is driven between the locked read and write, LOCK# will be asserted for the writeback.

NOTE

Only writebacks due to an external snoop hit to a modified line may be driven between the locked read and the locked write of a LOCKed sequence. No other writebacks (due to an internal snoop hit or data cache replacement) are allowed to invade a LOCKed sequence.

6.4.3.5. LOCK# TIMING AND LATENCY

The timing of LOCK# is shown in Figure 6-12. Note that LOCK# is asserted with the ADS# of the read cycle and remains active until the BRDY# of the write cycle is returned. Figure 6-13 shows an example of two consecutive locked operations. Note that the Pentium processor automatically inserts at least one idle clock between two *consecutive* locked operations to allow the LOCK# pin to be sampled inactive by external hardware. Figure 6-14 shows an example of a misaligned locked operation with SCYC asserted.

The maximum number of Pentium processor initiated cycles that will be locked together is four. Four cycles are locked together when data is misaligned for programmer generated locks (read, read, write, write). SCYC will be asserted for misaligned locked cycles. Note that accesses to the page tables/directories will not be misaligned.

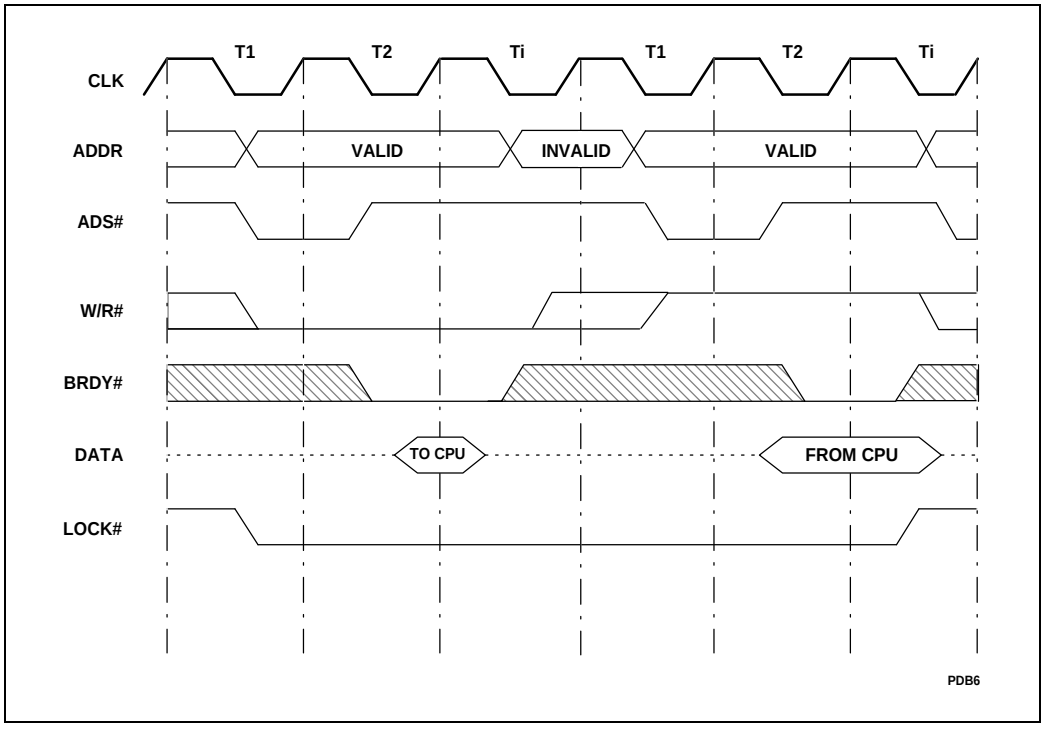


Figure 6-12. LOCK# Timing

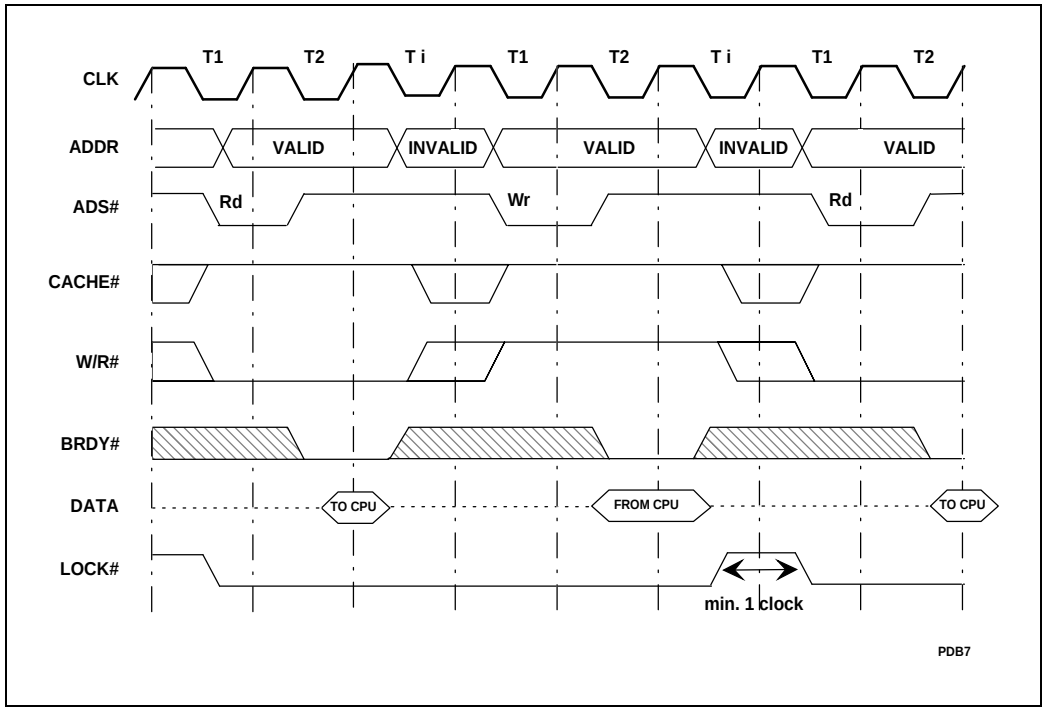


Figure 6-13. Two Consecutive Locked Operations

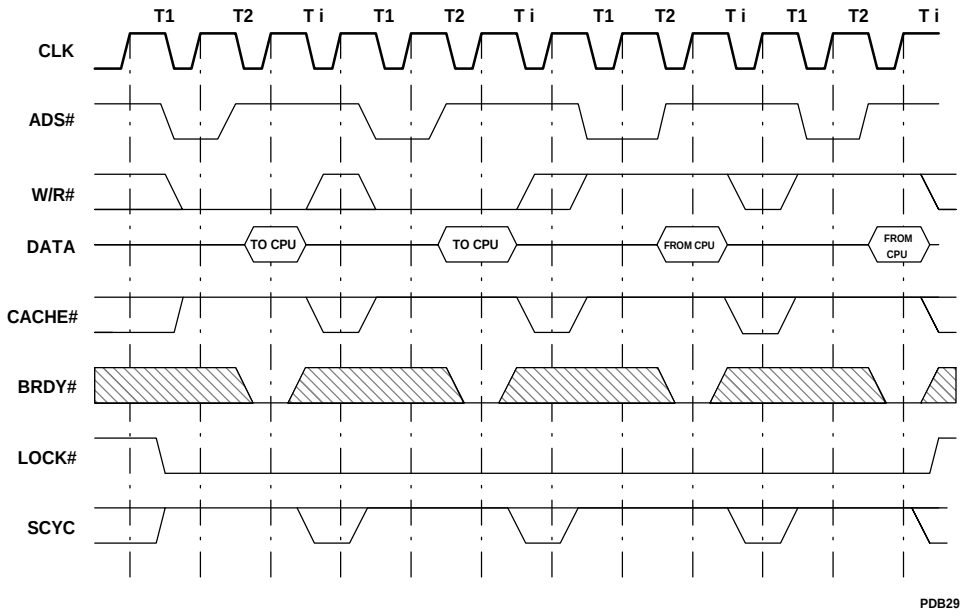


Figure 6-14. Misaligned Locked Cycles

6.4.4. BOFF#

In a multi-master system, another bus master may require the use of the bus to enable the Pentium processor to complete its current cycle. The BOFF# pin is provided to prevent this deadlock situation. If BOFF# is asserted, the Pentium processor will immediately (in the next clock) float the bus (see Figure 6-15). Any bus cycles in progress are aborted and any data returned to the processor in the clock BOFF# is asserted is ignored. In response to BOFF#, the Pentium processor floats the same pins as HOLD, but HLDA is not asserted. BOFF# overrides BRDY#, so if both are sampled active in the same clock, BRDY# is ignored. The Pentium processor samples the BOFF# pin every clock.

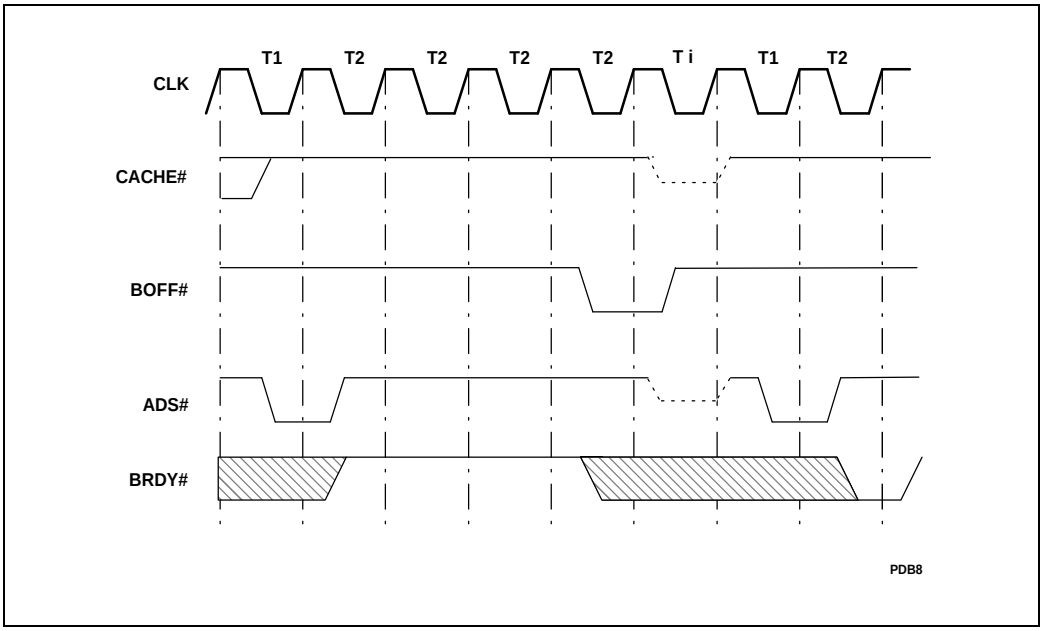


Figure 6-15. Back Off Timing

The device that asserts BOFF# to the Pentium processor is free to run any bus cycle while the Pentium processor is in the high impedance state. If BOFF# is asserted after the Pentium processor has started a cycle, the new master should wait for memory to return BRDY# before driving a cycle. Waiting for BRDY# provides a handshake to insure that the memory system is ready to accept a new cycle. If the bus is idle when BOFF# is asserted, the new master can start its cycle two clocks after issuing BOFF#. The system must wait two clocks after the assertion of BOFF# to begin its cycle to prevent address bus contention.

The bus remains in the high impedance state until BOFF# is negated. At that time, the Pentium processor restarts all aborted bus cycles from the beginning by driving out the address and status and asserting ADS#. Any data returned before BOFF# was asserted is used to continue internal execution, however that data is not placed in an internal cache. Any aborted bus cycles will be restarted from the beginning.

External hardware should assure that if the cycle attribute KEN# was returned to the processor (with the first BRDY# or NA#) before the cycle was aborted, it must be returned with the same value after the cycle is restarted. In other words, backoff cannot be used to change the cacheability property of the cycle. The WB/WT# attribute may be changed when the cycle is restarted.

If more than one cycle is outstanding when BOFF# is asserted, the Pentium processor will restart both outstanding cycles in their original order. The cycles will not be pipelined unless NA# is asserted appropriately.

A pending writeback cycle due to an external snoop hit will be reordered in front of any cycles aborted due to BOFF#. For example, if a snoop cycle is run concurrently with a line fill, and the snoop hits an M-state line and then BOFF# is asserted, the writeback cycle due to the snoop will be driven from the Pentium processor before the cache linefill cycle is restarted.

The system must not rely on the original cycle, that was aborted due to BOFF#, from restarting immediately after BOFF# is deasserted. In addition to reordering writebacks due to external snoop hit in front of cycles that encounter a BOFF#, the processor may also reorder bus cycles in the following situations:

1. A pending writeback cycle due to an internal snoop hit will be reordered in front of any cycles aborted due to BOFF#. If a read cycle is running on the bus, and an internal snoop of that read cycle hits a modified line in the data cache, and the system asserts BOFF#, the Pentium processor will drive out a writeback cycle resulting from the internal snoop hit. After completion of the writeback cycle, the processor will then restart the original read cycle. This circumstance can occur during accesses to the page tables/directories, and during prefetch cycles, since these accesses cause a bus cycle to be generated before the internal snoop to the data cache is performed.
2. If BOFF# is asserted during a data cache replacement writeback cycle, the writeback cycle will be aborted and then restarted once BOFF# is deasserted. However, if the processor encounters a request to access the page table/directory in memory during the BOFF#, this request will be reordered in front of the replacement writeback cycle that was aborted due to BOFF#. The Pentium processor will first run the sequence of bus cycles to service the page table/directory access and then restart the original replacement writeback cycle.

Asserting BOFF# in the same clock as ADS# may cause the Pentium processor to leave the ADS# signal floating low. Since ADS# is floating low, a peripheral device may think that a new bus cycle has begun even though the cycle was aborted. There are several ways to approach this situation:

1. Design the system’s state machines/logic such that ADS# is not recognized the clock after ADS# is sampled active.
2. Recognize a cycle as ADS# asserted and BOFF# negated in the previous clock.
3. Assert AHOLD one clock before asserting BOFF#.

6.4.5. Bus Hold

The Pentium processor provides a bus hold, hold acknowledge protocol using the HOLD and HLDA pins. HOLD is used to indicate to the Pentium processor that another bus master wants control of the bus. When the Pentium processor completes all outstanding bus cycles, it will release the bus by floating its external bus, and drive HLDA active. An example HOLD/HLDA transaction is shown in Figure 6-16.

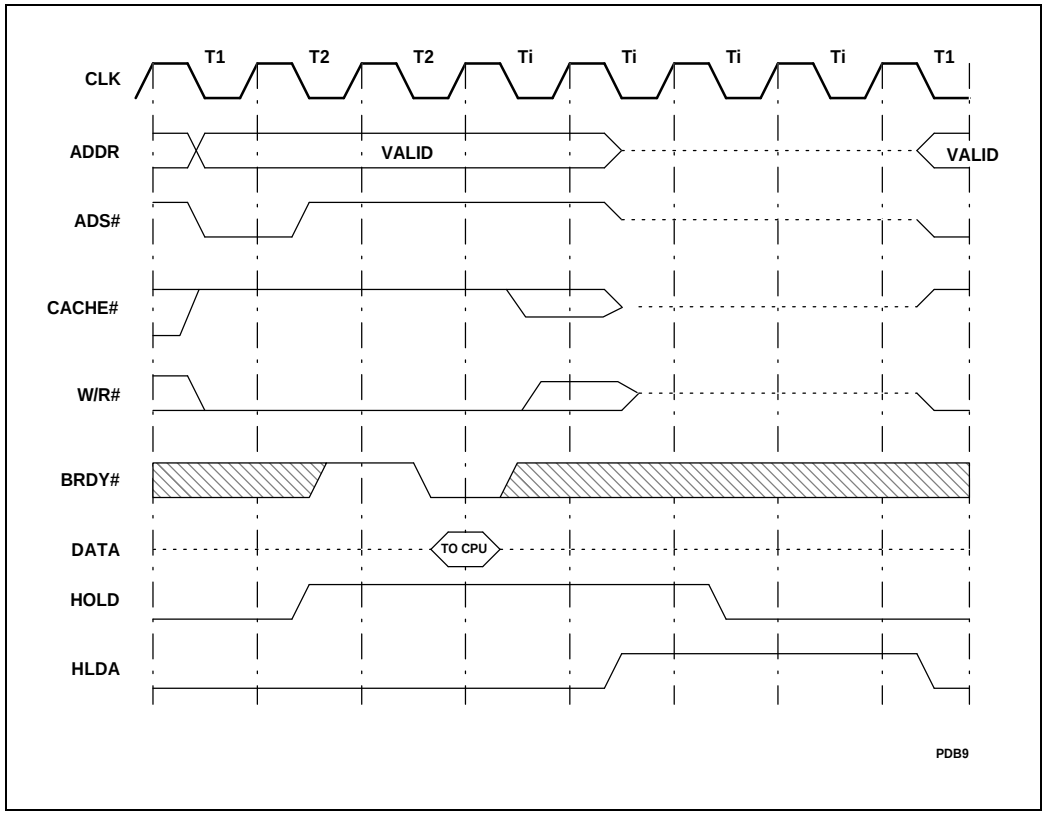


Figure 6-16. HOLD/HLDA Cycles

The Pentium processor recognizes HOLD while RESET is asserted, when BOFF# is asserted, and during BIST (built in self test). HOLD is not recognized when LOCK# is asserted. Once HOLD is recognized, HLDA will be asserted two clocks after the later of the last BRDY# or HOLD assertion. Because of this, it is possible that a cycle may begin after HOLD is asserted, but before HLDA is driven. The maximum number of cycles that will be driven after HOLD is asserted is one. BOFF# may be used if it is necessary to force the Pentium processor to float its bus in the next clock. Figure 6-16 shows the latest HOLD may be asserted relative to ADS# to guarantee that HLDA will be asserted before another cycle is begun.

The operation of HLDA is not affected by the assertion of BOFF#. If HOLD is asserted while BOFF# is asserted, HLDA will be asserted two clocks later. If HOLD goes inactive while BOFF# is asserted, HLDA is deasserted two clocks later.

Note that HOLD may be acknowledged between two bus cycles in a misaligned access.

All outputs are floated when HLDA is asserted except: APCHK#, BREQ, FERR#, HIT#, HITM#, HLDA, IERR#, PCHK#, PRDY, BP3-2, PM1/BP1, PM0/BP0, SMIACT# and TDO.

6.4.6. Interrupt Acknowledge

The Pentium processor generates interrupt acknowledge cycles in response to maskable interrupt requests generated on the interrupt request input (INTR) pin (if interrupts are enabled). Interrupt acknowledge cycles have a unique cycle type generated on the cycle type pins.

An example interrupt acknowledge transaction is shown in Figure 6-17. Interrupt acknowledge cycles are generated in locked pairs. Data returned during the first cycle is ignored, however the specified data setup and hold times must be met. The interrupt vector is returned during the second cycle on the lower 8 bits of the data bus. The Pentium processor has 256 possible interrupt vectors.

The state of address bit 2 (as decoded from the byte enables) distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4: A[31:3] = 0, BE4# = 0, BE[7:5]# = 1, and BE[3:0]# = 1. The address driven during the second interrupt acknowledge cycle is 0: A[31:3] = 0, BE0# = 0 and BE[7:1]# = 1h.

Interrupt acknowledge cycles are terminated when the external system returns BRDY#. Wait states can be added by withholding BRDY#. The Pentium processor automatically generates at least one idle clock between the first and second cycles, however the external system is responsible for interrupt controller (8259A) recovery.

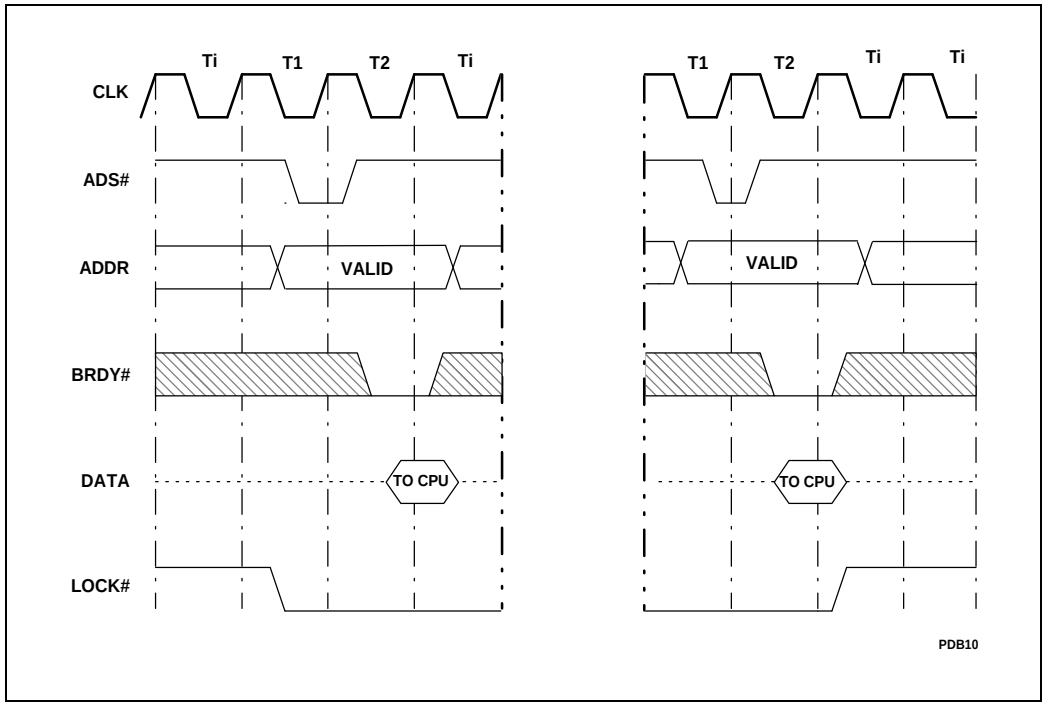


Figure 6-17. Interrupt Acknowledge Cycles

6.4.7. Flush Operations

The FLUSH# input is implemented in the Pentium processor as an asynchronous interrupt, similar to NMI. Therefore, unlike the Intel486 microprocessor, FLUSH# is recognized on instruction boundaries only. FLUSH# is latched internally. Once setup, hold and pulse width times have been met, FLUSH# may be deasserted, even if a bus cycle is in progress.

To execute a flush operation, the Pentium processor first writes back all modified lines to external memory. The lines in the internal caches are invalidated as they are written back. After the write-back and invalidation operations are complete, a special cycle, flush acknowledge, is generated by the Pentium processor to inform the external system.

6.4.8. Special Bus Cycles

The Pentium processor provides six special bus cycles to indicate that certain instructions have been executed, or certain conditions have occurred internally. The special bus cycles in Table 6-13 are defined when the bus cycle definition pins are in the following state: M/IO# = 0, D/C# = 0 and W/R# = 1. During most special cycles the data bus is undefined and

the address lines A31-A3 are driven to "0." The external hardware must acknowledge all special bus cycles by returning BRDY#.

Table 6-13. Special Bus Cycles Encoding

BE7#	BE6#	BE5#	BE4#	BE3#	BE2#	BE1#	BE0#	Special Bus Cycle
1	1	1	1	1	1	1	0	Shutdown
1	1	1	1	1	1	0	1	Flush (INVD,WBINVD instr)
1	1	1	1	1	0	1	1	Halt/Stop Grant[1]
1	1	1	1	0	1	1	1	Writeback (WBINVD instruction)
1	1	1	0	1	1	1	1	Flush Acknowledge (FLUSH# assertion)
1	1	0	1	1	1	1	1	Branch Trace Message

NOTE:

1. The definition of the Stop Grant bus cycle is the same as the HALT cycle definition, with the exception that the address bus is driven with the value 0000 0010H during the Stop Grant bus cycle.

Shutdown can be generated due to the following reasons:

1. If any other exception occurs while the Pentium processor is attempting to invoke the double-fault handler.
2. An internal parity error is detected.

Prior to going into shutdown, the Pentium processor will not writeback the M-state lines. Upon entering shutdown, the state of the CPU is unpredictable and may or may not be recoverable. RESET or INIT should be asserted to return the system to a known state. Although some system operations (i.e. FLUSH# and R/S#) are generally recognized during shutdown, these operations may not complete successfully in some cases once shutdown is entered. During shutdown, the internal caches remain in the same state unless an inquire cycle is run or the cache is flushed.

The Pentium processor will remain in shutdown until NMI, INIT, or RESET is asserted. Furthermore, upon exit from shutdown with NMI (to the NMI handler), the SS, ESP and EIP of the task that was executing when shutdown occurred can no longer be relied upon to be valid. Therefore, using NMI to exit shutdown should be used only for debugging purposes and not to resume execution from where shutdown occurred.

If invoking NMI to exit shutdown, use a task gate rather than an interrupt or trap gate in slot 2 of the IDT. One of the conditions that may lead to shutdown is an attempt to use an invalid stack segment selector (SS). In this case, if the NMI successfully exits shutdown, it will immediately re-enter shutdown because it has no valid stack on which to push the return address. It is more robust to vector NMI through a task gate rather than an interrupt gate in the IDT, since the task descriptor allocates a new stack for the NMI handler context.

The Flush Special Cycle is driven after the INVD (invalidate cache) or WBINVD (writeback invalidate cache) instructions are executed. The Flush Special Cycle is driven to indicate to the external system that the internal caches were invalidated and that external caches should also be invalidated.

NOTE

INVD should be used with care. This instruction does not writeback modified cache lines.

The Halt Special Cycle is driven when a Halt instruction is executed. Externally, halt differs from shutdown in only two ways:

1. In the resulting byte enables that are asserted.
2. The Pentium processor will exit the Halt state if INTR is asserted and maskable interrupts are enabled in addition to the assertion of NMI, INIT or RESET.

A special Stop Grant bus cycle will be driven after the processor recognizes the STPCLK# interrupt. The definition of the Stop Grant bus cycle is the same as the HALT cycle definition, with the exception that the address bus is driven with the value 0000 0010H during the Stop Grant bus cycle.

The Writeback Special Cycle is driven after the WBINVD instruction is executed and it indicates that modified lines in the Pentium processor data cache were written back to memory or a second level cache. The Writeback Special Cycle also indicates that modified lines in external caches should be written back. After the WBINVD instruction is executed, the Writeback Special cycle is generated, followed by the Flush Special Cycle. Note that INTR is not recognized while the WBINVD instruction is being executed.

When the FLUSH# pin is asserted to the Pentium processor, all modified lines in the data cache are written back and all lines in the code and data caches are invalidated. The Flush Acknowledge Special Cycle is driven after the writeback and invalidations are complete. The Flush Acknowledge Special Cycle is driven only in response to the FLUSH# pin being activated. Note that the Flush Acknowledge Special Cycle indicates that all modified lines were written back and all cache lines were invalidated while the Flush special cycle only indicates that all cache lines were invalidated.

The Branch Trace Message Special Cycle is part of the Pentium processor’s execution tracing protocol. The Branch Trace Message Special Cycle is the only special cycle that does not drive 0’s on the address bus, however like the other special cycles, the data bus is undefined. When the branch trace message is driven, bits 31-3 of the branch target linear address are driven on A[31:3].

6.4.9. Bus Error Support

Pentium processor provides basic support for bus error handling through data and address parity check. Even data parity will be generated by the processor for every enabled byte in write cycles and will be checked for all valid bytes in read cycles. The PCHK# output signals if a data parity error is encountered for reads.

Even address parity will be generated for A31-A5 during write and read cycles, and checked during inquire cycles. The APCHK# output signals if an address parity error is encountered during inquire cycles.

External hardware is free to take whatever actions are appropriate after a parity error. For example, external hardware may signal an interrupt if PCHK# or APCHK# is asserted. Please refer to the Error Detection chapter for the details.

6.4.10. Pipelined Cycles

The NA# input indicates to the Pentium processor that it may drive another cycle before the current one is completed. Cacheability (KEN#) and cache policy (WB/WT#) indicators for the current cycle are sampled in the same clock NA# is sampled active (or the first BRDY# for that cycle, whichever comes first). Note that the WB/WT# and KEN# inputs are sampled with the first of BRDY# or NA# even if NA# does not cause a pipelined cycle to be driven because there was no pending cycle internally or two cycles are already outstanding.

The NA# input is latched internally, so even if a cycle is not pending internally in the clock that NA# is sampled active, but becomes pending before the current cycle is complete, the pending cycle will be driven to the bus even if NA# was subsequently deasserted.

LOCK# and writeback cycles are not pipelined into other cycles and other cycles are not pipelined into them (regardless of the state of NA#). Special cycles and I/O cycles may be pipelined.

An example of burst pipelined back to back reads is shown in Figure 6-18. The assertion of NA# causes a pending cycle to be driven 2 clocks later. Note KEN# timing.

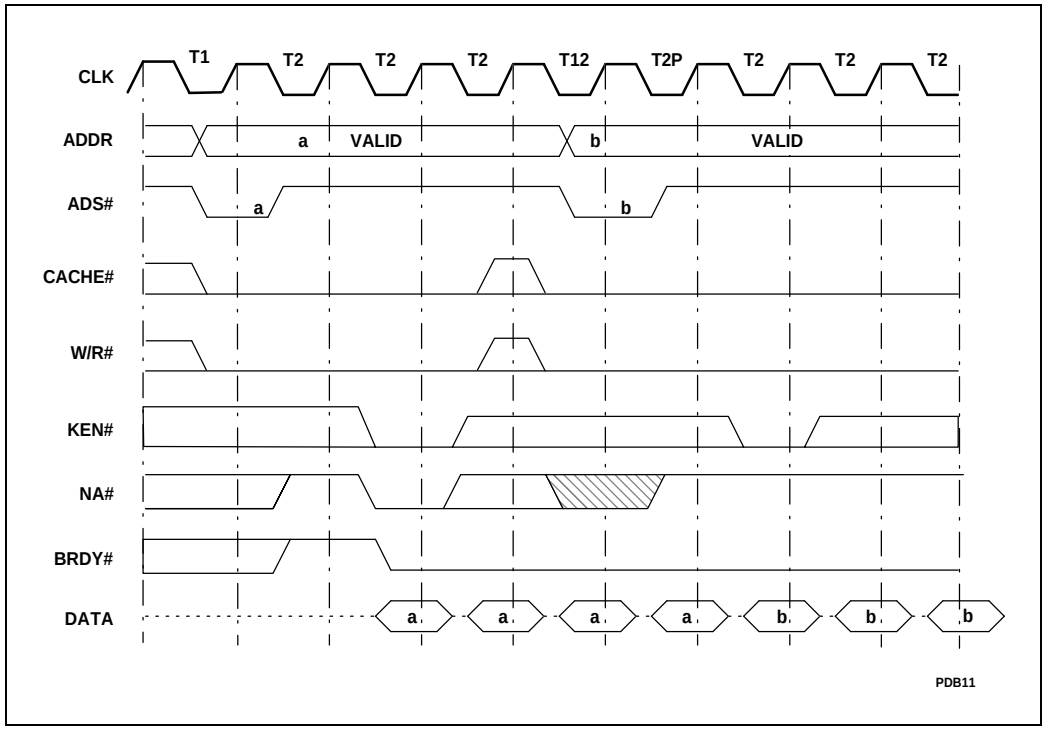


Figure 6-18. Two Pipelined Cache Linefills

Write cycles can be pipelined into read cycles and read cycles can be pipelined into write cycles, but one dead clock will be inserted between read and write cycles to allow bus turnover (see the bus state diagram in the Bus State Definition section of this chapter). Pipelined back-to-back read/write cycles are shown in Figure 6-19.

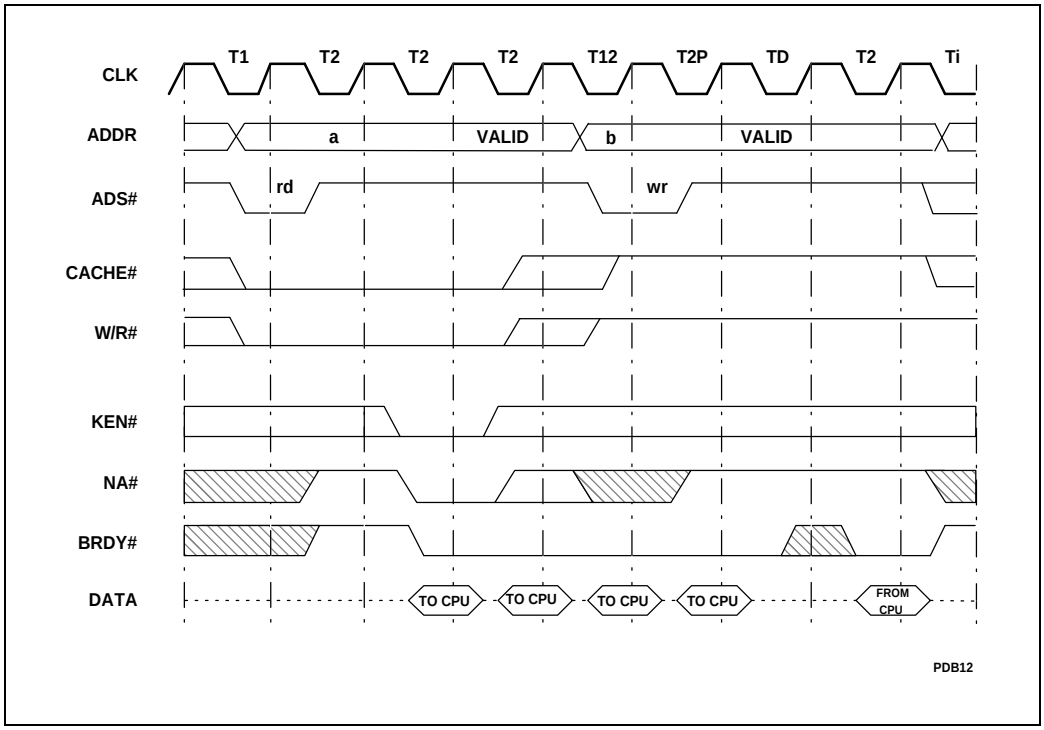


Figure 6-19. Pipelined Back-to-Back Read/Write Cycles

6.4.10.1. KEN# AND WB/WT# SAMPLING FOR PIPELINED CYCLES

KEN# and WB/WT# are sampled with NA# or BRDY# for that cycle, whichever comes first. Figure 6-20 and Figure 6-21 clarify this specification.

Figure 6-20 shows that even though two cycles have been driven, the NA# for the second cycle still causes KEN# and WB/WT# to be sampled for the second cycle. A third ADS# will not be driven until all the BRDY#s for cycle 1 have been returned to the Pentium processor.

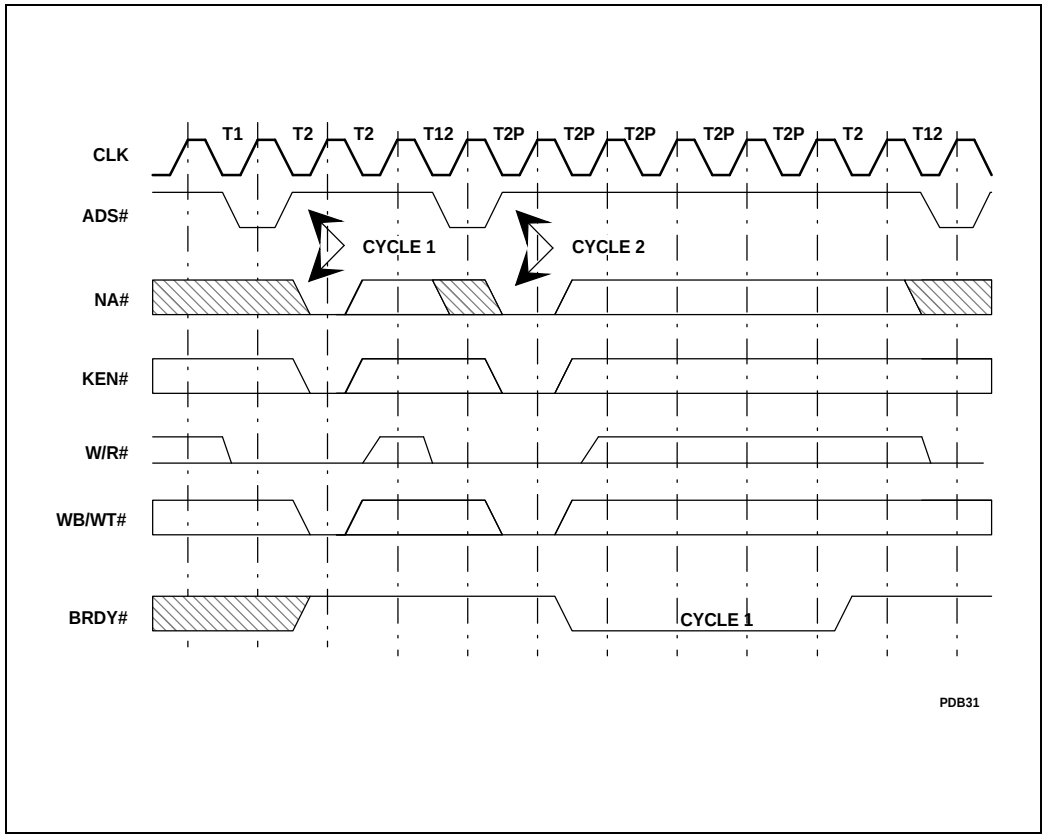


Figure 6-20. KEN# and WB/WT# Sampling with NA#

Figure 6-21 shows that two cycles are outstanding on the Pentium processor bus. The assertion of NA# caused the sampling of KEN# and WB/WT# for the first cycle. The assertion of the four BRDY#s for the first cycle DO NOT cause the KEN# and WB/WT# for the second cycle to be sampled. In this example, KEN# and WB/WT# for the second cycle are sampled with the first BRDY# for the second cycle.

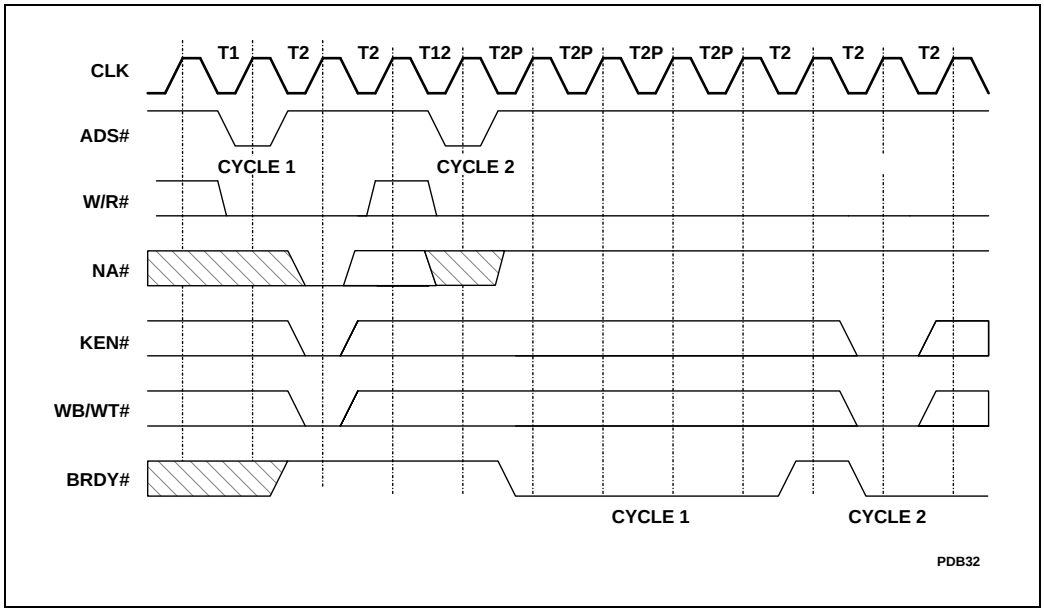


Figure 6-21. KEN# and WB/WT# Sampling with BRDY#

6.4.11. Dead Clock Timing Diagrams

The timing diagrams in Figure 6-22 and Figure 6-23 show bus cycles with and without a dead clock.

In Figure 6-22, cycles 1 and 2 can be either read or write cycles and no dead clock would be needed because only one cycle is outstanding when those cycles are driven. To prevent a dead clock from being necessary after cycle 3 is driven, it must be of the “same type” as cycle 2. That is if cycle 2 is a read cycle, cycle 3 must also be a read cycle in order to prevent a dead clock. If cycle 2 is a write cycle, cycle 3 must also be a write cycle to prevent a dead clock.

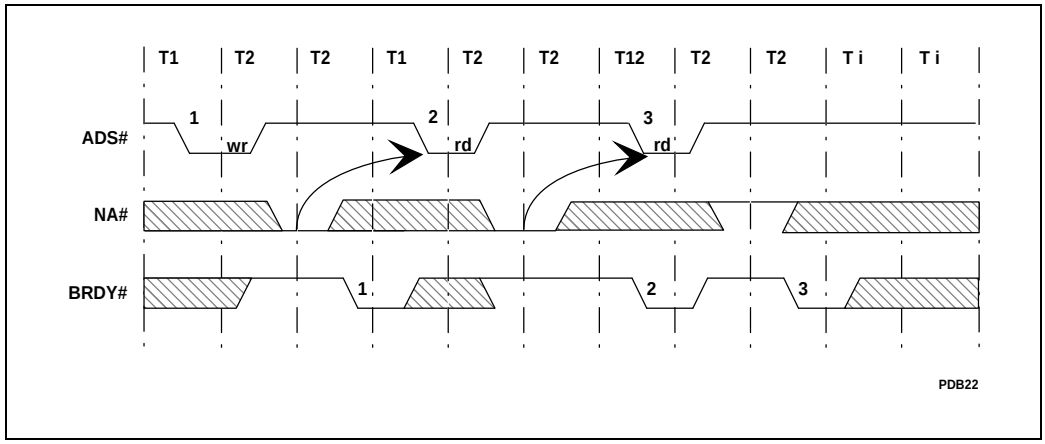


Figure 6-22. Bus Cycles Without Dead Clock

NOTE

Although the processor ignores BRDY# during this dead clock when configured in uni-processor mode, BRDY# may be falsely recognized in an inter-CPU pipelined cycle. As such, dual processing system designs must not drive BRDY# low during this dead clock.

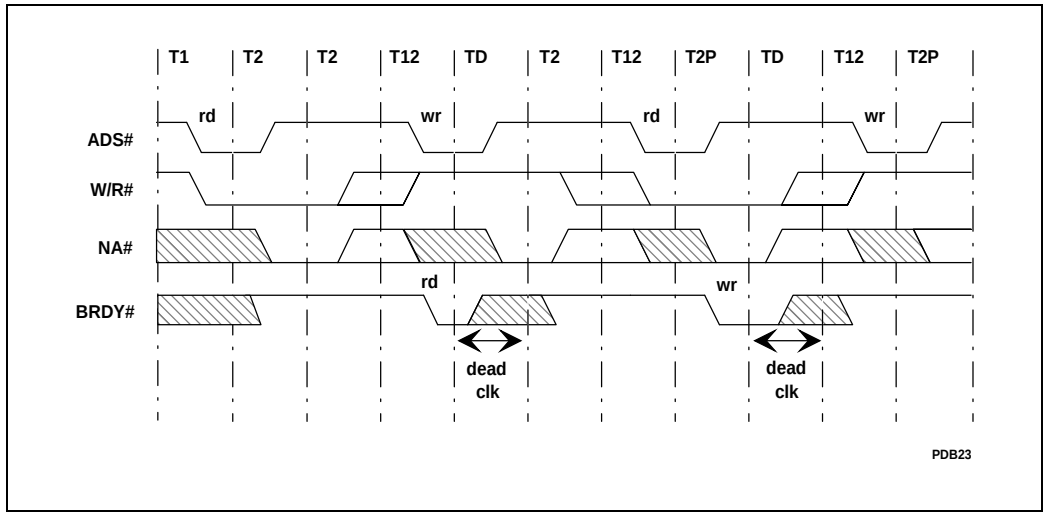


Figure 6-23. Bus Cycles with TD Dead Clock

6.5. CACHE CONSISTENCY CYCLES (INQUIRE CYCLES)

The purpose of an inquire cycle is to check whether a particular address is cached in a Pentium processor internal cache and optionally invalidate it. After an inquire cycle is complete, the system has information on whether or not a particular address location is cached and what state it is in.

An inquire cycle is typically performed by first asserting AHOLD to force the Pentium processor to float its address bus, waiting two clocks, and then driving the inquire address and INV and asserting EADS#. Inquire cycles may also be executed while the Pentium processor is forced off the bus due to HLDA, or BOFF#. Because the entire cache line is affected by an inquire cycle, only A31-A5 need to be driven with the valid inquire address. Although the value of A4-A3 is ignored, these inputs should be driven to a valid logic level during inquire cycles for circuit reasons. The INV pin is driven along with the inquire address to indicate whether the line should be invalidated (INV high) or marked as shared (INV low) in the event of an inquire hit.

After the Pentium processor determines if the inquire cycle hit a line in either internal cache, it drives the HIT# pin. HIT# is asserted (low) two clocks after EADS# is sampled asserted[1] if the inquire cycle hit a line in the code or data cache. HIT# is deasserted (high) two clocks after EADS# is sampled asserted if the inquire cycle missed in both internal caches. The HIT# output changes its value only as a result of an inquire cycle. It retains its value between inquire

[1]Since the EADS# input is ignored by the processor in certain clocks, the two clocks reference is from the clock in which EADS# is asserted and actually sampled by the processor at the end of this clock (i.e. rising edge of next clock) as shown in Figure 6-25.

cycles. In addition, the HITM# pin is asserted two clocks after EADS# if the inquire cycle hit a modified line in the data cache. HITM# is asserted to indicate to the external system that the Pentium processor contains the most current copy of the data and any device needing to read that data should wait for the Pentium processor to write it back. The HITM# output remains asserted until two clocks after the last BRDY# of the writeback cycle is asserted.

The external system must inhibit inquire cycles during BIST (initiated by INIT being sampled high on the falling edge of RESET), and during the Boundary Scan Instruction RUNBIST. When the model specific registers (test registers) are used to read or write lines directly to or from the cache it is important that external snoops (inquire cycles) are inhibited to guarantee predictable results when testing. This can be accomplished by inhibiting the snoops externally or by putting the processor in SRAM mode (CR0.CD=CR0.NW=1).

The EADS# input is ignored during external snoop writeback cycles (HITM# asserted), or during the clock after ADS# or EADS# is active. EADS# is also ignored when the processor is in SRAM mode, or when the processor is driving the address bus.

Note that the Pentium processor may drive the address bus in the clock after AHOLD is deasserted. It is the responsibility of the system designer to ensure that address bus contention does not occur. This can be accomplished by not deasserting AHOLD to the Pentium processor until all other bus masters have stopped driving the address bus.

Figure 6-24 shows an inquire cycle that misses both internal caches. Note that both the HIT# and HITM# signals are deasserted two clocks after EADS# is sampled asserted.

Figure 6-25 shows an inquire cycle that invalidates a non-modified line. Note that INV is asserted (high) in the clock that EADS# is returned. Note that two clocks after EADS# is sampled asserted, HIT# is asserted and HITM# is deasserted.

Figure 6-24 and Figure 6-25 both show that the AP pin is sampled/driven along with the address bus, and that the APCHK# pin is driven with the address parity status two clocks after EADS# is sampled asserted.

An inquire cycle that hits a M-state line is shown in Figure 6-26. Both the HIT# and HITM# outputs are asserted two clocks after EADS# is sampled asserted. ADS# for the writeback cycle will occur no earlier than two clocks after the assertion of HITM#.

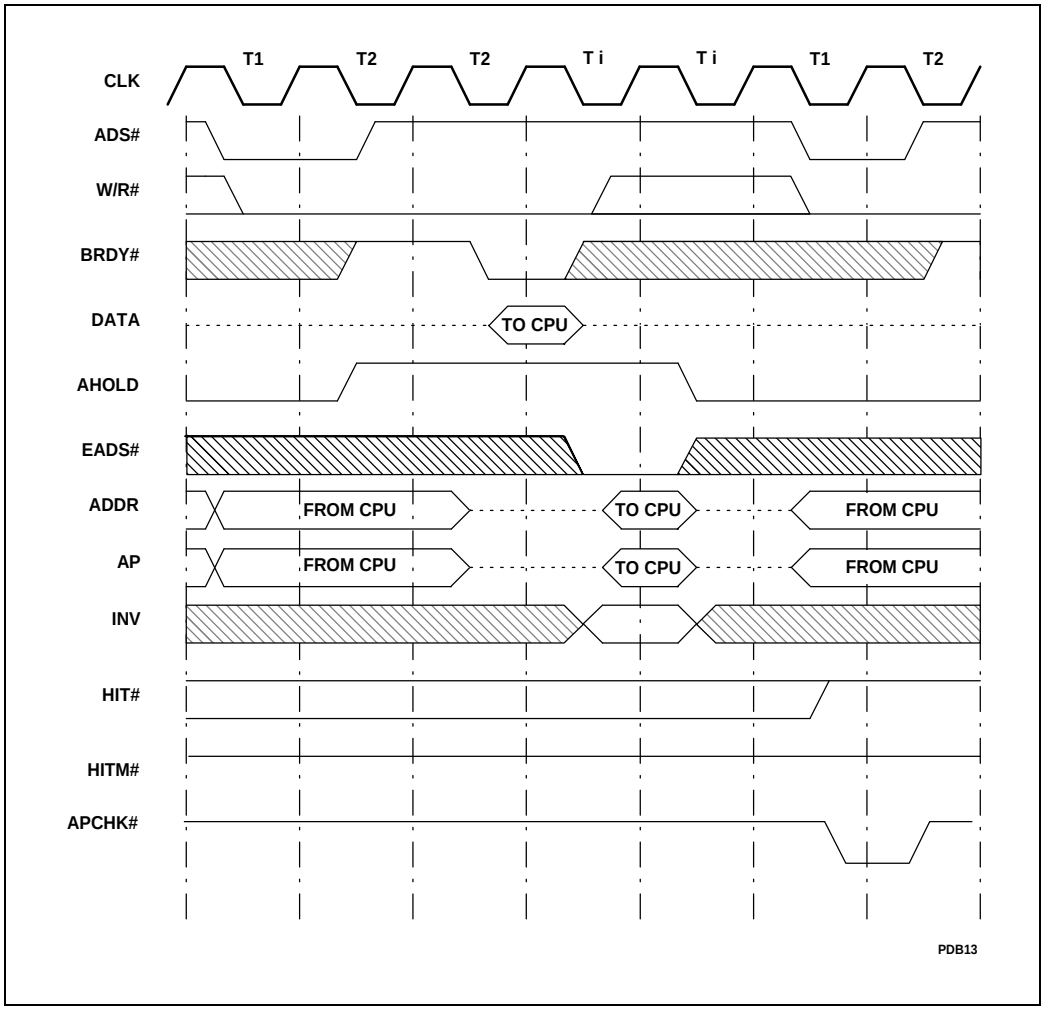


Figure 6-24. Inquire Cycle that Misses the Pentium® Processor Cache

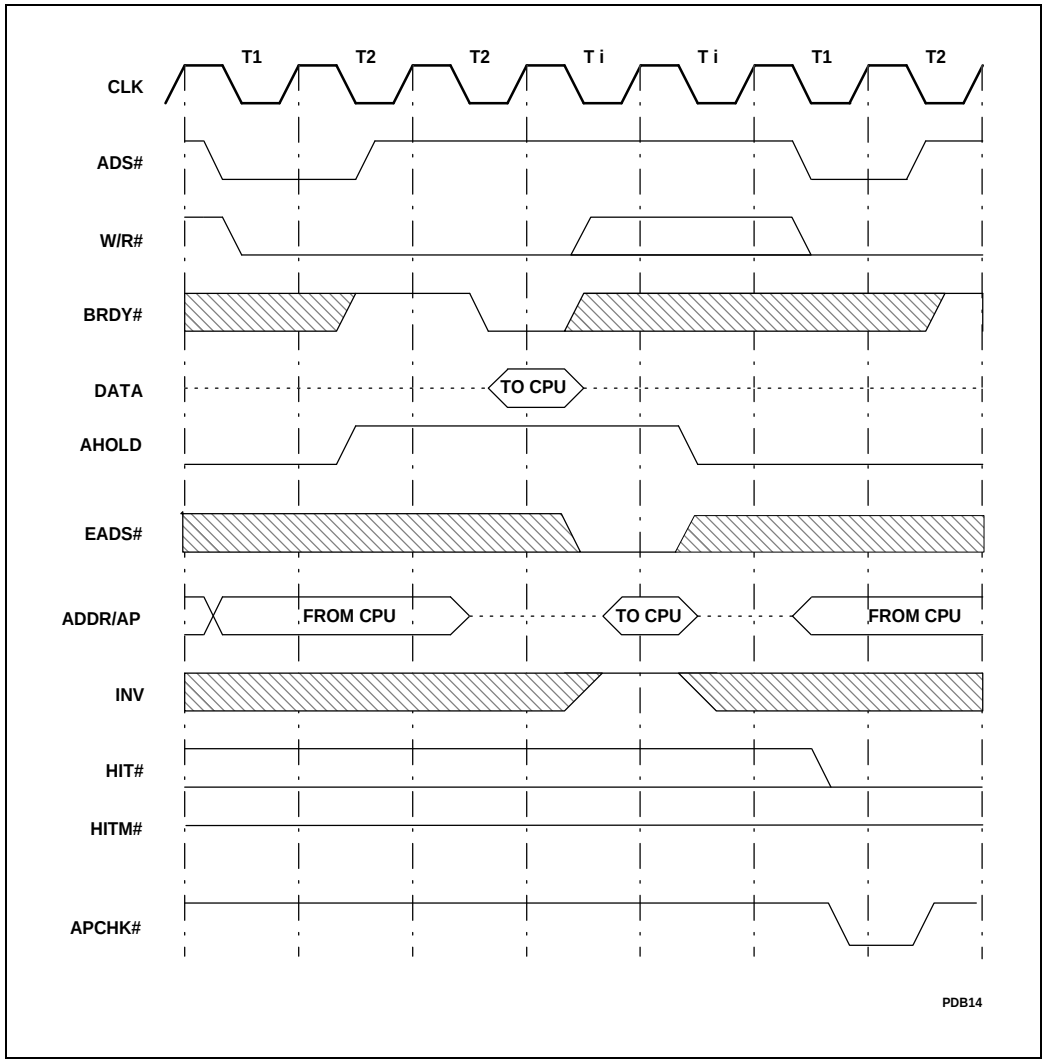


Figure 6-25. Inquire Cycle that Invalidates a Non-M-State Line

HITM# is asserted only if an inquire cycle (external snoop) hits a modified line in the Pentium processor data cache. HITM# is not asserted for internal snoop writeback cycles or cache replacement writeback cycles. HITM# informs the external system that the inquire cycle hit a modified line in the data cache and that line will be written back. Any ADS# driven by the Pentium processor while HITM# is asserted will be the ADS# of the writeback cycle. The HITM# signal will stay active until the last BRDY# is returned for the corresponding inquire cycle. Writeback cycles start at burst address 0.

Note that ADS# is asserted despite the AHOLD signal being active. This ADS# initiates a writeback cycle corresponding to the inquire hit. Such a cycle can be initiated while address lines are floating to support multiple inquires within a single AHOLD session. This functionality can be used during secondary cache replacement processing if its line is larger than the Pentium processor cache line (32 bytes). Although the cycle specification is driven properly by the processor, address pins are not driven because AHOLD forces the Pentium processor off the address bus. If AHOLD is cleared before the Pentium processor drives out the inquire writeback cycle, the Pentium processor will drive the correct address for inquire writeback in the next clock. The ADS# to initiate a writeback cycle as a result of an inquire hit is the only time ADS# will be asserted while AHOLD is also asserted.

Note that in the event of an address parity error during inquire cycles, the snoop cycle will not be inhibited. If the inquire hits a modified line in this situation and an active AHOLD prevents the Pentium processor from driving the address bus, the Pentium processor will potentially writeback a line at an address other than the one intended. If the Pentium processor is not driving the address bus during the writeback cycle, it is possible that memory will be corrupted.

If BOFF# or HLDA were asserted to perform the inquire cycle, the writeback cycle would wait until BOFF# or HLDA was deasserted.

State machines should not depend on a writeback cycle to follow an assertion of HITM#. HITM# may be negated without a corresponding writeback cycle being run. This may occur as a result of the internal caches being invalidated due to the INVD instruction or by testability accesses. Note that inquire cycles occurring during testability accesses will generate unpredictable results. In addition, a second writeback cycle will not be generated for an inquire cycle which hits a line that is already being written back, see Figure 6-28. This can happen if an inquire cycle hits a line in one of the Pentium processor writeback buffers.

6.5.1. Restrictions on Deassertion of AHOLD

To prevent the address and data buses from switching simultaneously, the following restrictions are placed on the negation of AHOLD: (i) AHOLD must not be negated in the same clock as the assertion of BRDY# during a write cycle; (ii) AHOLD must not be negated in the dead clock between write cycles pipelined into read cycles; and (iii) AHOLD must not be negated in the same clock as the assertion of ADS# while HITM# is asserted. Note that there are two clocks between EADS# being sampled asserted and HITM# being asserted, and a further minimum of two clocks between an assertion of HITM# and ADS#.

These restrictions on the deassertion of AHOLD are the only considerations the system designer needs to make to prevent the simultaneous switching of the address and data buses. All other considerations are handled internally.

Figure 6-26 can be used to illustrate restrictions (i) and (iii). AHOLD may be deasserted in Clock 2, 3, or 4, but not in Clock 5, 6, 7, 8 or 9.

Figure 6-27 and Figure 6-28 depict restrictions (i) and (ii) respectively. Note that there are no restrictions on the assertion of AHOLD.

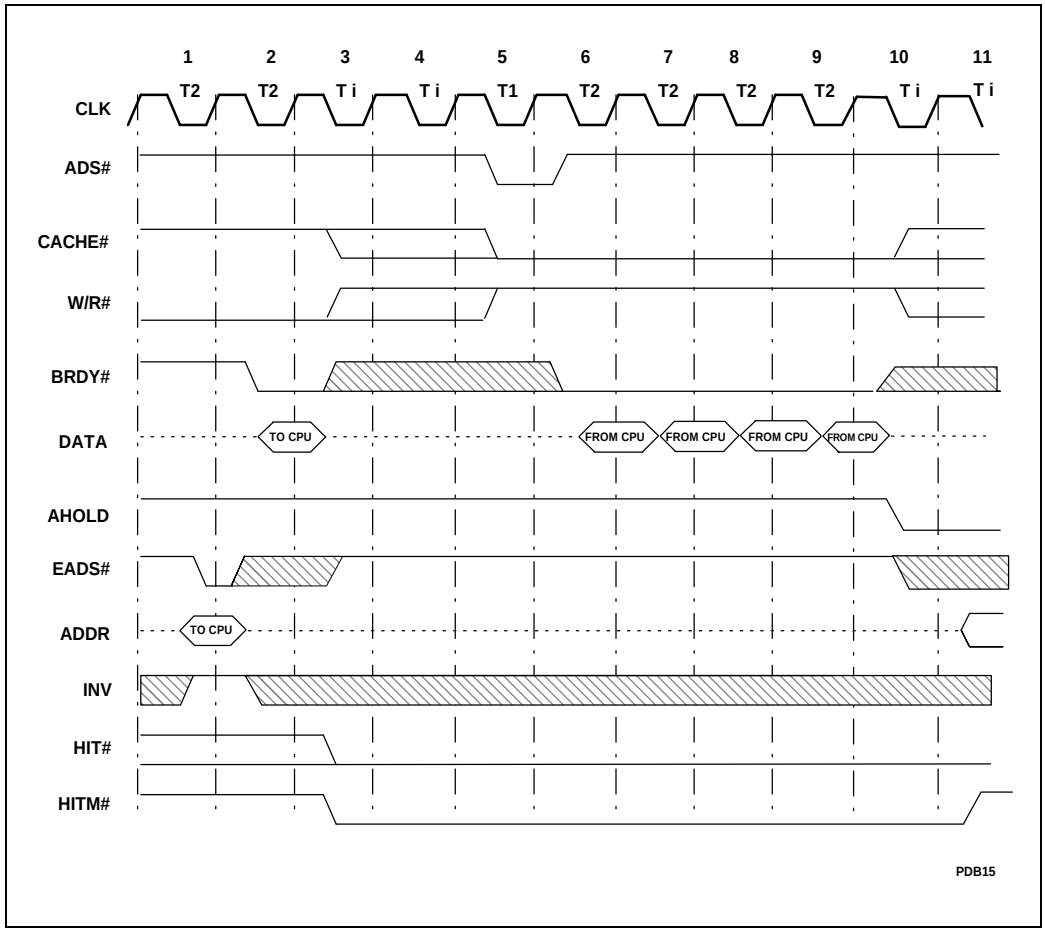


Figure 6-26. Inquire Cycle that Invalidates M-State Line

Figure 6-27 shows a writeback (due to a previous snoop that is not shown). ADS# for the writeback is asserted even though AHOLD is asserted. Note that AHOLD can be deasserted in Clock 2, 4, 7, or 9. AHOLD can not be deasserted in Clock 1, 3, 5, 6, or 8.

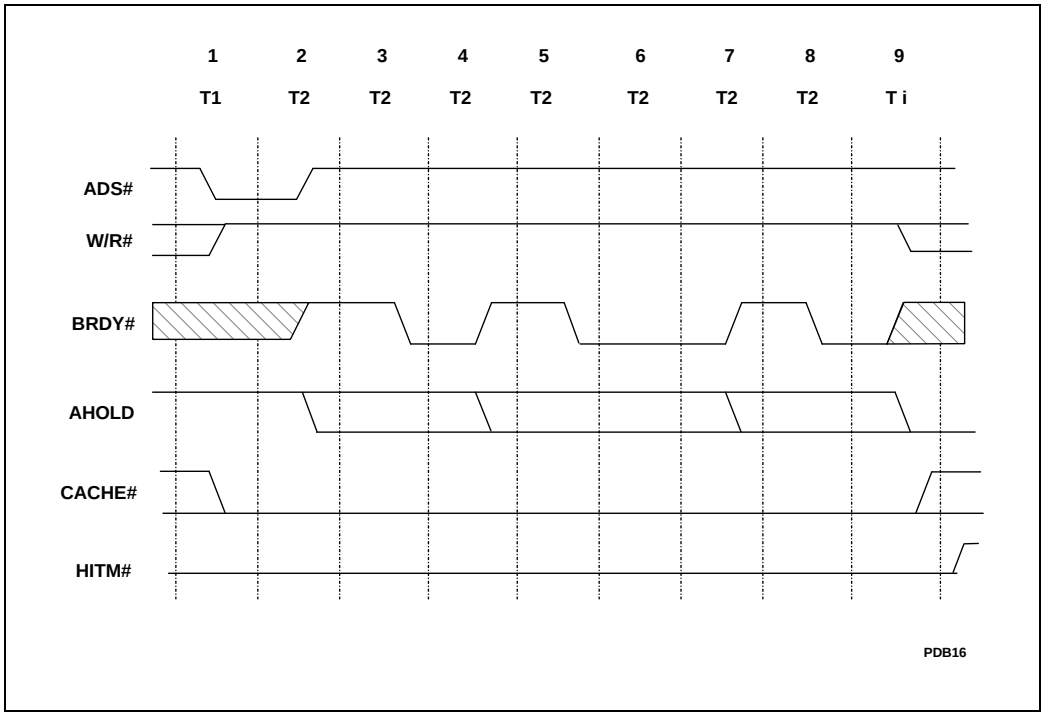


Figure 6-27. AHOLD Restriction during Write Cycles

Figure 6-28 shows a write cycle being pipelined into a read cycle. Note that if AHOLD is asserted in Clock 5, it can be deasserted in Clock 7 before the TD, or in Clock 10 after the TD, but it can not be deasserted in Clock 8 (the TD clock). AHOLD can not be deasserted in Clock 9 because BRDY# for the write cycle is being returned.

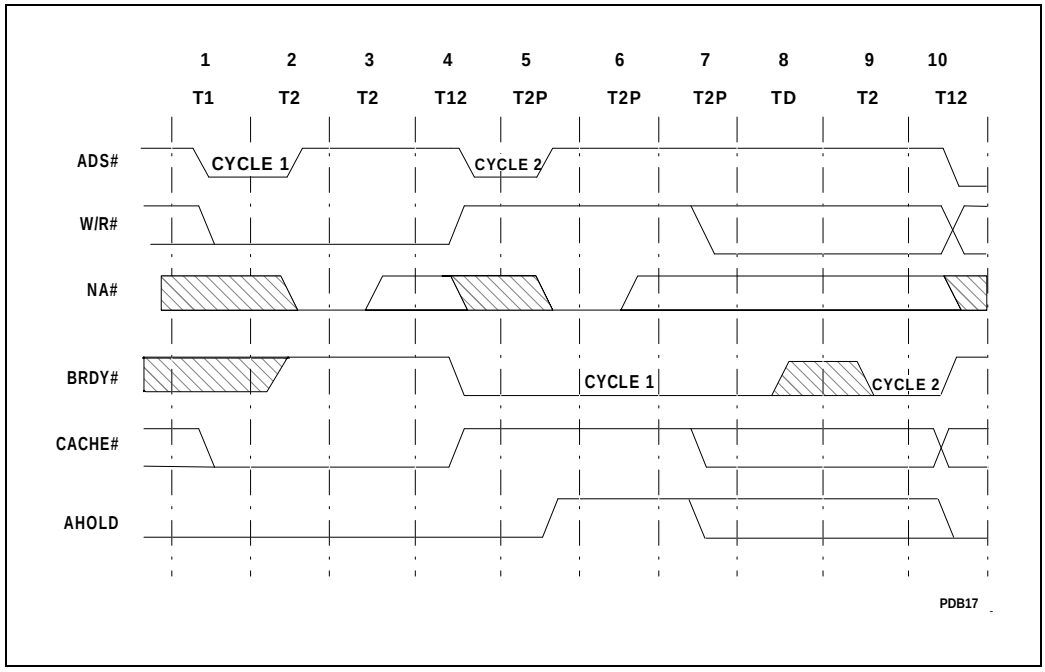


Figure 6-28. AHOLD Restriction during TD

6.5.2. Rate of Inquire Cycles

The Pentium processor can accept inquire cycles at a maximum rate of one every other clock. However, if an inquire cycle hits an M-state line of the Pentium processor, subsequent inquire cycles will be ignored until the line is written back and HITM# is deasserted. EADS# is also ignored the clock after ADS# is asserted.

6.5.3. Internal Snooping

"Internal snoop" is the term used to describe the snooping of the internal code or data caches that is not initiated by the assertion of EADS# by the external system. Internal snooping occurs in the three cases described below. Note that neither HIT# nor HITM# are asserted as a result of an internal snoop.

1. An internal snoop occurs if an access is made to the code cache, and that access is a miss. In this case, if the accessed line is in the S or E state in the data cache, the line is invalidated. If the accessed line is in the M state in the data cache, the line is written back then invalidated.

2. An internal snoop occurs if an access is made to the data cache, and that access is a miss or a writethrough. In this case, if the accessed line is valid in the code cache, the line is invalidated.
3. An internal snoop occurs if there is a write to the accessed and/or dirty bits in the page table/directory entries. In this case, if the accessed line is valid in either the code or data cache, the line is invalidated. If the accessed line is in the M state in the data cache, the line is written back then invalidated.

6.5.4. Snooping Responsibility

In systems with external second level caches allowing concurrent activity of the memory bus and Pentium processor bus, it is desirable to run invalidate cycles concurrently with other Pentium processor bus activity. Writes on the memory bus can cause invalidations in the secondary cache at the same time that the Pentium processor fetches data from the secondary cache. Such cases can occur at any time relative to each other, and therefore the order in which the invalidation is requested, and data is returned to the Pentium processor becomes important.

The Pentium processor always snoops the instruction and data caches when it accepts an inquire cycle. If a snoop comes in during a linefill, the Pentium processor also snoops the line currently being filled. If more than one cacheable cycle is outstanding (through pipelining), the addresses of both outstanding cycles are snooped.

For example, during linefills, the Pentium processor starts snooping the address(es) associated with the line(s) being filled after KEN# has been sampled active for the line(s). Each line is snooped until it is put in the cache. If a snoop hits a line being currently filled, the Pentium processor will assert HIT# and the line will end up in the cache in the S or I state depending on the value of the INV pin sampled during the inquire cycle. The Pentium processor will however use the data returned for that line as a memory operand for the instruction that caused the data cache miss/line fill or execute an instruction contained in a code cache miss/line fill.

Figure 6-29 and Figure 6-30 illustrate the snoop responsibility pickup. Figure 6-29 shows a non-pipelined cycle, while Figure 6-30 illustrates a pipelined cycle. The figures show the earliest EADS# assertion that will cause snooping of the line being cached relative to the first BRDY# or NA#.

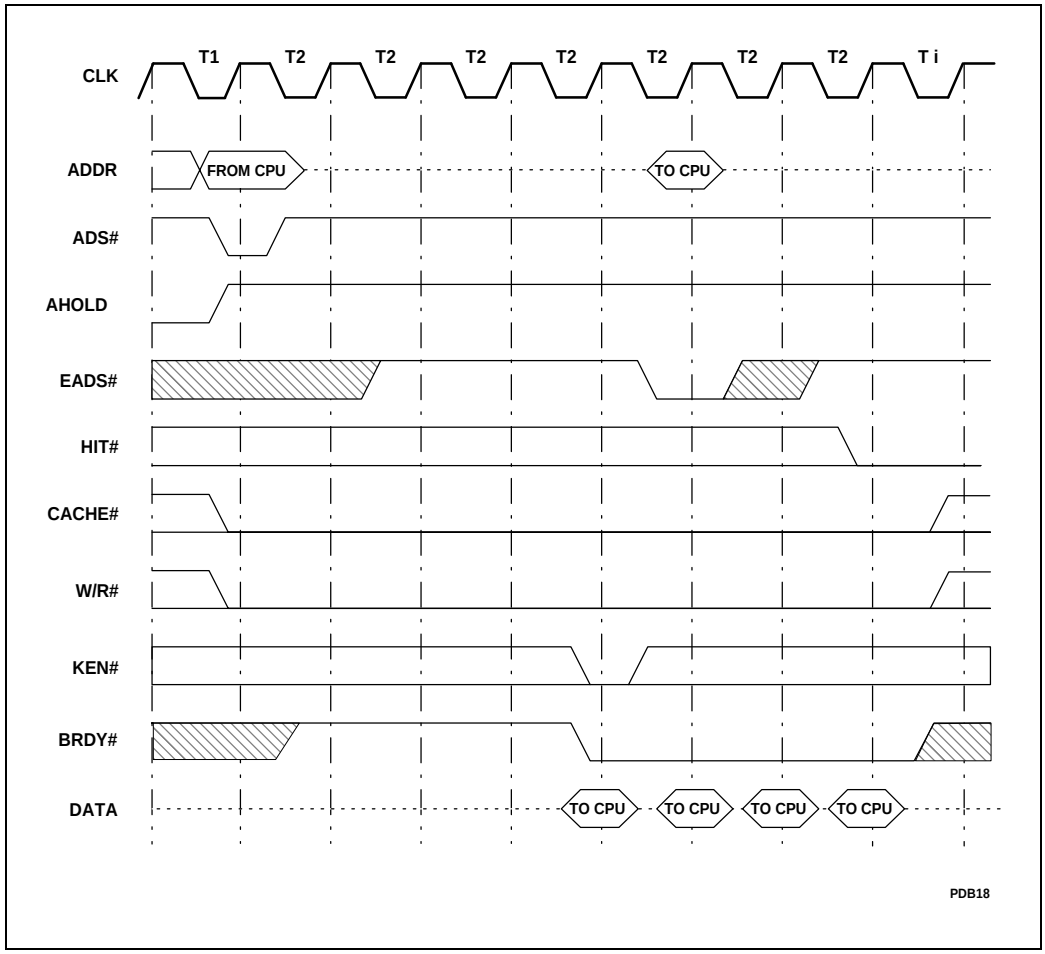


Figure 6-29. Snoop Responsibility Pickup — Non-Pipelined Cycles

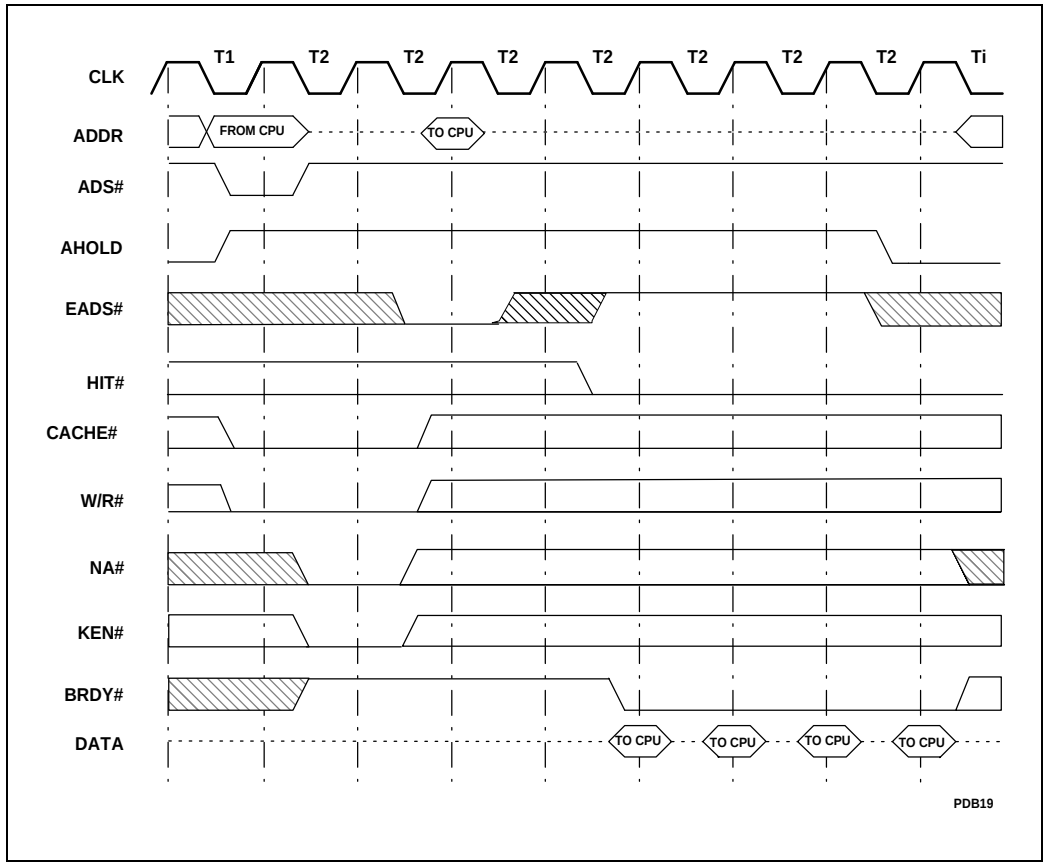


Figure 6-30. Snoop Responsibility Pickup — Pipelined Cycle

The Pentium processor also snoops M state lines in the writeback buffers until the writeback of the M state lines are complete. If a snoop hits an M state line in a writeback buffer, both HIT# and HITM# are asserted. Figure 6-31 illustrates snooping (snoop responsibility drop) of an M state line that is being written back because it has been replaced with a “new” line in the data cache. It shows the latest EADS# assertion, relative to the last BRDY# of the writeback cycle that will result in a snoop hit to the line being written back. HITM# stays asserted until the writeback is complete. Note that an additional ADS# is not asserted during the writeback cycle.

The HIT# signal is a super set of the HITM# signal; it is always asserted with HITM#.

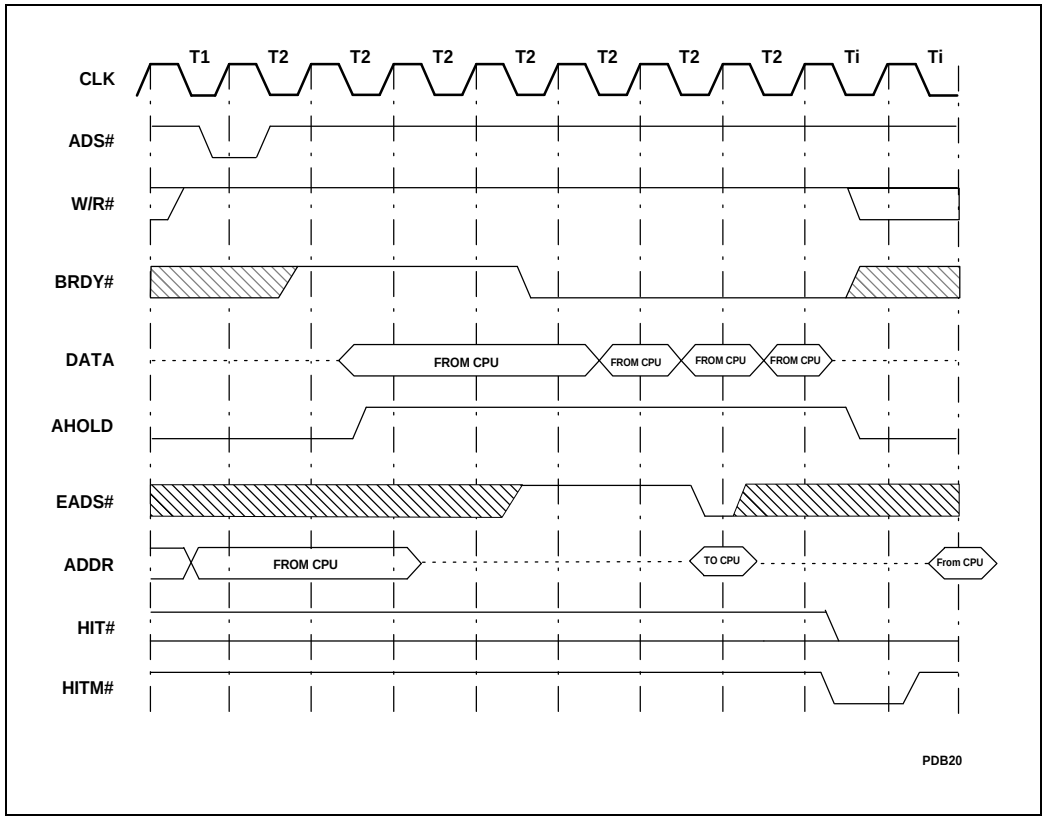


Figure 6-31. Latest Snooping of Writeback Buffer

6.6. SUMMARY OF DUAL PROCESSING BUS CYCLES

The following is a list of bus cycles or bus cycle sequences which would not occur in Pentium processor uni-processor systems, but may be seen in Dual processor systems.

- Locked cycle sequences
- Cycle pipelining
- Cycle ordering due to BOFF#
- Cache line state
- Back-to-back cycles
- Address parity checking
- Flush cycles
- PCHK# assertion

- Synchronous FLUSH# and RESET
- Floating point error handling

6.6.1. Locked Cycle Sequences

1. Locked read to address X
2. Locked write back to address X
3. Locked read to address X
4. Locked write to address X

May occur due to the inter-processor cache consistency mechanism. Refer to Chapter 3.

Implications

Processor bus hardware needs to handle this locked sequence. The only other time the system will see a locked write back is when an external snoop hits a modified line while a locked cycle is in progress (this will occur in a uni-processor or a dual-processor system).

6.6.2. Cycle Pipelining

Inter-processor (Primary/Dual processor) back-to-back write cycles will not be pipelined even if NA# has been asserted. The purpose of this rule is to prevent data bus contention during bus arbitration from one processor to the other. In dual processor mode, the Primary processor may pipeline I/O cycles into I/O cycles from the Dual processor (and vice versa) for any I/O instruction combination (i.e., except I/O writes into writes).

Implications

System hardware designers should be aware of these bus changes.

6.6.3. Cycle Ordering Due to BOFF#

Cycle ordering following an assertion of BOFF# may be different between uni-processor and dual processor modes. This occurs when there are pipelined cycles from both processors, a BOFF# stalls both cycles, and an external snoop hits a modified line in the LRMs cache.

Implications

System hardware designers should be aware of these bus changes.

6.6.4. Cache Line State

In Pentium processor family uni-processor systems, if a line is put into the E state by the system hardware using the WB/WT# signal during the line fill, then all subsequent writes to that line will be handled internally via the on-chip cache. In dual-processor systems, under certain circumstances, even if the system puts a line into the E state using WB/WT#, the dual-processor protocol may force the line to be stored in the S state. Private snooping in dual processor systems can also cause a line to be placed into the S or I state.

Implications

There are no system implications. The system may be required to handle writes to a line which would not otherwise have been seen.

NOTE

In a dual processing system where NW=1 and CD=1 are set, (i.e., SRAM mode), an inquire cycle will invalidate a cache line with INV on a HIT#.

6.6.5. Back-to-Back Cycles

Due to the dual-processor cache consistency protocol, the Primary and Dual processors may follow a write to address X with a write back to a 32-byte area which contains X. This will not occur in uni-processor systems. Also a read to address X may be followed by a write back to a 32-byte area which contains X.

Implications

There are no system implications.

6.6.6. Address Parity Checking

Address parity is checked during every private snoop between the Primary and Dual processors. Therefore, APCHK# may be asserted due to an address parity error during this private snoop. If an error is detected, APCHK# will be asserted 2 clocks after ADS# for one processor clock period. The system can choose to acknowledge this parity error indication at this time or do nothing.

Implications

There are no system implications. The system designers get extra address parity checking with dual processors due to the automatic private snooping.

6.6.7. Synchronous FLUSH# and RESET

When the Dual processor is present, the FLUSH# and RESET signals must be recognized by both processors at the same time.

Implications

FLUSH# and RESET must be asserted on the same clock to both the Primary and Dual processors.

6.6.8. PCHK# Assertion

In a dual-processor configuration, there is the possibility that the PCHK# signal can be asserted either 2 OR 3 CLKs following incorrect parity being detected on the data bus (depending on the bus-to-core ratio).

Implications

Chip sets must account for this difference from the Pentium processor in their logic or state machines.

6.6.9. Flush Cycles

The Primary and Dual processors incorporate a mechanism to present a unified view of the cache flush operation to the system when in dual processing mode. The Dual processor performs the cache flush operation first, then grants the bus to the Primary processor. The Primary processor flushes its internal caches, and then runs the cache flush special cycle.

Implications

The system hardware **must** not assert a subsequent FLUSH# to the processors until the flush acknowledge special cycle has completed on the processor bus. The assertion of FLUSH# to the processors prior to this point would result in a corruption of the dual processing bus arbitration state machines.

6.6.10. Floating Point Error Handling

The Pentium processor, when configured as a Dual processor, ignores the IGNNE# input. The FERR# output is also undefined in the Dual processor.

Implications

None.

7

Electrical Specifications

CHAPTER 7
ELECTRICAL SPECIFICATIONS

This section describes the electrical differences between the Pentium processor (75/90/100/120/133/150/166/200) and the Pentium processor with MMX technology, as well as their respective AC and DC specifications.

7.1. ELECTRICAL CHARACTERISTICS AND DIFFERENCES BETWEEN THE PENTIUM® PROCESSOR WITH MMX™ TECHNOLOGY AND THE PENTIUM® PROCESSOR (75/90/100/120/133/150/166/200)

When designing a Pentium processor with MMX technology system from a Pentium processor (75/90/100/120/133/150/166/200) system, there are a number of electrical differences that require attention. Designing a single motherboard that supports various members of the Pentium processor family including the Pentium processor with MMX technology, Pentium processor (75/90/100/120/133/150/166/200), Pentium OverDrive processor, or future Pentium OverDrive processor with MMX technology can be easily accomplished. Refer to the *Pentium® Processor Flexible Motherboard Design Guidelines Application Note* (Order # 243187) for more information and specific implementation examples.

The following sections highlight key electrical issues pertaining to the Pentium processor power supplies, connection specifications, and buffer models.

7.1.1. Power Supplies

The main electrical difference between the Pentium processor with MMX technology and the Pentium processor (75/90/100/120/133/150/166/200) is the operating voltage. The Pentium processor with MMX technology requires two separate voltage inputs, $V_{CC}2$ and $V_{CC}3$. The $V_{CC}2$ pins supply power to the Pentium processor with MMX technology core, while the $V_{CC}3$ pins supply power to the processor I/O pins.

The Pentium processor (75/90/100/120/133/150/166/200), on the other hand, requires a single voltage supply for all V_{CC} pins. This single supply powers both the core and I/O pins of the Pentium processor (75/90/100/120/133/150/166/200).

By connecting all of the $V_{CC}2$ pins together and all the $V_{CC}3$ pins together on separate power islands, Pentium processor (75/90/100/120/133/150/166/200) designs can easily be converted to support the Pentium processor with MMX technology. In order to maintain compatibility with Pentium processor (75/90/100/120/133/150/166/200)-based platforms, the Pentium processor with MMX technology supports the standard 3.3V specification on its $V_{CC}3$ pins.

7.1.1.1. POWER SUPPLY SEQUENCING

There is no specific power sequence required for powering up or powering down the separate $V_{CC}2$ and $V_{CC}3$ supplies of the Pentium processor with MMX technology. It is recommended that the $V_{CC}2$ and $V_{CC}3$ supplies be either both ON or both OFF within 1 second of each other.

7.1.2. Connection Specifications

Connection specifications for the power and ground inputs, 3.3V inputs and outputs, and the NC/INC and unused inputs are discussed in the following sections.

7.1.2.1. POWER AND GROUND CONNECTIONS

For clean on-chip power distribution, the Pentium processor has 53 V_{CC} (power) and 53 V_{SS} (ground) inputs.

Power and ground connections must be made to all V_{CC} and V_{SS} pins of the Pentium processor. On the circuit board, all V_{CC} pins must be connected to a V_{CC} plane. All V_{SS} pins must be connected to a V_{SS} plane.

It is imperative that the system decoupling be sufficient to maintain ALL V_{CC} pins of the processor within their specified operating range regardless of whether a unified-plane or split-plane processor is installed.

The unified-plane Pentium processor packages have a single internal V_{CC} plane. This plane may be used as the means of conduction between the $V_{CC}2$ and $V_{CC}3$ motherboard power planes when a unified-plane processor is installed in the system. Should such an implementation be used, it must be ensured that the maximum current flowing through the processor package does not exceed 8 Amps, including the power required by the processor. (The Pentium processor (75/90/100/120/133/150/166/200) is a unified plane processor).

Given the above specifications, many different implementations of power distribution are possible for Pentium processor based motherboard designs. These can be broadly categorized into two groups:

1. Unified-plane processors receive power externally to ALL $V_{CC}2$ and $V_{CC}3$ pins, while split-plane processors receive power from independent sources for $V_{CC}2$ and $V_{CC}3$.
2. Unified-plane processors receive power externally to either the $V_{CC}2$ OR $V_{CC}3$ pins, while split-plane processors receive power from independent sources for $V_{CC}2$ and $V_{CC}3$.

The second implementation discussed above implies that when a unified-plane processor is installed, either the $V_{CC}2$ or $V_{CC}3$ voltage regulator will shut down if the voltage from the other regulator is higher than its own output setpoint. This will leave one voltage regulator powering either the $V_{CC}2$ or $V_{CC}3$ pins directly. The remaining functioning voltage regulator must be capable of providing the total required current independently. In the case when a split-plane processor is installed, both voltage regulators must continue to function at the proper voltage levels.

Note that the future Pentium OverDrive processor with MMX technology, although specified for a single voltage, are not unified-plane processors and must have voltage supplied externally to both $V_{CC}2$ and $V_{CC}3$ pins. For the future Pentium OverDrive processor with MMX technology, the exact voltages at the $V_{CC}2$ and $V_{CC}3$ pins are not required to be the same level provided both are within the Socket 7 specifications.

7.1.2.1.1. V_{CC} Measurement Specification

The values of V_{CC} should be measured at the bottom side of the CPU pins using an oscilloscope with a 3 dB bandwidth of at least 20 MHz (100 MS/s digital sampling rate). There should be a short isolation ground lead attached to a CPU pin on the bottom side of the board.

The measurement should be taken at the following V_{CC}/V_{SS} pairs: AN13/AM10, AN21/AM18, AN29/AM26, AC37/Z36, U37/R36, L37/H36, A25/B28, A17/B20, A7/B10, G1/K2, S1/V2, AC1/Z2. Note that on the Pentium processor with MMX technology , one-half of these pins are $V_{CC}2$ while the others are $V_{CC}3$; the operating ranges for the $V_{CC}2$ and $V_{CC}3$ pins are specified at different voltages.

The display should show continuous sampling of the V_{CC} line, at 20 mV/div, and 500 nS/div with the trigger point set to the center point of the range. Slowly move the trigger to the high and low ends of the specification, and verify that excursions beyond these limits are not observed. There are no allowances for crossing the high and low limits of the voltage specification. For more information on measurement techniques, see the *Implementation Guidelines for 3.3V Pentium® Processors with VR/VRE Specifications* (order # 242687) and *Voltage Guidelines for Pentium® Processors with MMX™ Technology* (order # 243186) application notes.

7.1.2.1.2. Decoupling Recommendations

Liberal decoupling capacitance should be placed near the Pentium processor. The Pentium processor driving its large address and data buses at high frequencies can cause transient power surges, particularly when driving large capacitive loads.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the Pentium processor and decoupling capacitors as much as possible. These capacitors should be evenly distributed around each component on the power plane. Capacitor values should be chosen to ensure they eliminate both low and high frequency noise components.

For the Pentium processor, the power consumption can transition from a low level of power to a much higher level (or high to low power) very rapidly. A typical example would be entering or exiting the Stop Grant State. Another example would be executing a HALT instruction, causing the Pentium processor to enter the AutoHALT Power Down State, or transitioning from HALT to the Normal State. All of these examples may cause abrupt changes in the power being consumed by the Pentium processor. Note that the AutoHALT Power Down feature is always enabled even when other power management features are not implemented.

Bulk storage capacitors with a low ESR (Effective Series Resistance) in the 10Ω to 100Ω range are required to maintain a regulated supply voltage during the interval between the time the current load changes and the point that the regulated power supply output can react to the change in load. In order to reduce the ESR, it may be necessary to place several bulk storage capacitors in parallel.

These capacitors should be placed near the Pentium processor on the power plane(s) to ensure that the supply voltage stays within specified limits during changes in the supply current during operation.

Detailed decoupling recommendations are provided in the *Flexible Motherboard Design Guidelines* Application Note (Order #243187).

7.1.2.2. 3.3V INPUTS AND OUTPUTS

The inputs and outputs of the Pentium processor comply with the 3.3V JEDEC standard levels. Both inputs and outputs are also TTL-compatible, although the inputs cannot tolerate voltage swings above the $V_{IN}3$ (max.) specification.

System support components which use TTL-compatible inputs will interface to the Pentium processor without extra logic. This is because the Pentium processor drives according to the 5V TTL specification (but not beyond 3.3V).

For Pentium processor inputs, the voltage must not exceed the 3.3V $V_{IN}3$ (max.) specification. System support components can consist of 3.3V devices or open-collector devices. In an open-collector configuration, the external resistor should be biased to $V_{CC}3$.

All pins, other than the CLK and PICCLK of the Pentium processor (75/90/100/120/133/150/166/200), are 3.3V-only. If an 8259A interrupt controller is used, for example, the system must provide level converters between the 8259A and the Pentium processor.

The CLK and PICCLK inputs of the Pentium processor (75/90/100/120/133/150/166/200) are 5V tolerant. This allows a 5V clock driver to be used for the Pentium processor (75/90/100/120/133/150/166/200). These inputs, however, are NOT 5V tolerant on the Pentium processor with MMX technology. The Pentium processor with MMX technology CLK and PICCLK inputs are 3.3V tolerant only. A 3.3V clock driver should be used in systems designed to support both the Pentium processor with MMX technology and Pentium processor (75/90/100/120/133/150/166/200).

7.1.2.3. NC/INC AND UNUSED INPUTS

All NC and INC pins must remain unconnected.

For reliable operation, always connect unused inputs to an appropriate signal level. Unused active low inputs of the Pentium processor with MMX technology should be connected to $V_{CC}3$, and unused active low inputs of the Pentium processor (75/90/100/120/133/150/166/200) should be connected to V_{CC}. Unused active high inputs should be connected to V_{SS} (ground).

7.1.3. Buffer Models

The structure of the buffer models for the Pentium processor with MMX technology and the Pentium processor (75/90/100/120/133/150/166/200) are identical. Some of the values of the components have changed to reflect the minor manufacturing process and package differences between the processors. The system should see insignificant differences between the AC behavior of the Pentium processor with MMX technology and the Pentium processor (75/90/100/120/133/150/166/200).

Simulation of AC timings using the Pentium processor buffer models is recommended to ensure robust system designs. Pay specific attention to the signal quality restrictions imposed by 3.3V buffers.

7.2. ABSOLUTE MAXIMUM RATINGS

Table 7-1 provides stress ratings only. Functional operation at the Absolute Maximum Ratings is not implied or guaranteed. Functional operating conditions are given in the AC and DC specification tables.

Extended exposure to the maximum ratings may affect device reliability. Furthermore, although the Pentium processor contains protective circuitry to resist damage from electrostatic discharge, always take precautions to avoid high static voltages or electric fields.

Table 7-1. Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit	Notes
	Storage Temperature	-65	150	^{o}C	
	Case Temperature Under Bias	-65	110	^{o}C	
$V_{CC}3$	V_{CC3} Supply Voltage with respect to V_{SS}	-0.5	4.6	V	1
$V_{CC}2$	V_{CC2} Supply Voltage with respect to V_{SS}	-0.5	3.7	V	2
$V_{IN}3$	3V Only Buffer DC Input Voltage	-0.5	$V_{CC}3$ +0.5 not to exceed $V_{CC}3$ max	V	3
V_{INSB3}	5V Safe Buffer DC Input Voltage	-0.5	6.5	V	4

NOTES:

1. Applies to all the V_{CC} inputs of the Pentium® processor (75/90/100/120/133/150/0166/200).
2. The $V_{CC}2$ pins are defined only for the Pentium processor with MMX™ technology.
3. Applies to all Pentium processor inputs except CLK and PICCLK of the Pentium processor (75/90/100/120/133/150/166/200).
4. Applies only to CLK and PICCLK of the Pentium processor (75/90/100/120/133/150/166/200). See Table 7-4.

WARNING

Stressing the device beyond the Absolute Maximum Ratings may cause permanent damage. These are stress ratings only. Operation beyond the DC Specifications is not recommended or guaranteed and extended exposure beyond the DC Specifications may effect device reliability.

7.3. DC SPECIFICATIONS

Table 7-2 through Table 7-7 list the DC Specifications of the Pentium processor.

Table 7-2. V_{CC} and T_{CASE} Specifications

Symbol	Parameter	Min	Nom	Max	Unit	Notes
T_{CASE}	Case Temperature	0		70	°C	
$V_{CC}2$	$V_{CC}2$ Voltage	2.7	2.8	2.9	V	Range = 2.8 ± 3.57%, (1), (2)
$V_{CC}3$	$V_{CC}3$ Voltage	3.135	3.3	3.6	V	Applies to all V_{CC} inputs of the Pentium® processor (75/90/100/120/133/150/166/200), Range = 3.3 -5%, +9.09%, (2)

NOTES:

1. The $V_{CC}2$ specification applies only to the Pentium processor with MMX™ technology.
2. See the V_{CC} measurement specification section earlier in this chapter.

Table 7-3. 3.3V DC Specifications

Symbol	Parameter	Min	Max	Unit	Notes
$V_{IL}3$	Input Low Voltage	-0.3	0.8	V	TTL Level
$V_{IH}3$	Input High Voltage	2.0	$V_{CC}3$ +0.3	V	TTL Level (1)
$V_{OL}3$	Output Low Voltage		0.4	V	TTL Level (2, 4)
$V_{OH}3$	Output High Voltage	2.4		V	TTL Level (3)

NOTES:

1. Parameter measured at nominal V_{CC}.
2. Parameter measured at -4 mA.
3. Parameter measured at 3 mA.
4. In dual processing systems, up to a 10mA load from the second processor may be observed on the PCHK# signal. Based on silicon characterization data, V_{OL} of PCHK# will remain less than 400 mV even with a 10 mA load. PCHK# V_{OL} will increase to approximately 500 mV with a 14 mA load (worst case for a dual processor system with a 4 mA system load).

Table 7-4. 3.3V (5V Safe) DC Specifications

Symbol	Parameter	Min	Max	Unit	Notes
$V_{IL}5$	Input Low Voltage	-0.3	0.8	V	TTL Level (1)
$V_{IH}5$	Input High Voltage	2.0	5.55	V	TTL Level (1)

NOTE:

1. Applies only to CLK and PICCLK of the Pentium® processor (75/90/100/120/133/150/166/200).

Table 7-5. I_{CC} Specifications

Pentium® Processor with MMX™ Technology (Measured at $V_{cc}2$=2.9V and $V_{cc}3$=3.6V)					
Symbol	**Parameter**	**Min**	**Max**	**Unit**	**Notes**
$I_{CC}2$	Power Supply Current		5700 4750	mA mA	200 MHz (1) 166 MHz (1)
$I_{CC}3$	Power Supply Current		650 540	mA mA	200 MHz (1) 166 MHz (1)
Pentium Processor (75/90/100/120/133/150/166/200) (Measured at V_{cc}=3.6V)					
Symbol	**Parameter**	**Min**	**Max**	**Unit**	**Notes**
I_{CC}	Power Supply Current		4600 4250 3850 3400 3730 3250 2950 2650	mA mA mA mA mA mA mA mA	200 MHz (1) 166 MHz (1) 150 MHz (1) 133 MHz (1) 120 MHz (1) 100 MHz (1) 90 MHz (1) 75 MHz (1)

NOTE:

1. This value should be used for power supply design. It was determined using a worst case instruction mix and maximum V_{CC}. Power supply transient response and decoupling capacitors must be sufficient to handle the instantaneous current changes occurring during transitions from Stop Clock to full Active modes.

Table 7-6. Power Dissipation Requirements for Thermal Design

Pentium® Processor with MMX™ Technology (Measured at $V_{cc}2$=2.8V and $V_{cc}3$=3.3V)				
Parameter	**Typical (1)**	**Max (2)**	**Unit**	**Notes**
Active Power	7.3 (6) 6.1 (6)	15.7 (5) 13.1 (5)	Watts Watts	200 MHz 166 MHz
Stop Grant / Auto Halt Powerdown Power	N/A	2.41 2.05	Watts	200 MHz (3) 166 MHz (3)
Stop Clock Power	0.03	< 0.3	Watts	All frequencies (4)
Pentium Processor (75/90/100/120/133/150/166/200) (Measured at V_{cc}=3.3V)				
Parameter	**Typical (1)**	**Max (2)**	**Unit**	**Notes**
Active Power	6.5 (6) 5.4 (6) 4.9 (6) 4.3 (6) 5.06 (6) 3.9 (6) 3.5 (6) 3.0 (6)	15.5 (5) 14.5 (5) 11.6 (5) 11.2 (5) 12.81 (5) 10.1 (5) 9.0 (5) 8.0 (5)	Watts Watts Watts Watts Watts Watts Watts Watts	200 Mhz, V_{CC} = 3.5V 166 MHz, V_{CC} = 3.5V 150 MHz 133 MHz 120 MHz 100 MHz 90 MHz 75 MHz
Stop Grant / Auto Halt Powerdown Power		2.5 2.1 1.9 1.7 1.76 1.55 1.40 1.20	Watts Watts Watts Watts Watts Watts Watts Watts	200 MHz (3) 166 MHz (3) 150 MHz (3) 133 MHz (3) 120 MHz (3) 100 MHz (3) 90 MHz (3) 75 MHz (3)
Stop Clock Power	0.03	<0.3	Watts	All frequencies (4)

NOTES:

1. This is the typical power dissipation in a system. This value is expected to be the average value that will be measured in a system using a typical device at nominal V_{CC} running typical applications. This value is highly dependent upon the specific system configuration. Typical power specifications are not tested.
2. Systems must be designed to thermally dissipate the maximum active power dissipation. It is determined using worst case instruction mix with nominal V_{CC}, and also takes into account the thermal time constants of the package.
3. Stop Grant/Auto Halt Power Down Power Dissipation is determined by asserting the STPCLK# pin or executing the HALT instruction.
4. Stop Clock Power Dissipation is determined by asserting the STPCLK# pin and then removing the external CLK input.
5. Active Power (max) is the maximum power dissipation under normal operating conditions at nominal V_{CC}, worst-case temperature, while executing the worst case power instruction mix.
6. Active Power (typ) is the average power measured in a system using a typical device running typical applications under normal operating conditions at nominal Vcc and room temperature. Active Power, max is identical to Thermal Design Power (TDP), max.

Table 7-7. Input and Output Characteristics

Symbol	Parameter	Min	Max	Unit	Notes
C_{IN}	Input Capacitance		15	pF	(4)
C_O	Output Capacitance		20	pF	(4)
$C_{I/O}$	I/O Capacitance		25	pF	(4)
C_{CLK}	CLK Input Capacitance		15	pF	(4)
C_{TIN}	Test Input Capacitance		15	pF	(4)
C_{TOUT}	Test Output Capacitance		20	pF	(4)
C_{TCK}	Test Clock Capacitance		15	pF	(4)
I_{LI}	Input Leakage Current		±15	uA	$0 < V_{IN} < V_{IL}$, $V_{IH} > V_{IN} > V_{CC}$ (1)
I_{LO}	Output Leakage Current		±15	uA	$0 < V_{IN} < V_{IL}$, $V_{IH} > V_{IN} > V_{CC}$ (1)
I_{IH}	Input Leakage Current		200	uA	V_{IN} = 2.4V (3)
I_{IL}	Input Leakage Current		-400	uA	V_{IN} = 0.4V (2,5)

NOTES:

1. This parameter is for inputs/outputs without an internal pull up or pull down.
2. This parameter is for inputs with an internal pull up.
3. This parameter is for inputs with an internal pull down.
4. Guaranteed by design.
5. For the Pentium processor with MMX™ technology, the I_{IL} specification applies to the HITM# pin when it is driven as an input (e.g., during JTAG mode).

7.4. AC SPECIFICATIONS

The AC specifications consist of output delays, input setup requirements and input hold requirements. All AC specifications (with the exception of those for the TAP signals and APIC signals) are relative to the rising edge of the CLK input.

All timings are referenced to 1.5 volts for both “0” and “1” logic levels unless otherwise specified. Within the sampling window, a synchronous input must be stable for correct Pentium processor operation.

Each valid delay is specified for a 0 pF load. The system designer should use I/O buffer modeling to account for signal flight time delays.

Do not select a bus fraction and clock speed which will cause the processor to exceed its internal maximum frequency specification. Each Pentium processor is specified to operate within a single bus-to-core ratio and a specific minimum to maximum bus frequency range (corresponding to a minimum to maximum core frequency range). Operation in other bus-to-core ratios or outside the specified operating frequency range is not supported. For example, the 150 MHz Pentium processor does not operate beyond the 60 MHz bus frequency and only supports the 2/5 bus-to-core ratio; it does not support the 1/3, 1/2, or 2/3 bus-to-core ratios. Table 4-3 clarifies and summarizes these specifications.

7.4.1. AC Timing Tables for a 66-MHz Bus

Table 7-8. Pentium® Processor AC Specifications for 66-MHz Bus Operation

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	33.33	66.6	MHz	7-1	
t_{1a}	CLK Period	15.0	30.0	nS	7-1	
t_{1b}	CLK Period Stability		±250	pS		Adjacent Clocks (1,25)
t_2	CLK High Time	4.0		nS	7-1	2V(1)
t_3	CLK Low Time	4.0		nS	7-1	0.8V(1)
t_4	CLK Fall Time	0.15	1.5	nS	7-1	(2.0V–0.8V)(1)
t_5	CLK Rise Time	0.15	1.5	nS	7-1	(0.8V–2.0V)(1)
t_{6a}	PWT, PCD, CACHE# Valid Delay	1.0	7.0	nS	7-2	
t_{6b}	AP Valid Delay	1.0	8.5	nS	7-2	
t_{6c}	BE0-7#, LOCK# Valid Delay	0.9	7.0	nS	7-2	4
t_{6d}	ADS# Valid Delay	0.8	6.0	nS	7-2	
t_{6e}	ADSC#, D/C#, W/R#, SCYC, Valid Delay	0.8	7.0	nS	7-2	
t_{6f}	M/IO# Valid Delay	0.8	5.9	nS	7-2	
t_{6g}	A3–A16 Valid Delay	0.5	6.3	nS	7-2	
t_{6h}	A17–A31 Valid Delay	0.6	6.3	nS	7-2	
t_7	ADS#, ADSC#, AP, A3-A31, PWT, PCD, BE0-7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Float Delay		10.0	nS	7-3	1
t_{8a}	APCHK#, IERR#, FERR# Valid Delay	1.0	8.3	nS	7-2	4
t_{8b}	PCHK# Valid Delay	1.0	7.0	nS	7-2	4
t_{9a}	BREQ Valid Delay	1.0	8.0	nS	7-2	4
t_{9b}	SMIACT# Valid Delay	1.0	7.3	nS	7-2	4
t_{9c}	HLDA Valid Delay	1.0	6.8	nS	7-2	
t_{10a}	HIT# Valid Delay	1.0	6.8	nS	7-2	
t_{10b}	HITM# Valid Delay	0.7	6.0	nS	7-2	
t_{11a}	PM0-1, BP0-3 Valid Delay	1.0	10.0	nS	7-2	

Table 7-8. Pentium® Processor AC Specifications for 66-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{11b}	PRDY Valid Delay	1.0	8.0	nS	7-2	
t_{12}	D0-D63, DP0-7 Write Data Valid Delay	1.3	7.5	nS	7-2	
t_{13}	D0-D63, DP0-3 Write Data Float Delay		10.0	nS	7-3	1
t_{14}	A5-A31 Setup Time	6.0		nS	7-4	26
t_{15}	A5-A31 Hold Time	1.0		nS	7-4	
t_{16a}	INV, AP Setup Time	5.0		nS	7-4	
t_{16b}	EADS# Setup Time	5.0		nS	7-4	
t_{17}	EADS#, INV, AP Hold Time	1.0		nS	7-4	
t_{18a}	KEN# Setup Time	5.0		nS	7-4	
t_{18b}	NA#, WB/WT# Setup Time	4.5		nS	7-4	
t_{19}	KEN#, WB/WT#, NA# Hold Time	1.0		nS	7-4	
t_{20}	BRDY#, BRDYC# Setup Time	5.0		nS	7-4	
t_{21}	BRDY#, BRDYC# Hold Time	1.0		nS	7-4	
t_{22}	AHOLD, BOFF# Setup Time	5.5		nS	7-4	
t_{23}	AHOLD, BOFF# Hold Time	1.0		nS	7-4	
t_{24a}	BUSCHK#, EWBE#, HOLD Setup Time	5.0		nS	7-4	
t_{24b}	PEN# Setup Time	4.8		nS	7-4	
t_{25a}	BUSCHK#, EWBE#, PEN# Hold Time	1.0		nS	7-4	
t_{25b}	HOLD Hold Time	1.5		nS	7-4	
t_{26}	A20M#, INTR, STPCLK# Setup Time	5.0		nS	7-4	12, 16
t_{27}	A20M#, INTR, STPCLK# Hold Time	1.0		nS	7-4	13
t_{28}	INIT, FLUSH#, NMI, SMI#, IGNNE# Setup Time	5.0		nS	7-4	12, 16, 17
t_{29}	INIT, FLUSH#, NMI, SMI#, IGNNE# Hold Time	1.0		nS	7-4	13
t_{30}	INIT, FLUSH#, NMI, SMI#, IGNNE# Pulse Width, Async	2.0		CLKs		15, 17

Table 7-8. Pentium® Processor AC Specifications for 66-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{31}	R/S# Setup Time	5.0		nS	7-4	12, 16, 17
t_{32}	R/S# Hold Time	1.0		nS	7-4	13
t_{33}	R/S# Pulse Width, Async.	2.0		CLKs		15, 17
t_{34}	D0-D63, DP0-7 Read Data Setup Time	2.8		nS	7-4	
t_{35}	D0-D63, DP0-7 Read Data Hold Time	1.5		nS	7-4	
t_{36}	RESET Setup Time	5.0		nS	7-5	11, 12, 16
t_{37}	RESET Hold Time	1.0		nS	7-5	11, 13
t_{38}	RESET Pulse Width, V_{CC} & CLK Stable	15.0		CLKs	7-5	11, 17
t_{39}	RESET Active After V_{CC} & CLK Stable	1.0		mS	7-5	Power up
t_{40}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time	5.0		nS	7-5	12, 16, 17
t_{41}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Hold Time	1.0		nS	7-5	13
t_{42a}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time, Async.	2.0		CLKs		To RESET falling edge(16)
t_{42b}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#, BRDYC#, BUSCHK#) Hold Time, Async.	2.0		CLKs		To RESET falling edge(27)
t_{42c}	Reset Configuration Signals (BRDYC#, BUSCHK#) Setup Time, Async.	3.0		CLKs		To RESET falling edge(27)
t_{42d}	Reset Configuration Signal BRDYC# Hold Time, RESET driven synchronously	1.0		nS	7-5	To RESET falling edge(1,27)
t_{43a}	BF0, BF1, CPUTYP Setup Time	1.0		mS	7-5	To RESET falling edge(22)
t_{43b}	BF0, BF1, CPUTYP Hold Time	2.0		CLKs		To RESET falling edge(22)
t_{43c}	APICEN, BE4# Setup Time	2.0		CLKs		To RESET falling edge

Table 7-8. Pentium® Processor AC Specifications for 66-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{43d}	APICEN, BE4# Hold Time	2.0		CLKs		To RESET falling edge
t_{44}	TCK Frequency		16.0	MHz	7-1	
t_{45}	TCK Period	62.5		nS	7-1	
t_{46}	TCK High Time	25.0		nS	7-1	2V(1)
t_{47}	TCK Low Time	25.0		nS	7-1	0.8V(1)
t_{48}	TCK Fall Time		5.0	nS	7-1	(2.0V–0.8V) (1,8,9)
t_{49}	TCK Rise Time		5.0	nS	7-1	(0.8V–2.0V) (1,8,9)
t_{50}	TRST# Pulse Width	40.0		nS	7-7	Asynchronous (1)
t_{51}	TDI, TMS Setup Time	5.0		nS	7-6	7
t_{52}	TDI, TMS Hold Time	13.0		nS	7-6	7
t_{53}	TDO Valid Delay	2.5	20.0	nS	7-6	8
t_{54}	TDO Float Delay		25.0	nS	7-6	1, 8
t_{55}	All Non-Test Outputs Valid Delay	2.5	20.0	nS	7-6	3, 8, 10
t_{56}	All Non-Test Outputs Float Delay		25.0	nS	7-6	1, 3, 8, 10
t_{57}	All Non-Test Inputs Setup Time	5.0		nS	7-6	3, 7, 10
t_{58}	All Non-Test Inputs Hold Time	13.0		nS	7-6	3, 7, 10
APIC AC Specifications						
t_{60a}	PICCLK Frequency	2.0	16.66	MHz	7-1	
t_{60b}	PICCLK Period	60.0	500.0	nS	7-1	
t_{60c}	PICCLK High Time	15.0		nS	7-1	
t_{60d}	PICCLK Low Time	15.0		nS	7-1	
t_{60e}	PICCLK Rise Time	0.15	2.5	nS	7-1	
t_{60f}	PICCLK Fall Time	0.15	2.5	nS	7-1	
t_{60g}	PICD0-1 Setup Time	3.0		nS	7-4	To PICCLK
t_{60h}	PICD0-1 Hold Time	2.5		nS	7-4	To PICCLK

Table 7-8. Pentium® Processor AC Specifications for 66-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{60i}	PICD0-1 Valid Delay (LtoH)	4.0	38.0	nS	7-2	From PICCLK (28)
t_{60j}	PICD0-1 Valid Delay (HtoL)	4.0	22.0	nS	7-2	From PICCLK (28)
t_{61}	PICCLK Setup Time	5.0		nS		To CLK (30)
t_{62}	PICCLK Hold Time	2.0		nS		To CLK (30)
t_{63}	PICCLK Ratio (CLK/PICCLK)	4				31

NOTE: See notes following Table 7-13.

Table 7-9. Pentium® Processor Dual Processor Mode AC Specifications for 66-MHz Bus Operation

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{80a}	PBREQ#, PBGNT#, PHIT# Flight Time	0	2.0	nS		29
t_{80b}	PHITM# Flight Time	0	1.8	nS		29
t_{83a}	A5-A31 Setup Time	3.7		nS	7-4	18, 21, 26
t_{83b}	D/C#, W/R#, CACHE#, LOCK#, SCYC Setup Time	4.0			7-4	18, 21
t_{83c}	ADS#, M/IO# Setup Time	5.8		nS	7-4	18, 21
t_{83d}	HIT#, HITM# Setup Time	6.0		nS	7-4	18, 21
t_{83e}	HLDA Setup Time	6.0		nS	7-4	18, 21
t_{84a}	CACHE#, HIT# Hold Time	1.0		nS	7-4	18, 21
t_{84b}	ADS#, D/C#, W/R#, M/IO#, A5-A31, HLDA, SCYC Hold Time	0.8		nS	7-4	18, 21
t_{84c}	LOCK# Hold Time	0.9		nS	7-4	18, 21
t_{84d}	HITM# Hold Time	0.7		nS	7-4	18, 21
t_{85}	DPEN# Valid Time		10.0	CLKs		18, 19, 23
t_{86}	DPEN# Hold Time	2.0		CLKs		18, 20, 23
t_{87}	APIC ID (BE0#-BE3#) Setup Time	2.0		CLKs	7-5	To RESET falling edge(23)
t_{88}	APIC ID (BE0#-BE3#) Hold Time	2.0		CLKs	7-5	From RESET falling edge(23)
t_{89}	D/P# Valid Delay	1.0	8.0	nS	7-2	Primary processor only

NOTE: See notes following Table 7-13.

7.4.2. AC Timing Tables for a 60-MHz Bus

Table 7-10. Pentium® Processor AC Specifications for 60-MHz Bus Operation

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	30.0	60.0	MHz	7-1	
t_{1a}	CLK Period	16.67	33.33	nS	7-1	
t_{1b}	CLK Period Stability		±250	pS		Adjacent Clocks (1,25)
t_2	CLK High Time	4.0		nS	7-1	2V (1)
t_3	CLK Low Time	4.0		nS	7-1	0.8V (1)
t_4	CLK Fall Time	0.15	1.5	nS	7-1	(2.0V–0.8V) (1,5)
t_5	CLK Rise Time	0.15	1.5	nS	7-1	(0.8V–2.0V) (1,5)
t_{6a}	PWT, PCD, CACHE# Valid Delay	1.0	7.0	nS	7-2	
t_{6b}	AP Valid Delay	1.0	8.5	nS	7-2	
t_{6c}	BE0-7#, LOCK# Valid Delay	0.9	7.0	nS	7-2	4
t_{6d}	ADS#, ADSC#, D/C#, M/IO#, W/R#, SCYC, Valid Delay	0.8	7.0	nS	7-2	
t_{6e}	A3–A16 Valid Delay	0.5	6.3	nS	7-2	
t_{6f}	A17–A31 Valid Delay	0.6	6.3	nS	7-2	
t_7	ADS#, ADSC#, AP, A3-A31, PWT, PCD, BE0-7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Float Delay		10.0	nS	7-3	1
t_{8a}	APCHK#, IERR#, FERR# Valid Delay	1.0	8.3	nS	7-2	4
t_{8b}	PCHK# Valid Delay	1.0	7.0	nS	7-2	4
t_{9a}	BREQ, HLDA Valid Delay	1.0	8.0	nS	7-2	4
t_{9b}	SMIACT# Valid Delay	1.0	7.6	nS	7-2	
t_{10a}	HIT# Valid Delay	1.0	8.0	nS	7-2	
t_{10b}	HITM# Valid Delay	0.7	6.0	nS	7-2	
t_{11a}	PM0-1, BP0-3 Valid Delay	1.0	10.0	nS	7-2	

Table 7-10. Pentium® Processor AC Specifications for 60-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{11b}	PRDY Valid Delay	1.0	8.0	nS	7-2	
t_{12}	D0-D63, DP0-7 Write Data Valid Delay	1.3	7.5	nS	7-2	
t_{13}	D0-D63, DP0-3 Write Data Float Delay		10.0	nS	7-3	1
t_{14}	A5-A31 Setup Time	6.0		nS	7-4	26
t_{15}	A5-A31 Hold Time	1.0		nS	7-4	
t_{16a}	INV, AP Setup Time	5.0		nS	7-4	
t_{16b}	EADS# Setup Time	5.5		nS	7-4	
t_{17}	EADS#, INV, AP Hold Time	1.0		nS	7-4	
t_{18a}	KEN# Setup Time	5.0		nS	7-4	
t_{18b}	NA#, WB/WT# Setup Time	4.5		nS	7-4	
t_{19}	KEN#, WB/WT#, NA# Hold Time	1.0		nS	7-4	
t_{20}	BRDY#, BRDYC# Setup Time	5.0		nS	7-4	
t_{21}	BRDY#, BRDYC# Hold Time	1.0		nS	7-4	
t_{22}	AHOLD, BOFF# Setup Time	5.5		nS	7-4	
t_{23}	AHOLD, BOFF# Hold Time	1.0		nS	7-4	
t_{24}	BUSCHK#, EWBE#, HOLD, PEN# Setup Time	5.0		nS	7-4	
t_{25a}	BUSCHK#, EWBE#, PEN# Hold Time	1.0		nS	7-4	
t_{25b}	HOLD Hold Time	1.5		nS	7-4	
t_{26}	A20M#, INTR, STPCLK# Setup Time	5.0		nS	7-4	12, 16
t_{27}	A20M#, INTR, STPCLK# Hold Time	1.0		nS	7-4	13
t_{28}	INIT, FLUSH#, NMI, SMI#, IGNNE# Setup Time	5.0		nS	7-4	12, 16, 17
t_{29}	INIT, FLUSH#, NMI, SMI#, IGNNE# Hold Time	1.0		nS	7-4	13
t_{30}	INIT, FLUSH#, NMI, SMI#, IGNNE# Pulse Width, Async	2.0		CLKs		15, 17

Table 7-10. Pentium® Processor AC Specifications for 60-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{31}	R/S# Setup Time	5.0		nS	7-4	12, 16, 17
t_{32}	R/S# Hold Time	1.0		nS	7-4	13
t_{33}	R/S# Pulse Width, Async.	2.0		CLKs		15, 17
t_{34}	D0-D63, DP0-7 Read Data Setup Time	3.0		nS	7-4	
t_{35}	D0-D63, DP0-7 Read Data Hold Time	1.5		nS	7-4	
t_{36}	RESET Setup Time	5.0		nS	7-5	11, 12, 16
t_{37}	RESET Hold Time	1.0		nS	7-5	11, 13
t_{38}	RESET Pulse Width, V_{CC} & CLK Stable	15		CLKs	7-5	11, 17
t_{39}	RESET Active After V_{CC} & CLK Stable	1.0		mS	7-5	Power up
t_{40}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time	5.0		nS	7-5	12, 16, 17
t_{41}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Hold Time	1.0		nS	7-5	13
t_{42a}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time, Async.	2.0		CLKs		To RESET falling edge(16)
t_{42b}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#, BRDYC#, BUSCHK#) Hold Time, Async.	2.0		CLKs		To RESET falling edge(27)
t_{42c}	Reset Configuration Signals (BRDYC#, BUSCHK#) Setup Time, Async.	3.0		CLKs		To RESET falling edge(27)
t_{42d}	Reset Configuration Signal BRDYC# Hold Time, RESET driven synchronously	1.0		nS	7-5	To RESET falling edge(1,27)
t_{43a}	BF0, BF1, CPUTYP Setup Time	1.0		mS	7-5	To RESET falling edge(22)
t_{43b}	BF0, BF1, CPUTYP Hold Time	2.0		CLKs		To RESET falling edge(22)

Table 7-10. Pentium® Processor AC Specifications for 60-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{43c}	APICEN, BE4# Setup Time	2.0		CLKs		To RESET falling edge
t_{43d}	APICEN, BE4# Hold Time	2.0		CLKs		To RESET falling edge
t_{44}	TCK Frequency		16.0	MHz	7-1	
t_{45}	TCK Period	62.5		nS	7-1	
t_{46}	TCK High Time	25.0		nS	7-1	2V (1)
t_{47}	TCK Low Time	25.0		nS	7-1	0.8V (1)
t_{48}	TCK Fall Time		5.0	nS	7-1	(2.0V–0.8V) (1,8,9)
t_{49}	TCK Rise Time		5.0	nS	7-1	(0.8V–2.0V) (1,8,9)
t_{50}	TRST# Pulse Width	40.0		nS	7-7	Asynchronous (1)
t_{51}	TDI, TMS Setup Time	5.0		nS	7-6	7
t_{52}	TDI, TMS Hold Time	13.0		nS	7-6	7
t_{53}	TDO Valid Delay	2.5	20.0	nS	7-6	8
t_{54}	TDO Float Delay		25.0	nS	7-6	1, 8
t_{55}	All Non-Test Outputs Valid Delay	2.5	20.0	nS	7-6	3, 8, 10
t_{56}	All Non-Test Outputs Float Delay		25.0	nS	7-6	1, 3, 8, 10
t_{57}	All Non-Test Inputs Setup Time	5.0		nS	7-6	3, 7, 10
t_{58}	All Non-Test Inputs Hold Time	13.0		nS	7-6	3, 7, 10
APIC AC Specifications						
t_{60a}	PICCLK Frequency	2.0	16.66	MHz	7-1	
t_{60b}	PICCLK Period	60.0	500.0	nS	7-1	
t_{60c}	PICCLK High Time	15.0		nS	7-1	
t_{60d}	PICCLK Low Time	15.0		nS	7-1	
t_{60e}	PICCLK Rise Time	0.15	2.5	nS	7-1	
t_{60f}	PICCLK Fall Time	0.15	2.5	nS	7-1	

Table 7-10. Pentium® Processor AC Specifications for 60-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{60g}	PICD0-1 Setup Time	3.0		nS	7-4	To PICCLK
t_{60h}	PICD0-1 Hold Time	2.5		nS	7-4	To PICCLK
t_{60i}	PICD0-1 Valid Delay (LtoH)	4.0	38.0	nS	7-2	From PICCLK (28)
t_{60j}	PICD0-1 Valid Delay (HtoL)	4.0	22.0	nS	7-2	From PICCLK (28)
t_{61}	PICCLK Setup Time	5.0		nS		To CLK (30)
t_{62}	PICCLK Hold Time	2.0		nS		To CLK (30)
t_{63}	PICCLK Ratio (CLK/PICCLK)	4				31

NOTE: See notes following Table 7-13.

Table 7-11. Pentium® Processor Dual Processor Mode AC Specifications for 60-MHz Bus Operation

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{80a}	PBREQ#, PBGNT#, PHIT# Flight Time	0	2.0	nS		29
t_{80b}	PHITM# Flight Time	0	1.8	nS		29
t_{83a}	A5-A31 Setup Time	3.9		nS	7-4	18, 21, 26
t_{83b}	D/C#, W/R#, CACHE#, LOCK#, SCYC Setup Time	4.0		nS	7-4	18, 21
t_{83c}	ADS#, M/IO# Setup Time	6.0		nS	7-4	18, 21
t_{83d}	HIT#, HITM# Setup Time	6.0		nS	7-4	18
t_{83e}	HLDA Setup Time	6.0		nS	7-4	18
t_{84a}	CACHE#, HIT# Hold Time	1.0		nS	7-4	18, 21
t_{84b}	ADS#, D/C#, W/R#, M/IO#, A5-A31, HLDA, SCYC Hold Time	0.8		nS	7-4	18, 21
t_{84c}	LOCK# Hold Time	0.9		nS	7-4	18, 21
t_{84d}	HITM# Hold Time	0.7		nS	7-4	18, 21
t_{85}	DPEN# Valid Time		10.0	CLKs		18, 19, 23
t_{86}	DPEN# Hold Time	2.0		CLKs		18, 20, 23
t_{87}	APIC ID (BE0#–BE3#) Setup Time	2.0		CLKs	7-5	To RESET falling edge(23)
t_{88}	APIC ID (BE0#-BE3#) Hold Time	2.0		CLKs	7-5	From RESET falling edge(23)
t_{89}	D/P# Valid Delay	1.0	8.0	nS	7-2	Primary Processor Only

NOTE: See notes following Table 7-13.

7.4.3. AC Timing Tables for a 50-MHz Bus

Table 7-12. Pentium® Processor AC Specifications for 50-MHz Bus Operation

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	25.0	50.0	MHz	7-1	Max Core Freq = 100 MHz
t_{1a}	CLK Period	20.0	40.0	nS	7-1	
t_{1b}	CLK Period Stability		±250	pS	7-1	Adjacent Clocks (1,25)
t_2	CLK High Time	4.0		nS	7-1	2V,(1)
t_3	CLK Low Time	4.0		nS	7-1	0.8V, (1)
t_4	CLK Fall Time	0.15	1.5	nS	7-1	(2.0V–0.8V), (1,5)
t_5	CLK Rise Time	0.15	1.5	nS	7-1	(0.8V–2.0V), (1,5)
t_{6a}	PWT, PCD, CACHE# Valid Delay	1.0	7.0	nS	7-2	
t_{6b}	AP Valid Delay	1.0	8.5	nS	7-2	
t_{6c}	BE0-7#, LOCK# Valid Delay	0.9	7.0	nS	7-2	4
t_{6d}	ADS#, ADSC#, D/C#, M/IO#, W/R#, SCYC Valid Delay	0.8	7.0	nS	7-2	
t_{6e}	A3-A16 Valid Delay	0.5	7.0	nS	7-2	
t_{6f}	A17-A31 Valid Delay	0.6	7.0	nS	7-2	
t_7	ADS#, ADSC#, AP, A3-A31, PWT, PCD, BE0-7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Float Delay		10.0	nS	7-3	1
t_8	APCHK#, IERR#, FERR#, PCHK# Valid Delay	1.0	8.3	nS	7-2	4
t_{9a}	BREQ, HLDA, SMIACT# Valid Delay	1.0	8.0	nS	7-2	4
t_{10a}	HIT# Valid Delay	1.0	8.0	nS	7-2	
t_{10b}	HITM# Valid Delay	0.7	6.0	nS	7-2	
t_{11a}	PM0-1, BP0-3 Valid Delay	1.0	10.0	nS	7-2	

Table 7-12. Pentium® Processor AC Specifications for 50-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{11b}	PRDY Valid Delay	1.0	8.0	nS	7-2	
t_{12}	D0-D63, DP0-7 Write Data Valid Delay	1.3	8.5	nS	7-2	
t_{13}	D0-D63, DP0-3 Write Data Float Delay		10.0	nS	7-3	1
t_{14}	A5-A31 Setup Time	6.5		nS	7-4	26
t_{15}	A5-A31 Hold Time	1.0		nS	7-4	
t_{16a}	INV, AP Setup Time	5.0		nS	7-4	
t_{16b}	EADS# Setup Time	6.0		nS	7-4	
t_{17}	EADS#, INV, AP Hold Time	1.0		nS	7-4	
t_{18a}	KEN# Setup Time	5.0		nS	7-4	
t_{18b}	NA#, WB/WT# Setup Time	4.5		nS	7-4	
t_{19}	KEN#, WB/WT#, NA# Hold Time	1.0		nS	7-4	
t_{20}	BRDY#, BRDYC# Setup Time	5.0		nS	7-4	
t_{21}	BRDY#, BRDYC# Hold Time	1.0		nS	7-4	
t_{22}	BOFF# Setup Time	5.5		nS	7-4	
t_{22a}	AHOLD Setup Time	6.0		nS	7-4	
t_{23}	AHOLD, BOFF# Hold Time	1.0		nS	7-4	
t_{24}	BUSCHK#, EWBE#, HOLD, PEN# Setup Time	5.0		nS	7-4	
t_{25}	BUSCHK#, EWBE#, PEN# Hold Time	1.0		nS	7-4	
t_{25a}	HOLD Hold Time	1.5		nS	7-4	
t_{26}	A20M#, INTR, STPCLK# Setup Time	5.0		nS	7-4	12, 16
t_{27}	A20M#, INTR, STPCLK# Hold Time	1 .0		nS	7-4	13
t_{28}	INIT, FLUSH#, NMI, SMI#, IGNNE# Setup Time	5.0		nS	7-4	12, 16, 17
t_{29}	INIT, FLUSH#, NMI, SMI#, IGNNE# Hold Time	1.0		nS	7-4	13

Table 7-12. Pentium® Processor AC Specifications for 50-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{30}	INIT, FLUSH#, NMI, SMI#, IGNNE# Pulse Width, Async	2.0		CLKs		15, 17
t_{31}	R/S# Setup Time	5.0		nS	7-4	12, 16, 17
t_{32}	R/S# Hold Time	1.0		nS	7-4	13
t_{33}	R/S# Pulse Width, Async.	2. 0		CLKs		15, 17
t_{34}	D0-D63, DP0-7 Read Data Setup Time	3.8		nS	7-4	
t_{35}	D0-D63, DP0-7 Read Data Hold Time	1.5		nS	7-4	
t_{36}	RESET Setup Time	5.0		nS	7-5	11, 12, 16
t_{37}	RESET Hold Time	1.0		nS	7-5	11, 13
t_{38}	RESET Pulse Width, V_{CC} & CLK Stable	15		CLKs	7-5	11, 17
t_{39}	RESET Active After V_{CC} & CLK Stable	1.0		mS	7-5	Power up
t_{40}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time	5.0		nS	7-5	12, 16, 17
t_{41}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Hold Time	1.0		nS	7-5	13
t_{42a}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time, Async	2.0		CLKs		To RESET falling edge(16)
t_{42b}	Reset Configuration Signals (INIT, FLUSH#, FRCMC#, BRDYC#, BUSCHK#) Hold Time, Async	2.0		CLKs		To RESET falling edge(27)
t_{42c}	Reset Configuration Signals (BRDYC#, BUSCHK#) Setup Time, Async.	3.0		CLKs		To RESET falling edge(27)
t_{42d}	Reset Configuration Signal BRDYC# Hold Time, RESET driven synchronously	1.0		nS	7-5	To RESET falling edge(1,27)
t_{43a}	BF0, BF1, CPUTYP Setup Time	1.0		mS	7-5	To RESET falling edge(22)

Table 7-12. Pentium® Processor AC Specifications for 50-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{43b}	BF0, BF1, CPUTYP Hold Time	2.0		CLKs		To RESET falling edge(22)
t_{43c}	APICEN, BE4# Setup Time	2.0		CLKs		To RESET falling edge
t_{43d}	APICEN, BE4# Hold Time	2.0		CLKs		To RESET falling edge
t_{44}	TCK Frequency		16.0	MHz	7-1	
t_{45}	TCK Period	62.5		nS	7-1	
t_{46}	TCK High Time	25.0		nS	7-1	2V (1)
t_{47}	TCK Low Time	25.0		nS	7-1	0.8V (1)
t_{48}	TCK Fall Time		5.0	nS	7-1	(2.0V–0.8V) (1,8,9)
t_{49}	TCK Rise Time		5.0	nS	7-1	(0.8V–2.0V) (1,8,9)
t_{50}	TRST# Pulse Width	40.0		nS	7-7	Asynchronous (1)
t_{51}	TDI, TMS Setup Time	5.0		nS	7-6	7
t_{52}	TDI, TMS Hold Time	13.0		nS	7-6	7
t_{53}	TDO Valid Delay	2.5	20.0	nS	7-6	8
t_{54}	TDO Float Delay		25.0	nS	7-6	1, 8
t_{55}	All Non-Test Outputs Valid Delay	2.5	20.0	nS	7-6	3, 8, 10
t_{56}	All Non-Test Outputs Float Delay		25.0	nS	7-6	1, 3, 8, 10
t_{57}	All Non-Test Inputs Setup Time	5.0		nS	7-6	3, 7, 10
t_{58}	All Non-Test Inputs Hold Time	13.0		nS	7-6	3, 7, 10
APIC AC Specifications						
t_{60a}	PICCLK Frequency	2.0	16.66	MHz	7-1	
t_{60b}	PICCLK Period	60.0	500.0	nS	7-1	
t_{60c}	PICCLK High Time	15.0		nS	7-1	
t_{60d}	PICCLK Low Time	15.0		nS	7-1	
t_{60e}	PICCLK Rise Time	0.15	25	nS	7-1	

Table 7-12. Pentium® Processor AC Specifications for 50-MHz Bus Operation (Contd.)

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{60f}	PICCLK Fall Time	0.15	25	nS	7-1	
t_{60g}	PICD0-1 Setup Time	3.0		nS	7-4	To PICCLK
t_{60h}	PICD0-1 Hold Time	2.5		nS	7-4	To PICCLK
t_{60i}	PICD0-1 Valid Delay (LtoH)	4.0	38.0	nS	7-2	From PICCLK (28)
t_{60j}	PICD0-1 Valid Delay (HtoL)	4.0	22.0	nS	7-2	From PICCLK (28)
t_{61}	PICCLK Setup Time	5.0		nS		To CLK (30)
t_{62}	PICCLK Hold Time	2.0		nS		To CLK (30)
t_{63}	PICCLK Ratio (CLK/PICCLK)	4				31

NOTE: See notes following Table 7-13.

Table 7-13. Pentium® Processor Dual-Processor Mode AC Specifications for 50-MHz Bus Operation

See Table 7-2 for V_{CC} and T_{CASE} Specifications, C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{80a}	PBREQ#, PBGNT#, PHIT# Flight Time	0	2.0	nS	7-8	29
t_{80b}	PHITM# Flight Time	0	1.8	nS	7-8	29
t_{83a}	A5-A31 Setup Time	6.5		nS	7-4	18, 21, 26
t_{83b}	D/C#, W/R#, CACHE#, LOCK#, SCYC Setup Time	6.0		nS	7-4	18, 21
t_{83c}	ADS#, M/IO# Setup Time	8.0		nS	7-4	18, 21
t_{83d}	HIT#, HITM# Setup Time	8.0		nS	7-4	18
t_{83e}	HLDA Setup Time	6.0		nS	7-4	18
t_{84a}	CACHE#, HIT# Hold Time	1.0		nS	7-4	18, 21
t_{84b}	ADS#, D/C#, W/R#, M/IO#, A5-A31, HLDA, SCYC Hold Time	0.8		nS	7-4	18, 21
t_{84c}	LOCK# Hold Time	0.9		nS	7-4	18, 21
t_{84d}	HITM# Hold Time	0.7		nS	7-4	18, 21
t_{85}	DPEN# Valid Time		10.0	CLKs		18, 19, 23
t_{86}	DPEN# Hold Time	2.0		CLKs		18, 20, 23
t_{87}	APIC ID (BE0#-BE3#) Setup Time	2.0		CLKs	7-5	To RESET falling edge(23)
t_{88}	APIC ID (BE0#-BE3#) Hold Time	2.0		CLKs	7-5	From RESET falling edge(23)
t_{89}	D/P# Valid Delay	1.0	8.0	nS	7-2	Primary Processor Only

NOTES:

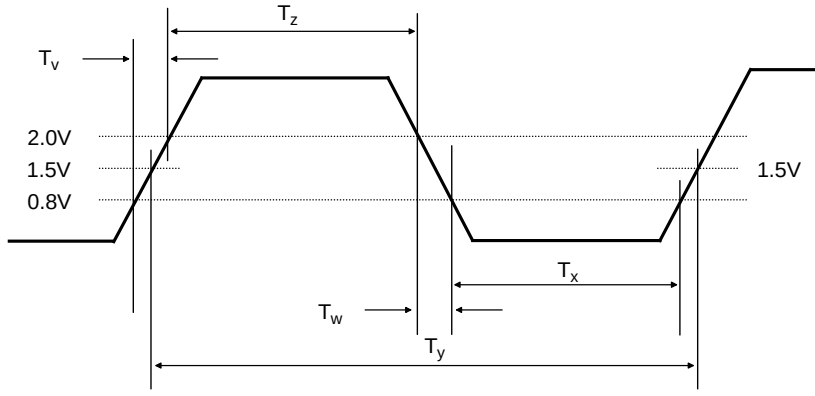
Notes 2, 6, and 14 are general and apply to all standard TTL signals used with the Pentium® processor family.

1. Not 100% tested. Guaranteed by design/characterization.
2. TTL input test waveforms are assumed to be 0 to 3V transitions with 1V/nS rise and fall times.
3. Non-test outputs and inputs are the normal output or input signals (besides TCK, TRST#, TDI, TDO, and TMS). These timings correspond to the response of these signals due to boundary scan operations.
4. APCHK#, FERR#, HLDA, IERR#, LOCK#, and PCHK# are glitch-free outputs. Glitch-free signals monotonically transition without false transitions (i.e., glitches).
5. 0.8V/ns ≤ CLK input rise/fall time ≤ 8V/ns.
6. 0.3V/ns ≤ input rise/fall time ≤ 5V/ns.
7. Referenced to TCK rising edge.
8. Referenced to TCK falling edge.

9. 1 ns can be added to the maximum TCK rise and fall times for every 10 MHz of frequency below 33 MHz.
10. During probe mode operation, do not use the boundary scan timings (t_{55-58}).
11. FRCMC# should be tied to V_{CC} (high) to ensure proper operation of the Pentium processor 75/90/100/120/133/150/166/200 as a primary processor.
12. Setup time is required to guarantee recognition on a specific clock. The Pentium processor must meet this specification for dual processor operation for the FLUSH# and RESET signals.
13. Hold time is required to guarantee recognition on a specific clock. The Pentium processor must meet this specification for dual processor operation for the FLUSH# and RESET signals.
14. All TTL timings are referenced from 1.5V.
15. To guarantee proper asynchronous recognition, the signal must have been de-asserted (inactive) for a minimum of 2 clocks before being returned active and must meet the minimum pulse width.
16. This input may be driven asynchronously. However, when operating two processors in dual processing mode, FLUSH# and RESET must be asserted synchronously to both processors.
17. When driven asynchronously, RESET, NMI, FLUSH#, R/S#, INIT, and SMI# must be de-asserted (inactive) for a minimum of 2 clocks before being returned active.
18. Timings are valid only when dual processor is present.
19. Maximum time DPEN# is valid from rising edge of RESET.
20. Minimum time DPEN# is valid after falling edge of RESET.
21. The D/C#, M/IO#, W/R#, CACHE#, and A5-A31 signals are sampled only on the CLK that ADS# is active.
22. BF0, BF1 and CPUTYP should be strapped to 3.3V or V_{SS}.
23. RESET is synchronous in dual processing mode and functional redundancy checking mode. All signals which have a setup or hold time with respect to a falling or rising edge of RESET in UP mode, should be measured with respect to the first processor clock edge in which RESET is sampled either active or inactive in dual processing and functional redundancy checking modes.
24. The PHIT# and PHITM# signals operate at the core frequency.
25. These signals are measured on the rising edge of adjacent CLKs at 1.5V. To ensure a 1:1 relationship between the amplitude of the input jitter and the internal and external clocks, the jitter frequency spectrum should not have any power spectrum peaking between 500 KHz and 1/3 of the CLK operating frequency. The amount of jitter present must be accounted for as a component of CLK skew between devices. The internal clock generator requires a constant frequency CLK input within ± 250 ps. Therefore, the CLK input cannot be changed dynamically.
26. In dual processing mode, timing t_{14} is replaced by t_{83a}. Timing t_{14} is required for external snooping (e.g., address setup to the CLK in which EADS# is sampled active) in both uniprocessor and dual processor modes.
27. BRDYC# and BUSCHK# are used as reset configuration signals to select buffer size.
28. This assumes an external pullup resistor to V_{CC} and a lumped capacitive load. The pullup resistor must be between 300 ohms and 1k ohms, the capacitance must be between 20 pF and 240 pF, and the RC product must be between 6ns and 36ns. V_{OL} for PICD0-1 is 0.55V.
29. This is a flight time specification, that includes both flight time and clock skew. The flight time is the time from where the unloaded driver crosses 1.5V (50% of min V_{CC}), to where the receiver crosses the 1.5V level (50% of min V_{CC}). See Figure 7-8. The minimum flight time minus the clock skew must be greater than zero.
30. This is for the Lock Step operation of the component only. This guarantees that APIC interrupts will be recognized on specific clocks to support two processors running in a Lock Step fashion, including FRC mode. FRC on the APIC pins is not supported, but mismatches on these pins will result in a mismatch on other pins of the processor.
31. The CLK to PICCLK ratio for Lock Step operation has to be an integer and the ratio (CLK/PICCLK) cannot be smaller than 4.

* Each valid delay is specified for a 0 pF load. The system designer should use I/O buffer models to account for signal flight time delays.

7.4.4. Timing Waveforms



$T_v = t_5, t_{49}, t_{60e}$

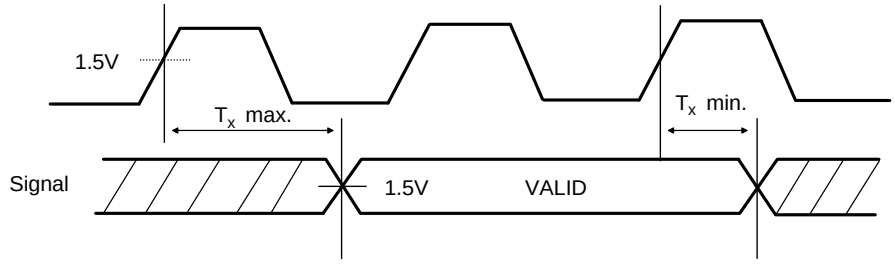
$T_w = t_4, t_{48}, t_{60f}$

$T_x = t_3, t_{47}, t_{60e}$

$T_y = t_1, t_{45}, t_{60b}$

$T_z = t_2, t_{46}, t_{60c}$

Figure 7-1. Clock Waveform



$T_x = t_6, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{60i}, t_{80a}, t_{89}$

Figure 7-2. Valid Delay Timings

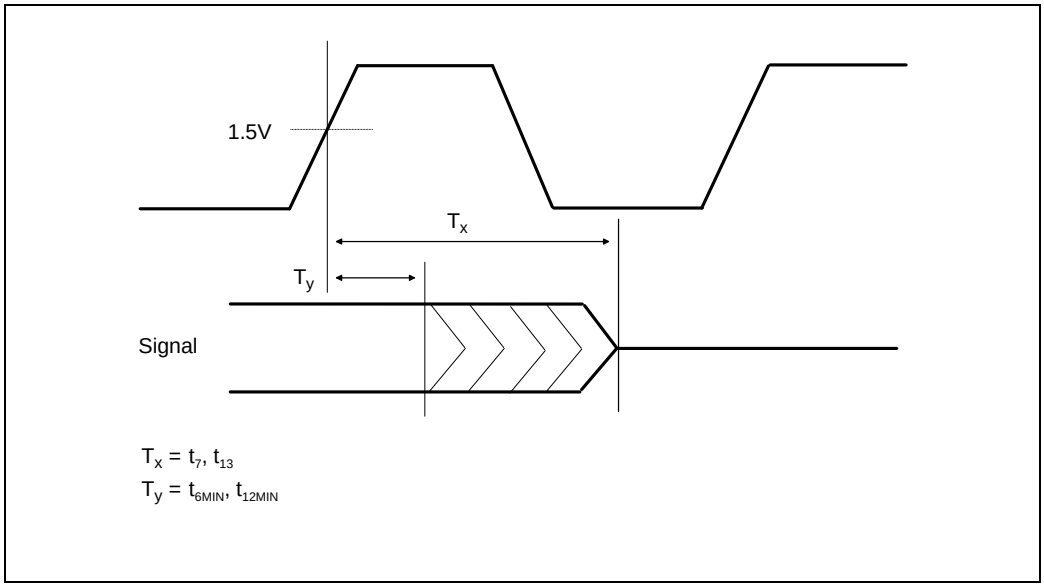


Figure 7-3. Float Delay Timings

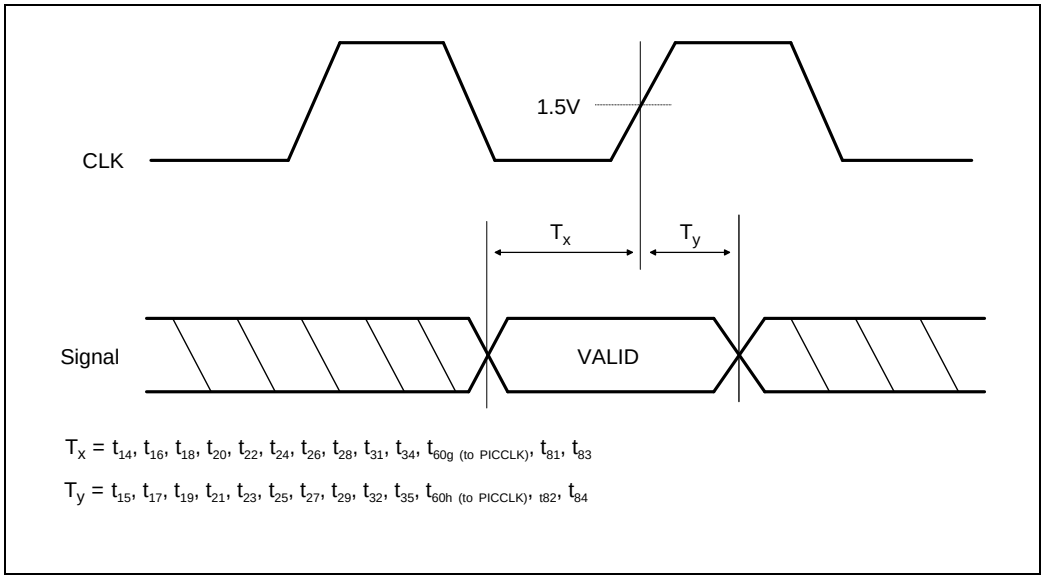
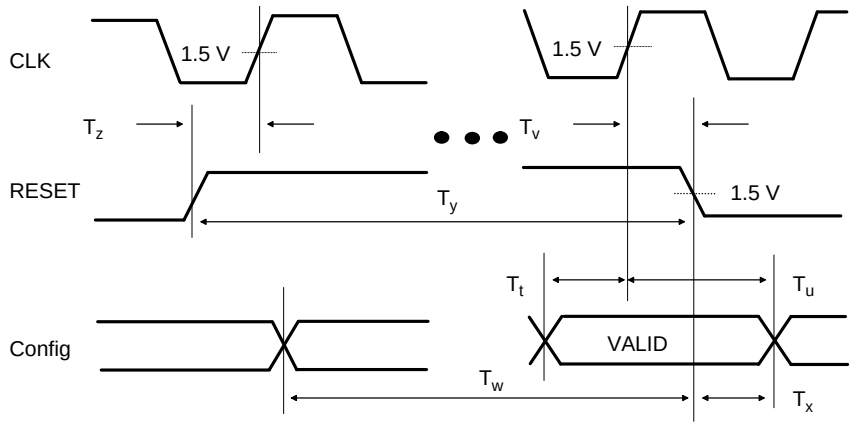


Figure 7-4. Setup and Hold Timings



$T_t = t_{40}$

$T_u = t_{41}$

$T_v = t_{37}$

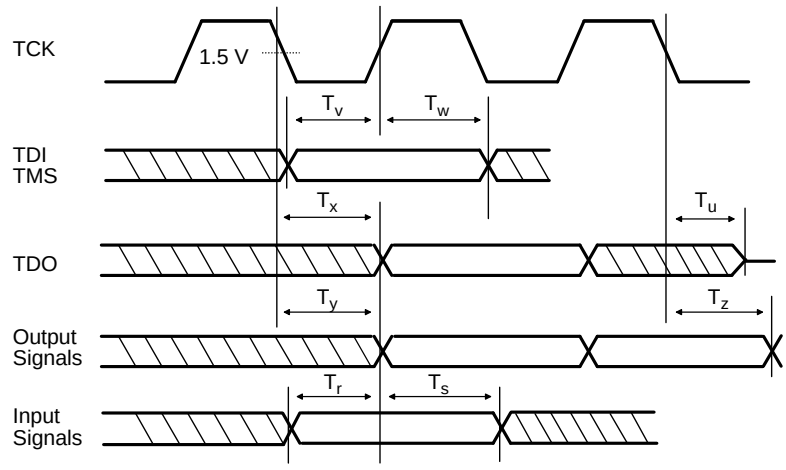
$T_w = t_{42}, t_{43a}, t_{43c}, t_{87}$

$T_x = t_{43b}, t_{43d}, t_{88}$

$T_y = t_{38}, t_{39}$

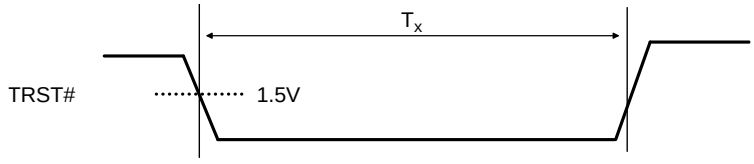
$T_z = t_{36}$

Figure 7-5. Reset and Configuration Timings



$T_r = t_{57}$
$T_s = t_{58}$
$T_u = t_{54}$
$T_v = t_{51}$
$T_w = t_{52}$
$T_x = t_{53}$
$T_y = t_{55}$
$T_z = t_{56}$

Figure 7-6. Test Timings



$T_x = t_{50}$

Figure 7-7. Test Reset Timings

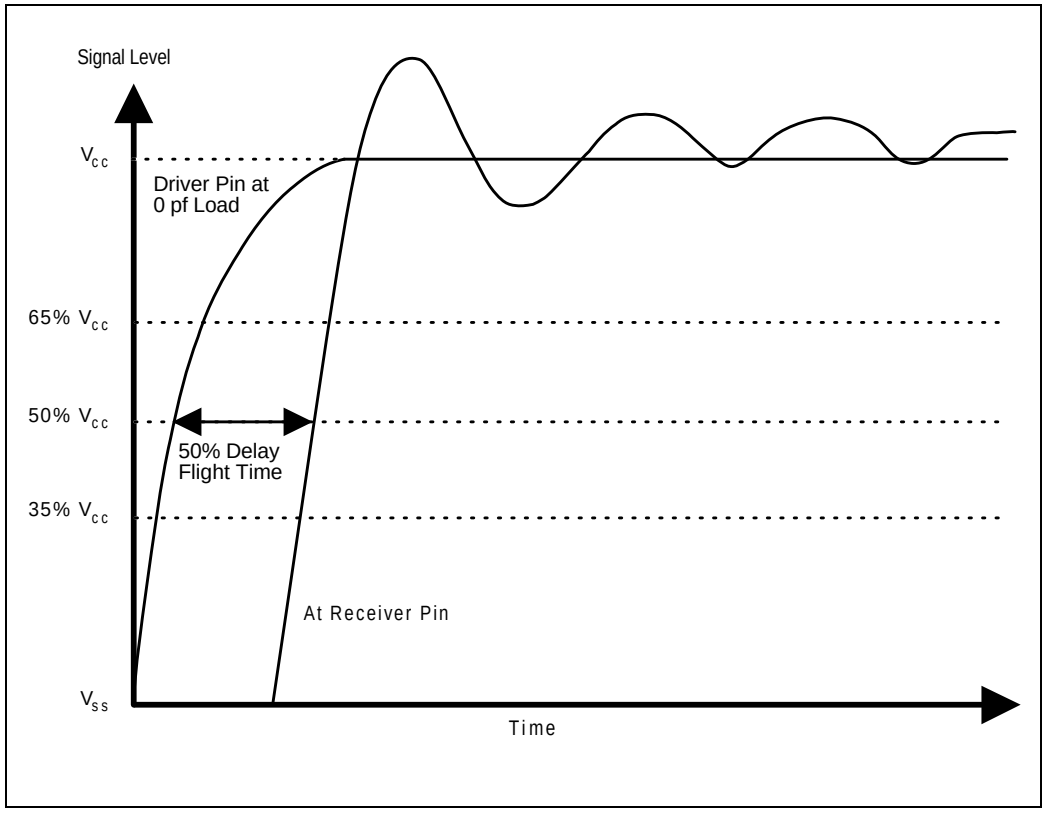


Figure 7-8. Vcc Measurement of Flight Time

8

I/O Buffer Models

CHAPTER 8
I/O BUFFER MODELS

This Chapter describes the 3.3V I/O buffer models of the Pentium processor.

The first order I/O buffer model is a simplified representation of the complex input and output buffers used in the Pentium processor family. Figures 8-1 and 8-2 show the structure of the input buffer model and Figure 8-3 shows the output buffer model. Tables 8-1 and 8-2 show the parameters used to specify these models.

Although simplified, these buffer models will accurately model flight time and signal quality. For these parameters, there is very little added accuracy in a complete transistor model.

The following two models represent the input buffer models. The first model, Figure 8-1, represents all of the input buffers of the Pentium processor except for a special group of input buffers. The second model, Figure 8-2, represents these special buffers. These buffers are the inputs: AHOLD, EADS#, KEN#, WB/WT#, INV, NA#, EWBE#, BOFF#, CLK, and PICCLK.

The Pentium processor (75/90/100/120/133/150/166/200) supports 5V tolerant buffers on the CLK and PICCLK pins. It is important to note that all inputs of the Pentium processor with MMX technology are 3.3V tolerant only. The CLK and PICCLK pins are not 5V tolerant on the Pentium processor with MMX technology.

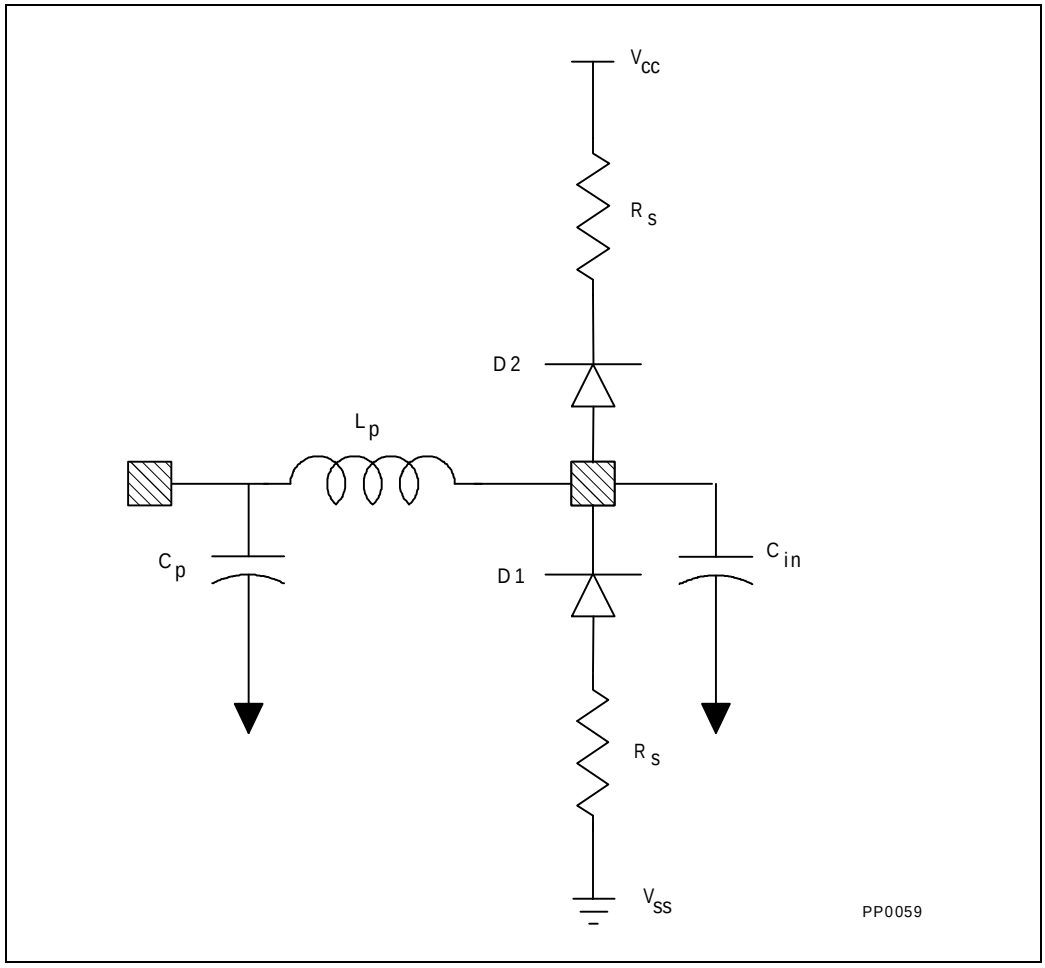


Figure 8-1. Input Buffer Model, Except Special Group

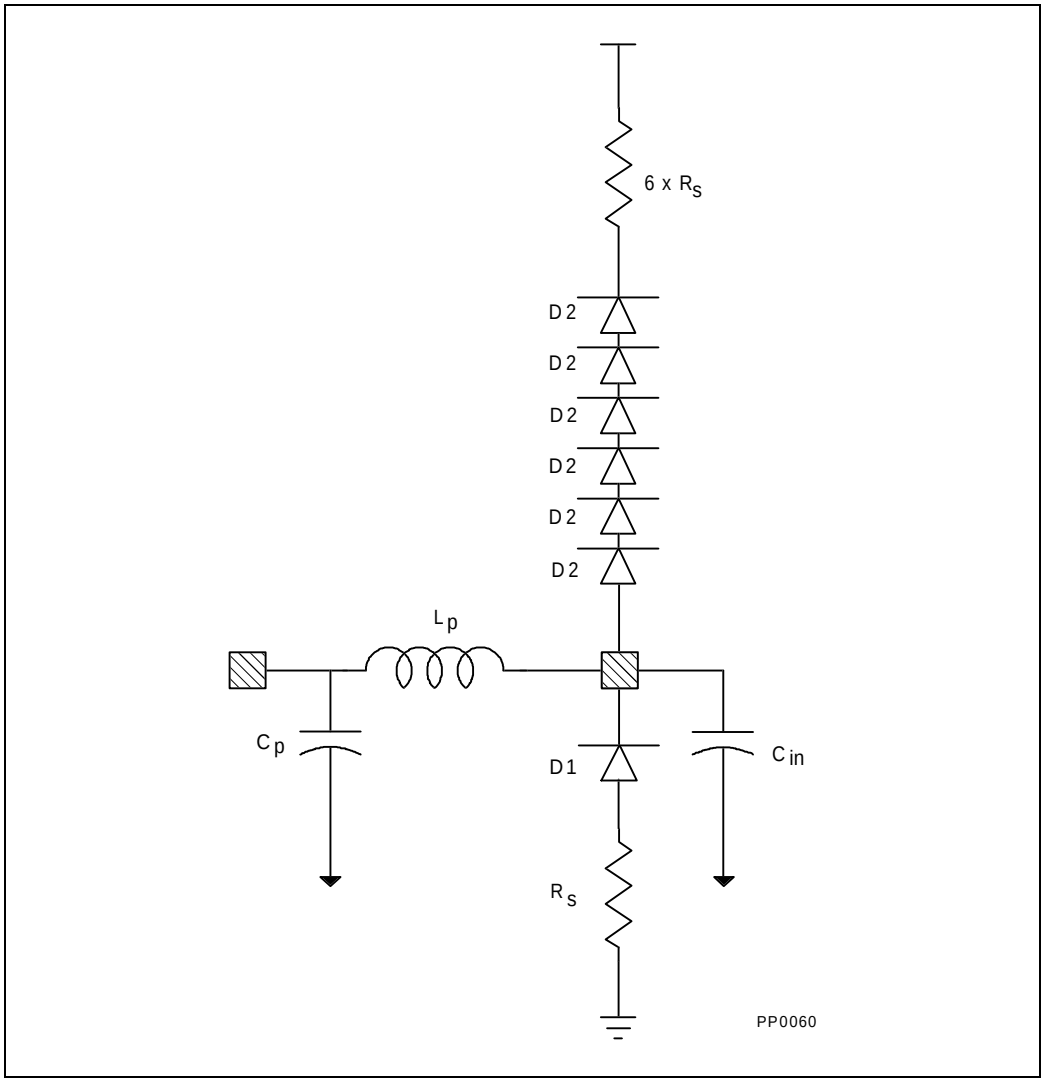


Figure 8-2. Input Buffer Model for Special Group

Table 8-1. Parameters Used in the Specification of the First Order Input Buffer Model

Parameter	Description
Cin	Minimum and Maximum value of the capacitance of the input buffer model.
Lp	Minimum and Maximum value of the package inductance.
Cp	Minimum and Maximum value of the package capacitance.
Rs	Diode Series Resistance
D1, D2	Ideal Diodes

Figure 8-3 shows the structure of the output buffer model. This model is used for all of the output buffers of the Pentium processor.

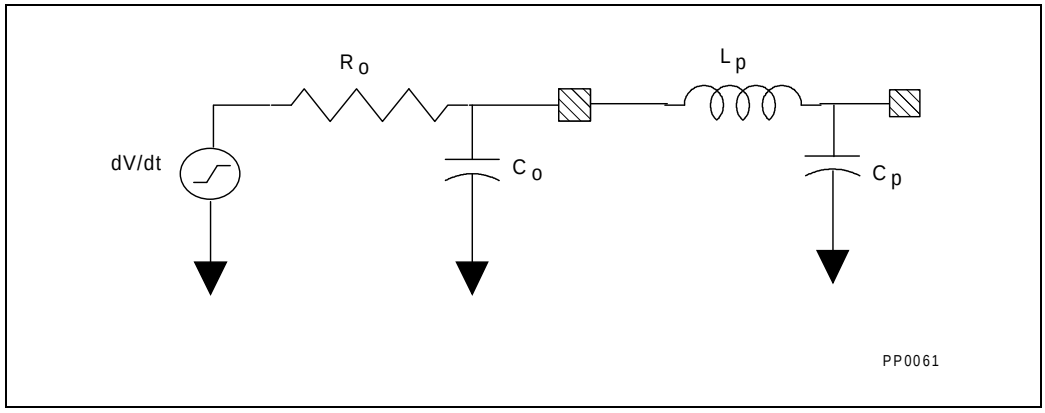


Figure 8-3. First Order Output Buffer Model

Table 8-2. Parameters Used in the Specification of the First Order Output Buffer Model

Parameter	Description
dV/dt	Minimum and maximum value of the rate of change of the open circuit voltage source used in the output buffer model.
Ro	Minimum and maximum value of the output impedance of the output buffer model.
Co	Minimum and Maximum value of the capacitance of the output buffer model.
Lp	Minimum and Maximum value of the package inductance.
Cp	Minimum and Maximum value of the package capacitance.

In addition to the input and output buffer parameters, input protection diode models are provided for added accuracy. These diodes have been optimized to provide ESD protection

and provide some level of clamping. Although the diodes are not required for simulation, it may be more difficult to meet specifications without them.

Note however, some signal quality specifications require that the diodes be removed from the input model. The series resistors (Rs) are a part of the diode model. Remove these when removing the diodes from the input model.

8.1. BUFFER MODEL PARAMETERS

This section gives the parameters for each Pentium processor input, output, and bi-directional signals, as well as the settings for the configurable buffers.

In dual processor mode, a few signals change from output signals to I/O signals. These signals are: ADS#, M/IO#, D/C#, W/R#, LOCK#, CACHE#, SCYC, HLDA, HIT#, and HITM#. When simulating these signals use the correct operation of the buffer while in DP mode.

Some pins on the Pentium processor have selectable buffer sizes to allow for faster switching of the buffer in heavily loaded environments. The buffer selection is done through the setting of configuration pins at power on RESET. Once selected, these cannot be changed without a power on RESET. The BUSCHK# and BRDYC# pins are used to select the different buffer size. All configurable pins get set to the selected buffer size. There is no selection for specific signal groups to get specific buffers. Keep in mind that the largest buffer size is not always the best selection especially in a lightly loaded environment. AC timing and signal quality simulations should be done to ensure that the buffers used meet required timing and signal quality specifications for the components that will be used in the specific board design.

The pins with selectable buffer sizes use the configurable output buffer EB2. Table 8-3 shows the drive level required at falling edge of RESET, to select the buffer strength. Once selected, the buffer size cannot be changed without a power on RESET. The buffer sizes selected should be the appropriate size required, otherwise AC timings might not be met, or too much overshoot and ringback may occur. There are no other selection choices, all the configurable buffers get set to the same size at the same time.

Table 8-3 shows the proper settings on BRDYC# and BUSCHK# for proper buffer size selection.

Table 8-3. Buffer Selection Chart

Environment	BRDYC#	BUSCHK#	Buffer Selection
Typical Stand Alone Component	1	X	EB2
Loaded Component	0	1	EB2A
Heavily Loaded Component	0	0	EB2B

NOTE:
X is a “DON’T CARE” (0 or 1).

Please refer to Table 8-4 for the groupings of the buffers.

Table 8-4. Signal to Buffer Type

Signals	Type	Driver Buffer Type	Receiver Buffer Type
CLK	I		ER0
A20M#, AHOLD, BF[1:0], BOFF#, BRDY#, BRDYC#, BUSCHK#, EADS#, EWBE#, FLUSH#, FRCMC#[2], HOLD, IGNNE#, INIT, INTR, INV, KEN#, NA#, NMI, PEN#, PICCLK, R/S#, RESET, SMI#, STPCLK#, TCK, TDI, TMS, TRST#, WB/WT#	I		ER1
ADSC#, APCHK#, BE[7:5]#, BP[3:2], BREQ, FERR#, IERR#, PCD, PCHK#, PM0/BP0, PM1/BP1, PRDY, PWT, SMIACT#, TDO, D/P#	O	ED1	
A[31:21], AP, BE[4:0]#, CACHE#, D/C#, D[63:0], DP[7:0], HLDA, LOCK#, M/IO#, PBGNT#, PBREQ#, PHIT#, PHITM#, SCYC	I/O	EB1	EB1
A[20:3], ADS#, HITM#, W/R#	I/O	EB2/A/B	EB2
HIT#	I/O	EB3	EB3
PICD0, PICD1	I/O	EB4	EB4

NOTES:

1. VCC2DET# has no buffer model — it is simply a short to V_{SS} on the Pentium processor with MMX technology. This pin is an INC on the Pentium processor (75/90/100/120/133/150/166/200)
2. FRCMC# is defined only for the Pentium processor (75/90/100/120/133/150/166/200).

The input, output and bi-directional buffers values are listed in Table 8-5. Table 8-5 contains listings for all three types, do not get them confused during simulation. When a bi-directional pin is operating as an input, just use the Cin, Cp and Lp values, if it is operating as a driver use all the data parameters.

Table 8-5. Input, Output and Bi-directional Buffer Model Parameters

Buffer Type	Transition	dV/dt (V/nsec)		Ro (Ohms)		Cp (pF)		Lp (nH)		Co/Cin (pF)	
		min	max	min	max	min	max	min	max	min	max
ER0	Rising					3.0	5.0	4.0	7.2	0.8	1.2
(input)	Falling					3.0	5.0	4.0	7.2	0.8	1.2
ER1	Rising					1.1	6.1	4.7	15.3	0.8	1.2
(input)	Falling					1.1	6.1	4.7	15.3	0.8	1.2
ED1	Rising	3/3.0	3.7/0.9	21.6	53.1	1.1	8.2	4.0	17.7	2.0	2.6
(output)	Falling	3/2.8	3.7/0.8	17.5	50.7	1.1	8.2	4.0	17.7	2.0	2.6
EB1	Rising	3/3.0	3.7/0.9	21.6	53.1	1.3	8.7	4.0	18.7	2.0	2.6
(bidir)	Falling	3/2.8	3.7/0.8	17.5	50.7	1.3	8.7	4.0	18.7	2.0	2.6
EB2	Rising	3/3.0	3.7/0.9	21.6	53.1	1.3	8.3	4.4	16.7	9.1	9.7
(bidir)	Falling	3/2.8	3.7/0.8	17.5	50.7	1.3	8.3	4.4	16.7	9.1	9.7
EB2A	Rising	3/2.4	3.7/0.9	10.1	22.4	1.3	8.3	4.4	16.7	9.1	9.7
(bidir)	Falling	3/2.4	3.7/0.9	9.0	21.2	1.3	8.3	4.4	16.7	9.1	9.7
EB2B	Rising	3/1.8	3.7/0.7	5.5	12.9	1.3	8.3	4.4	16.7	9.1	9.7
(bidir)	Falling	3/1.8	3.7/0.7	4.6	12.3	1.3	8.3	4.4	16.7	9.1	9.7
EB3	Rising	3/3.0	3.7/0.9	21.6	53.1	1.9	7.5	9.9	14.3	3.3	3.9
(bidir)	Falling	3/2.8	3.7/0.8	17.5	50.7	1.9	7.5	9.9	14.3	3.3	3.9
EB4	Rising	3/3.0	3.7/0.9	100K*	100K*	2.0	6.9	5.8	14.6	5.0	7.0
(bidir)	Falling	3/2.8	3.7/0.8	17.5	50.7	2.0	6.9	5.8	14.6	5.0	7.0

* The buffer is an open drain. For simulation purposes it should be modeled by a very large internal resistor with an additional external pull-up.

Table 8-6. Input Buffer Model Parameters: D (Diodes)

Symbol	Parameter	D1	D2
IS	Saturation Current	1.4e-14A	2.78e-16A
N	Emission Coefficient	1.19	1.00
RS	Series Resistance	6.5 ohms	6.5 ohms
TT	Transit Time	3 ns	6 ns
VJ	PN Potential	0.983V	0.967V
CJ0	Zero Bias PN Capacitance	0.281 pF	0.365 pF
M	PN Grading Coefficient	0.385	0.376

8.2. SIGNAL QUALITY SPECIFICATIONS

Signals driven by the system into the Pentium processor must meet signal quality specifications to guarantee that the components read data properly and to ensure that incoming signals do not affect the reliability of the component. There are two signal quality parameters: Ringback and Settling Time. Reference Section 8.2.3 for the CLK/PICCLK signal quality specification for the Pentium processor with MMX technology.

8.2.1. Ringback

Excessive ringback can contribute to long-term reliability degradation of the Pentium processor, and can cause false signal detection. Ringback is simulated at the input pin of a component using the input buffer model. Ringback can be simulated with or without the diodes that are in the input buffer model.

Ringback is the absolute value of the maximum voltage at the receiving pin below V_{CC} (or above V_{SS}) relative to V_{CC} (or V_{SS}) level after the signal has reached its maximum voltage level. The input diodes are assumed present.

Maximum Ringback on Inputs = 0.8V
(with diodes)

If simulated without the input diodes, follow the Maximum Overshoot/Undershoot specification. By meeting the overshoot/undershoot specification, the signal is guaranteed not to ringback excessively.

If simulated with the diodes present in the input model, follow the maximum ringback specification.

Overshoot (Undershoot) is the absolute value of the maximum voltage above V_{CC} (below V_{SS}). The guideline assumes the absence of diodes on the input.

- Maximum Overshoot/Undershoot on 3.3V Pentium processor Inputs (including CLK and PICCLK) = 1.4V above $V_{CC}3$ (without diodes)

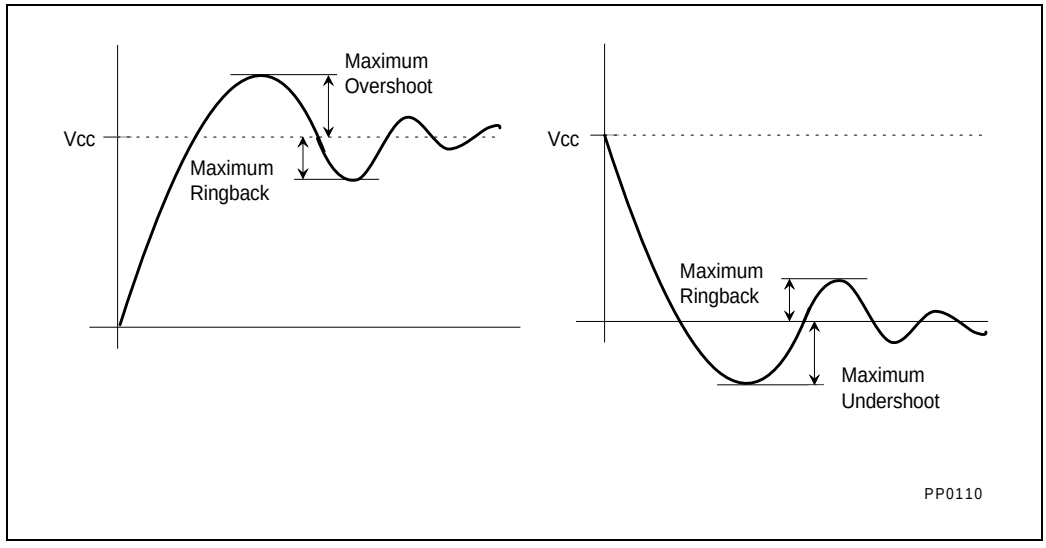


Figure 8-4. Overshoot/Undershoot and Ringback Guidelines

8.2.2. Settling Time

The settling time is defined as the time a signal requires at the receiver to settle within 10% of V_{CC} or V_{SS}. Settling time is the maximum time allowed for a signal to reach within 10% of its final value.

Most available simulation tools are unable to simulate settling time so that it accurately reflects silicon measurements. On a physical board, second-order effects and other effects serve to dampen the signal at the receiver. Because of all these concerns, settling time is a recommendation or a tool for layout tuning and not a specification.

Settling time is simulated at the slow corner, to make sure that there is no impact on the flight times of the signals if the waveform has not settled. Settling time may be simulated with the diodes included or excluded from the input buffer model. If diodes are included, settling time recommendation will be easier to meet.

Although simulated settling time has not shown good correlation with physical, measured settling time, settling time simulations can still be used as a tool to tune layouts.

Use the following procedure to verify board simulation and tuning with concerns for settling time.

1. Simulate settling time at the slow corner for a particular signal.
2. If settling time violations occur, simulate signal trace with DC diodes in place at the receiver pin. The DC diode behaves almost identically to the actual (non-linear) diode on the part as long as excessive overshoot does not occur.
3. If settling time violations still occur, simulate flight times for 5 consecutive cycles for that particular signal.
4. If flight time values are consistent over the 5 simulations, settling time should not be a concern. If however, flight times are not consistent over the 5 simulations, tuning of the layout is required.
5. Note that, for signals that are allocated 2 cycles for flight time, the recommended settling time is doubled.

Maximum Settling Time to within 10% of V_{CC} is: 12.5ns @66 MHz

14.2ns @60 MHz

17.5ns @50 MHz

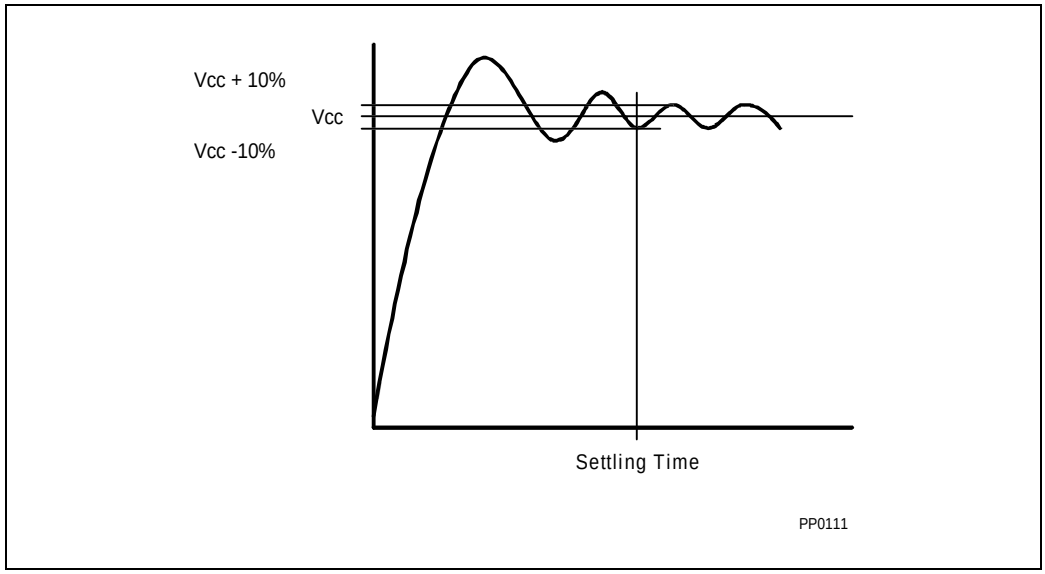


Figure 8-5. Settling Time

8.2.3. CLK/PICCLK Signal Quality Specification for the Pentium® Processor with MMX™ Technology

The maximum overshoot, maximum undershoot, overshoot threshold duration, undershoot threshold duration, and maximum ringback specifications for CLK/PICCLK are described below:

MAXIMUM OVERSHOOT AND MAXIMUM UNDERSHOOT SPECIFICATION: The maximum overshoot of the CLK/PICCLK signals should not exceed $V_{CC}3$,nominal + 0.9V. The maximum undershoot of the CLK/PICCLK signals must not drop below -0.9V.

OVERSHOOT THRESHOLD DURATION SPECIFICATION: The overshoot threshold duration is defined as the sum of all time during which the CLK/PICCLK signal is above $V_{CC}3$,nominal + 0.5V within a single clock period. The overshoot threshold duration must not exceed 20% of the period.

UNDERSHOOT THRESHOLD DURATION SPECIFICATION: The undershoot threshold duration is defined as the sum of all time during which the CLK/PICCLK signal is below -0.5V within a single clock period. The undershoot threshold duration must not exceed 20% of the period.

MAXIMUM RINGBACK SPECIFICATION: The maximum ringback of CLK/PICCLK associated with their high states (overshoot) must not drop below $V_{CC}3$ - 0.8V as shown in Figure 8-7. Similarly, the maximum ringback of CLK/PICCLK associated with their low states (undershoot) must not exceed 0.8V as shown in Figure 8-9.

Refer to Table 8-7 and Table 8-8 for a summary of the clock overshoot and undershoot specifications for the Pentium processor with MMX technology.

Table 8-7. Overshoot Specification Summary

Specification Name	Value	Units	Notes
Threshold Level	$V_{CC}3$,nominal + 0.5	V	(1) (2)
Maximum Overshoot Level	$V_{CC}3$,nominal + 0.9	V	(1) (2)
Maximum Threshold Duration	20% of clock period above threshold voltage	nS	(2)
Maximum Ringback	$V_{CC}3$,nominal - 0.8	V	(1) (2)

NOTES:

1. $V_{CC}3$, nominal refers to the voltage measured at the bottom side of the $V_{CC}3$ pins. See Section 7.1.2.1.1 for details.
2. See Figure 8-6 and Figure 8-7.

Table 8-8. Undershoot Specification Summary

Specification Name	Value	Units	Notes
Threshold Level	-0.5	V	(1)
Minimum Undershoot Level	-0.9	V	(1)
Maximum Threshold Duration	20% of clock period below threshold voltage	nS	(1)
Maximum Ringback	0.8	V	(1)

NOTE:

1. See Figure 8-8 and Figure 8-9.

8.2.3.1. CLOCK SIGNAL MEASUREMENT METHODOLOGY

The waveform of the clock signals should be measured at the bottom side of the processor pins using an oscilloscope with a 3 dB bandwidth of at least 20 MHz (100 MS/s digital sampling rate). There should be a short isolation ground lead attached to a processor pin on the bottom side of the board. An 1 MOhm probe with loading of less than 1 pF (e.g., Tektronics 6243 or Tektronics 6245) is recommended. The measurement should be taken at the CLK (AK18) and PICCLK (H34) pins and their nearest V_{SS} pins (AM18 and H36, respectively).

MAXIMUM OVERSHOOT, MAXIMUM UNDERSHOOT AND MAXIMUM RINGBACK SPECIFICATIONS: The display should show continuous sampling (e.g., infinite persistence) of the waveform at 500 mV/div and 5 nS/div (for CLK) or 20 nS/div (for PICCLK) for a recommended duration of approximately five seconds. Adjust the vertical position to measure the maximum overshoot and associated ringback with the largest possible granularity. Similarly, readjust the vertical position to measure the maximum undershoot and associated ringback. There is no allowance for crossing the maximum overshoot, maximum undershoot or maximum ringback specifications.

OVERSHOOT THRESHOLD DURATION SPECIFICATION: A snapshot of the clock signal should be taken at 500 mV/div and 500 pS/div (for CLK) or 2 nS/div (for PICCLK). Adjust the vertical position and horizontal offset position to view the threshold duration. The overshoot threshold duration is defined as the sum of all time during which the clock signal is above $V_{CC}3$,nominal + 0.5V within a single clock period. The overshoot threshold duration must not exceed 20% of the period.

UNDERSHOOT THRESHOLD DURATION SPECIFICATION: A snapshot of the clock signal should be taken at 500 mV/div and 500 pS/div (for CLK) or 2 nS/div (for PICCLK). Adjust the vertical position and horizontal offset position to view the threshold duration. The undershoot threshold duration is defined as the sum of all time during which the clock signal is below -0.5V within a single clock period. The undershoot threshold duration must not exceed 20% of the period.

These overshoot and undershoot specifications are illustrated graphically in Figure 8-6 to Figure 8-9.

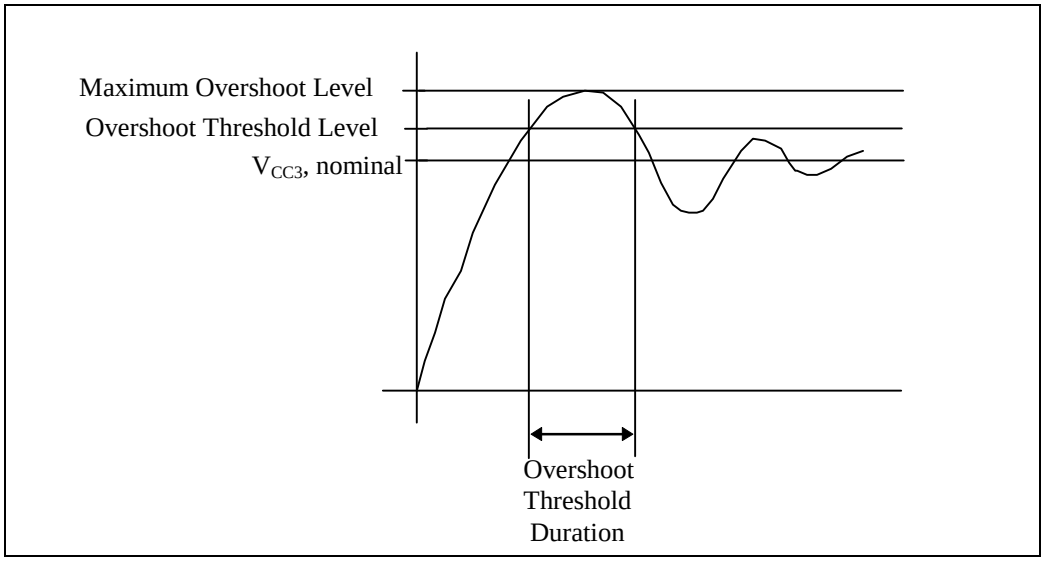


Figure 8-6. Maximum Overshoot Level, Overshoot Threshold Level, and Overshoot Threshold Duration

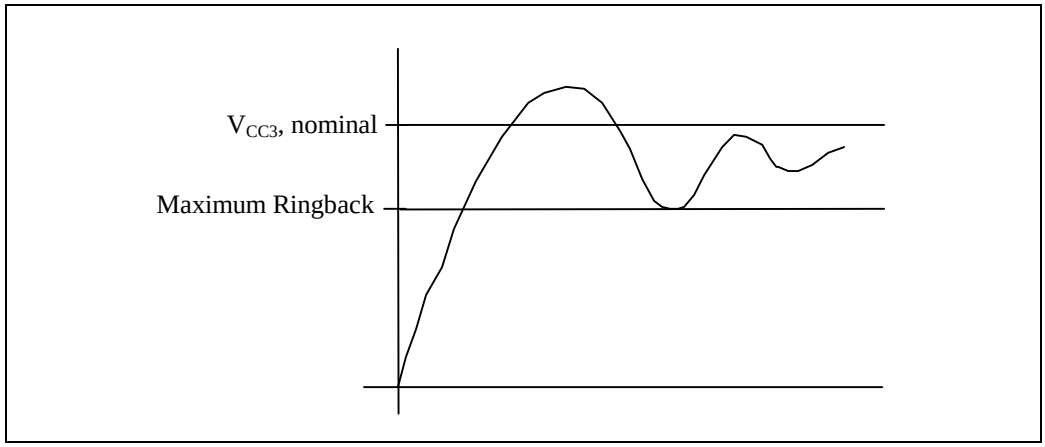


Figure 8-7. Maximum Ringback Associated with the Signal High State

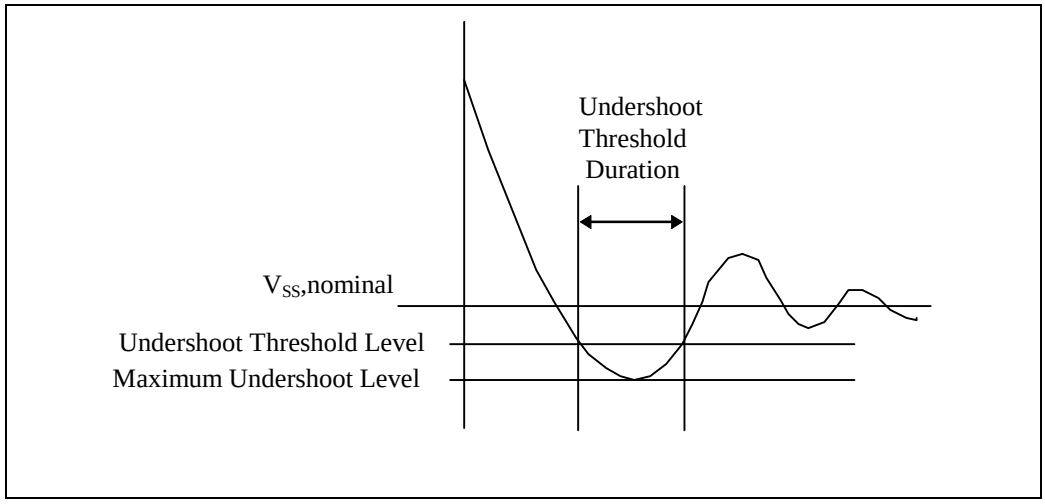


Figure 8-8. Maximum Undershoot Level, Undershoot Threshold Level, and Undershoot Threshold Duration

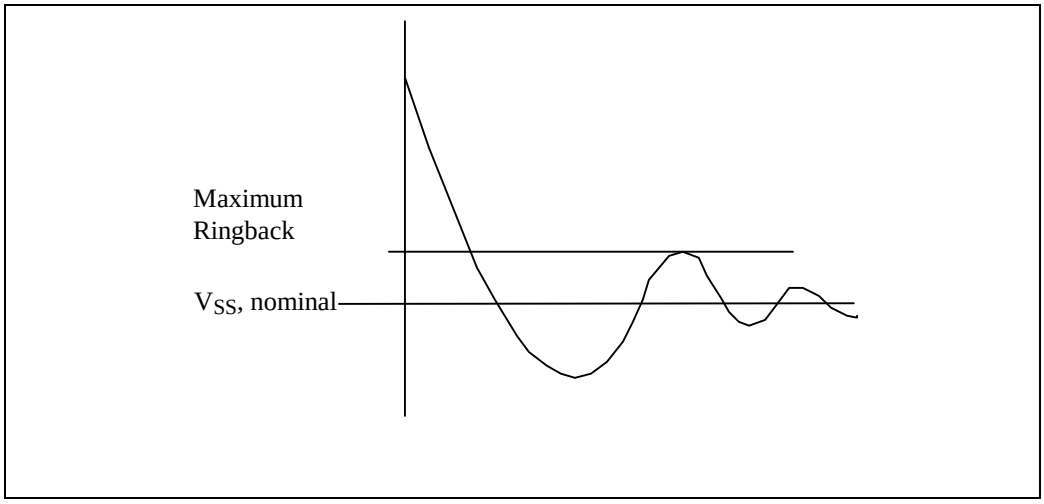


Figure 8-9. Maximum Ringback Associated with the Signal Low State

9

Mechanical Specifications

CHAPTER 9
MECHANICAL SPECIFICATIONS

The Pentium processor is packaged in 296-pin ceramic Staggered Pin Grid Array (SPGA) or plastic (PPGA) packages. The pins of the Pentium processor are arranged in a 37 x 37 matrix and the package dimensions are 1.95" x 1.95" (Table 9-1). There is a nickel plated copper heat slug attached to the top of the packages.

The Pentium processor is available in both ceramic and plastic packages.

Table 9-1. Package Information Summary

	Package Type	Total Pins	Pin Array	Package Size
Ceramic Staggered Pin Grid Array	SPGA	296	37 x 37	1.95" x 1.95" 4.95 cm x 4.95 cm
Plastic Staggered Pin Grid Array	PPGA	296	37 x 37	1.95" x 1.95" 4.95 cm x 4.95 cm

Figure 9-1 shows the dimensions of the ceramic packages. Table 9-2 provides the mechanical specifications of the ceramic packages. Figure 9-2 shows the dimensions of the plastic package, and Table 9-3 provides the mechanical specfications for the plastic package.

Note: for more information, see Application Note 577, "An Introduction to PPGA Packaging" (Order # 243103).

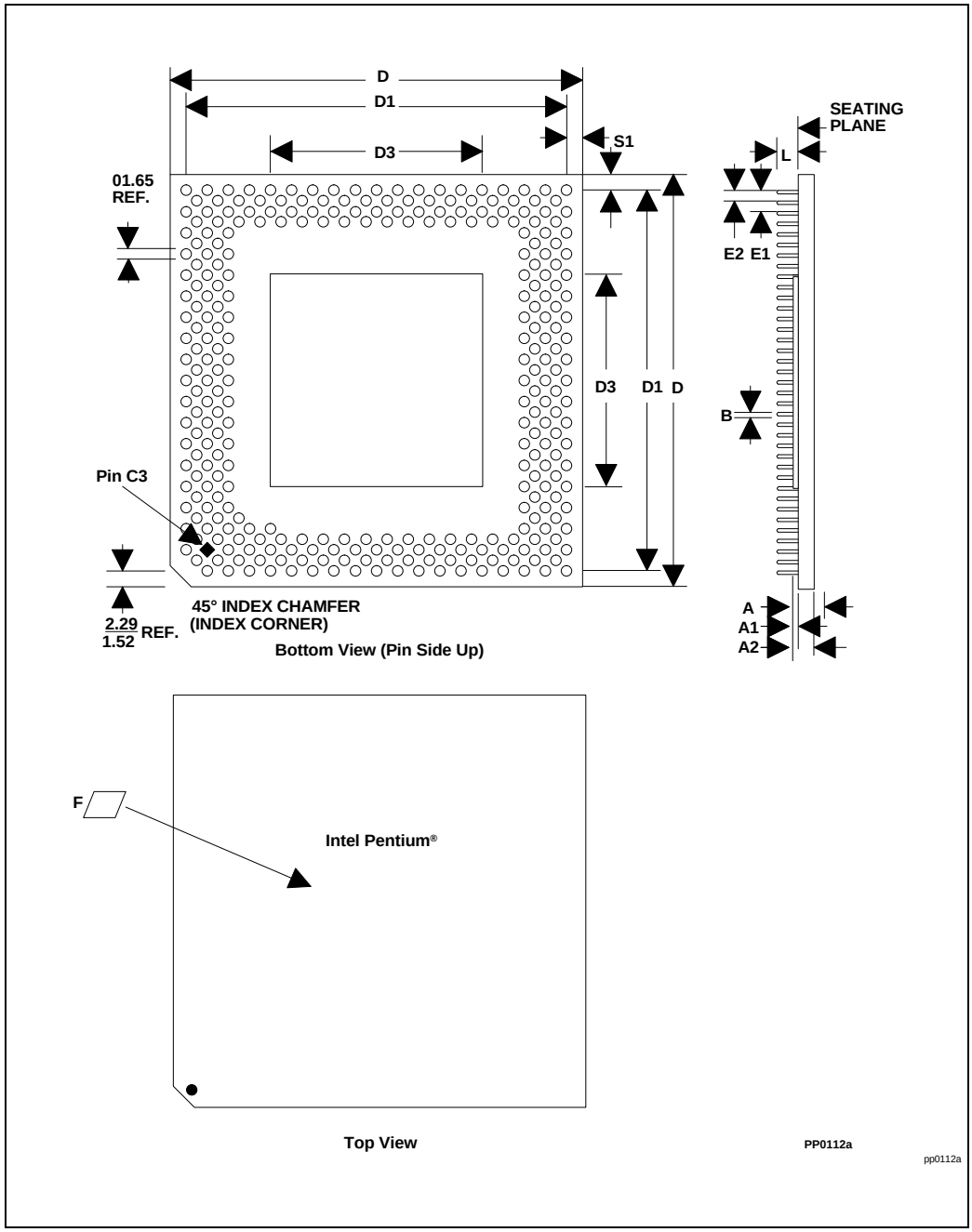


Figure 9-1. Ceramic Package (without the Heat Spreader) Dimensions

Table 9-2. Ceramic Package (without the Heat Spreader) Dimensions

Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	2.62	2.97		0.103	0.117	
A_1	0.69	0.84	Ceramic Lid	0.027	0.033	Ceramic Lid
A_2	3.31	3.81	Ceramic Lid	0.130	0.150	Ceramic Lid
B	0.43	0.51		0.017	0.020	
D	49.28	49.78		1.940	1.960	
D_1	45.59	45.85		1.795	1.805	
e_1	2.29	2.79		0.090	0.110	
L	3.05	3.30		0.120	0.130	
N	296		Lead Count	296		Lead Count
S_1	1.52	2.54		0.060	0.100	

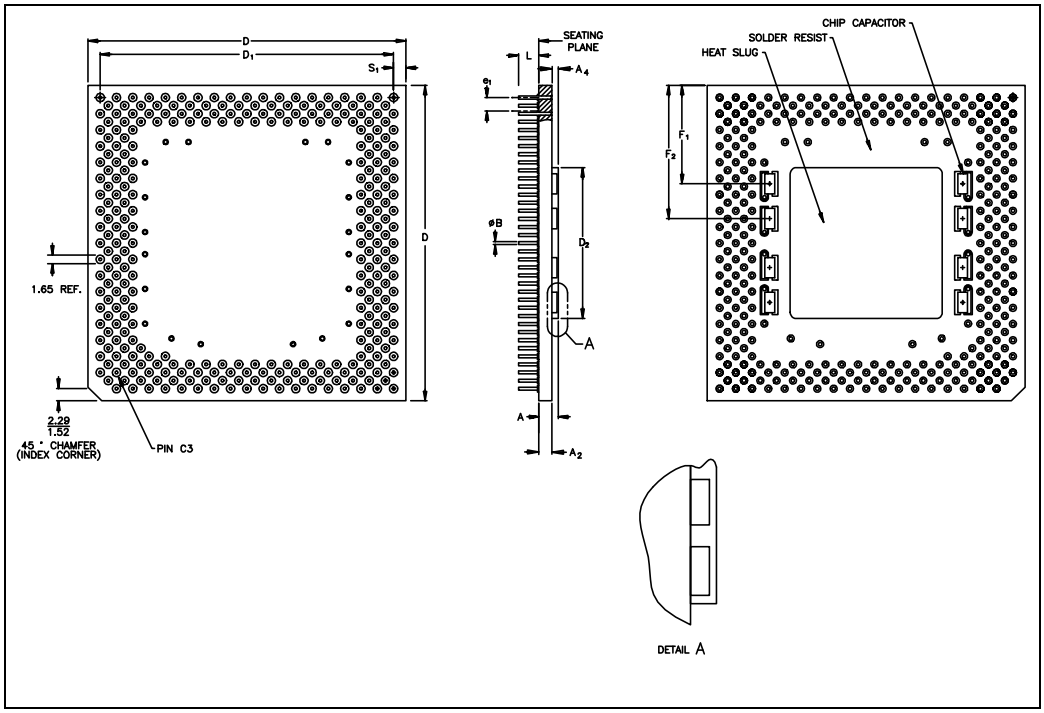


Figure 9-2. PPGA Package Dimensions

Table 9-3. PPGA Package Dimensions

	Millimeters			Inches		
Symbol	**Min**	**Max**	**Notes**	**Min**	**Max**	**Notes**
A	2.72	3.33		0.107	0.131	
A_1	1.83	2.23		0.072	0.088	
A_2	1.00			0.039		
B	0.40	0.51		0.016	0.020	
D	49.43	49.63		1.946	1.954	
D_1	45.59	45.85		1.795	1.805	
D_2	23.44	23.95		0.923	0.943	
e_1	2.29	2.79		0.090	0.110	
F_1	17.56			0.692		
F_2	23.04			0.907		
L	3.05	3.30		0.120	0.130	
N	296		Lead Count	296		Lead Count
S_1	1.52	2.54		0.060	0.100	

10

Thermal Specifications

CHAPTER 10
THERMAL SPECIFICATIONS

The Pentium processor is specified for proper operation when case temperature, T_{CASE}, (T_C) is within the specified range of 0°C to 70°C.

10.1. MEASURING THERMAL VALUES

To verify that the proper T_C (case temperature) is maintained, it should be measured at the center of the package top surface (opposite of the pins). The measurement is made in the same way with or without a heat sink attached. When a heat sink is attached, a hole (smaller than 0.150" diameter) should be drilled through the heat sink to allow probing the center of the package. See Figure 10-1 for an illustration of how to measure T_C.

To minimize the measurement errors, it is recommended to use the following approach:

- Use 36-gauge or finer diameter K, T, or J type thermocouples. Intel's laboratory testing was done using a thermocouple made by Omega (part number: 5TC-TTK-36-36).
- Attach the thermocouple bead or junction to the center of the package top surface using high thermal conductivity cements. The laboratory testing was done by using Omega Bond (part number: OB-101).
- The thermocouple should be attached at a 90-degrees angle as shown in Figure 10-1.
- The hole size should be smaller than 0.150" in diameter.

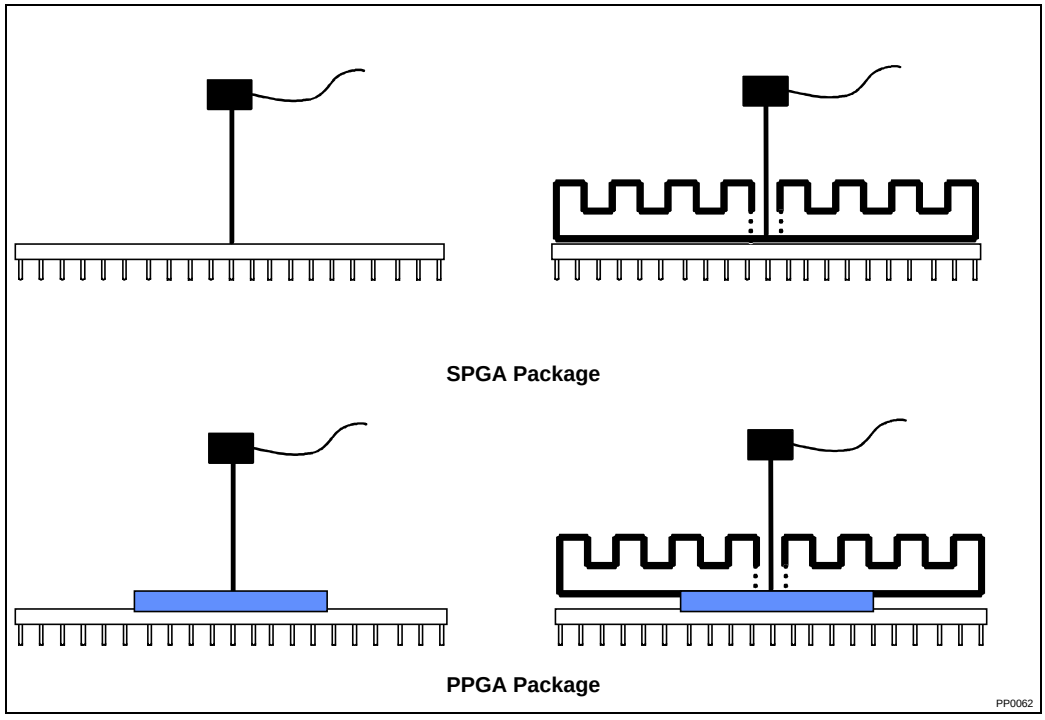


Figure 10-1. Technique for Measuring T_C [1]

NOTE:

1. The same technique applies to measuring TC of the package with or without a heat spreader.

10.1.1. Thermal Equations and Data

For the Pentium processor, an ambient temperature, T_A (air temperature around the processor), is not specified directly. The only restriction is that T_C is met. To calculate T_A values, the following equations may be used:

$$T_J = T_C + (P * \Theta_{JC})$$

$$T_A = T_J - (P * \Theta_{JA})$$

$$T_A = T_C - (P * \Theta_{CA})$$

$$\Theta_{CA} = \Theta_{JA} - \Theta_{JC}$$

where, T_A and T_C = ambient and case temperature. (°C)

Θ_{CA} = case-to-ambient thermal resistance. (°C/Watt)

Θ_{JA} = junction-to-ambient thermal resistance. (°C/Watt)

Θ_{JC} = junction-to-case thermal resistance. (°C/Watt)

P = maximum power consumption (Watt)

(See the DC specifications in Chapter 7 for detailed power consumption specifications.)

Table 10-1 through Table 10-5 list the Θ_{JC} and Θ_{CA} values for the Pentium processor (75/90/100/120/133/150/166/200) and Pentium processor with MMX technology. Figure 10-2 through Figure 10-6 show Θ_{CA} vs. Heatsink height graphically.

Θ_{CA} is the thermal resistance from package case to ambient. The Θ_{CA} values shown in these tables are typical values. The actual Θ_{CA} values depend on the heat sink design, interface between the heat sink and package, the air flow in the system, and thermal interactions between the processor and surrounding components through PCB and the ambient. Θ_{JC} is the thermal resistance from die to package case. The Θ_{JC} values shown in these tables are typical values. The actual Θ_{JC} values depend on actual thermal conductivity and the process of die attach.

For more detailed information regarding thermal design issues, see the *"Pentium® Processor Thermal Design Guidelines,"* Application Note (Order # 241575).

Table 10-1. Thermal Resistances for SPGA Packages (without Heat Spreader) for the Pentium® Processor (75/90/100/120)

Heat Sink in Inches	Θ_{JC} (°C/Watt)	Θ_{CA} (°C/Watt) vs. Laminar Airflow (linear ft/min)					
		0	100	200	400	600	800
0.25	0.8	9.1	8.0	6.6	4.5	3.6	3.0
0.35	0.8	8.8	7.5	6.0	4.0	3.3	2.8
0.45	0.8	8.4	7.0	5.3	3.6	2.9	2.5
0.55	0.8	8.1	6.5	4.7	3.2	2.6	2.3
0.65	0.8	7.7	6.0	4.3	3.0	2.4	2.1
0.80	0.8	7.0	5.3	3.9	2.8	2.2	2.0
1.00	0.8	6.3	4.7	3.6	2.6	2.1	1.8
1.20	0.8	5.9	4.3	3.3	2.4	2.0	1.8
1.40	0.8	5.4	3.9	3.0	2.2	1.9	1.7
Without Heat Sink	1.3	14.4	13.1	11.7	8.8	7.4	6.5

NOTES:

Heat sinks are omni directional pin aluminum alloy.

Features were based on standard extrusion practices for a given height

Pin size ranged from 50 to 129 mils

Pin spacing ranged from 93 to 175 mils

Based thickness ranged from 79 to 200 mils

Heat sink attach was 0.005" of thermal grease.

Attach thickness of 0.002" will improve performance approximately 0.3^{O}C/Watt

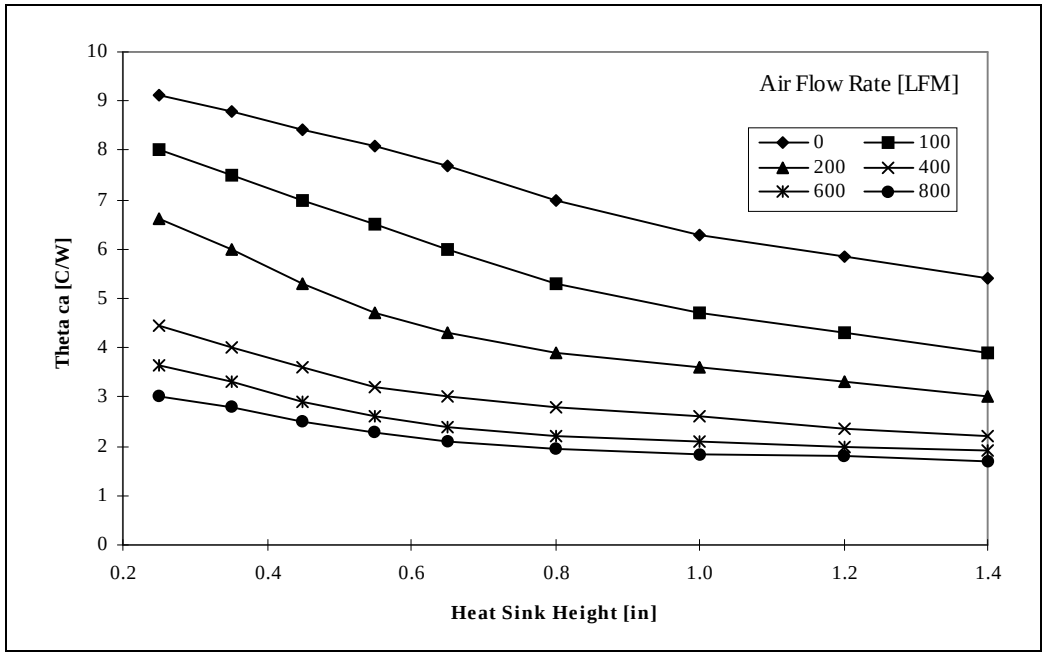


Figure 10-2. Thermal Resistance vs. Heatsink Height — SPGA Package without Heat Spreader for the Pentium® Processor (75/90/100/120)

Table 10-2. Thermal Resistances for SPGA Packages (without Heat Spreader) for the Pentium® Processor (120/133/150/166/200)

Heat Sink Height in Inches	θ_{JC} (°C/Watt)	θ_{CA}(°C/Watt) vs. Laminar Airflow (linear ft/min)					
		0	100	200	400	600	800
0.25	1.2	9.4	8.3	6.9	4.8	3.9	3.3
0.35	1.2	9.1	7.8	6.3	4.3	3.6	3.1
0.45	1.2	8.7	7.3	5.6	3.9	3.2	2.8
0.55	1.2	8.4	6.8	5.0	3.5	2.9	2.6
0.65	1.2	8.0	6.3	4.6	3.3	2.7	2.4
0.80	1.2	7.3	5.6	4.2	3.1	2.5	2.3
1.00	1.2	6.6	5.0	3.9	2.9	2.4	2.1
1.20	1.2	6.2	4.6	3.6	2.7	2.3	2.1
1.40	1.2	5.7	4.2	3.3	2.5	2.2	2.0
Without Heat Sink	1.7	14.5	13.8	12.6	10.5	8.6	7.5

NOTES:

Heat sinks are omni directional pin aluminum alloy.

Features were based on standard extrusion practices for a given height

Pin size ranged from 50 to 129 mils

Pin spacing ranged from 93 to 175 mils

Based thickness ranged from 79 to 200 mils

Heat sink attach was 0.005” of thermal grease.

Attach thickness of 0.002” will improve performance approximately 0.3^{O}C/Watt

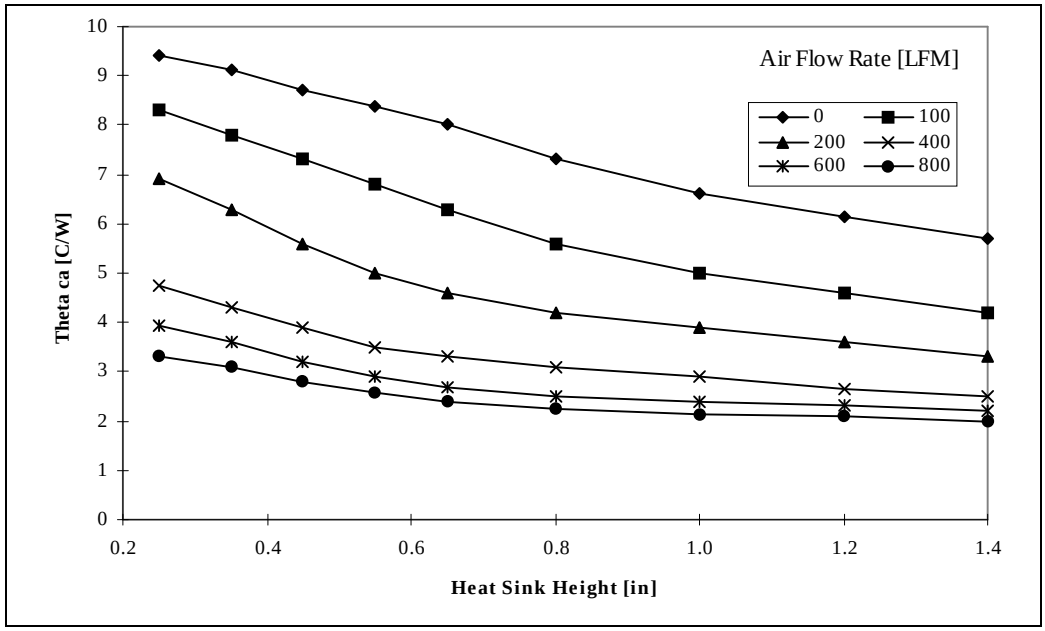


Figure 10-3. Thermal Resistance vs. Heatsink Height — SPGA Packages (without Heat Spreader) for the Pentium® Processor (120/133/150/166/200)

Table 10-3. Thermal Resistances for PPGA Packages for the Pentium® Processor (120/133/150/166/200)

Heat Sink Height in Inches	θ_{JC} (°C/Watt)	θ_{CA}(°C/Watt) vs. Laminar Airflow (linear ft/min)					
		0	100	200	400	600	800
0.25	0.5	9.0	7.9	6.5	4.4	3.5	2.9
0.35	0.5	8.7	7.4	5.9	3.9	3.2	2.7
0.45	0.5	8.3	6.9	5.2	3.5	2.8	2.4
0.55	0.5	8.0	6.4	4.6	3.1	2.5	2.2
0.65	0.5	7.6	5.9	4.2	2.9	2.3	2.0
0.80	0.5	6.9	5.2	3.8	2.7	2.1	1.9
1.00	0.5	6.2	4.6	3.5	2.5	2.0	1.7
1.20	0.5	5.8	4.2	3.2	2.3	1.9	1.7
1.40	0.5	5.3	3.8	2.9	2.1	1.8	1.6
Without Heat Sink	1.3	13.0	12.3	11.4	8.0	6.6	5.7

NOTES:

Heat sinks are omni directional pin aluminum alloy.

Features were based on standard extrusion practices for a given height

Pin size ranged from 50 to 129 mils

Pin spacing ranged from 93 to 175 mils

Based thickness ranged from 79 to 200 mils

Heat sink attach was 0.005” of thermal grease.

Attach thickness of 0.002” will improve performance approximately 0.1ºC/Watt

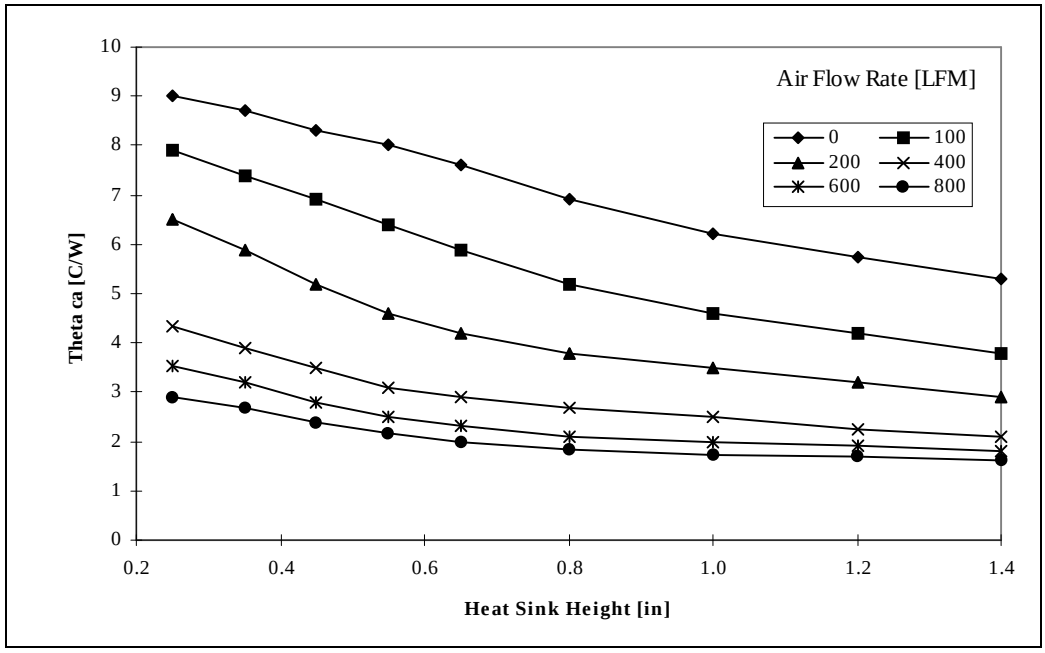


Figure 10-4. Thermal Resistance vs. Heatsink Height — PPGA Packages for the Pentium® Processor (120/133/150/166/200)

Table 10-4. Thermal Resistance for SPGA Packages for the Pentium® Processor with MMX™ Technology

Heat Sink Height in Inches	θ_{JC} (°C/Watt)	θ_{CA}(°C/Watt) vs. Laminar Airflow (linear ft/min)					
		0	100	200	400	600	800
0.25	0.9	9.2	8.1	6.7	4.6	3.7	3.1
0.35	0.9	8.9	7.6	6.1	4.1	3.4	2.9
0.45	0.9	8.5	7.1	5.4	3.7	3.0	2.6
0.55	0.9	8.2	6.6	4.8	3.3	2.7	2.4
0.65	0.9	7.8	6.1	4.4	3.1	2.5	2.2
0.80	0.9	7.1	5.4	4.0	2.9	2.3	2.1
1.00	0.9	6.4	4.8	3.7	2.7	2.2	1.9
1.20	0.9	6.0	4.4	3.4	2.5	2.1	1.9
1.40	0.9	5.5	4.0	3.1	2.3	2.0	1.8
Without Heat Sink	1.4	14.4	13.4	12.1	9.7	8.0	7.0

NOTES:

Heat sinks are omni directional pin aluminum alloy.

Features were based on standard extrusion practices for a given height

Pin size ranged from 50 to 129 mils

Pin spacing ranged from 93 to 175 mils

Based thickness ranged from 79 to 200 mils

Heat sink attach was 0.005" of thermal grease.

Attach thickness of 0.002" will improve performance approximately 0.3^{o}C/Watt

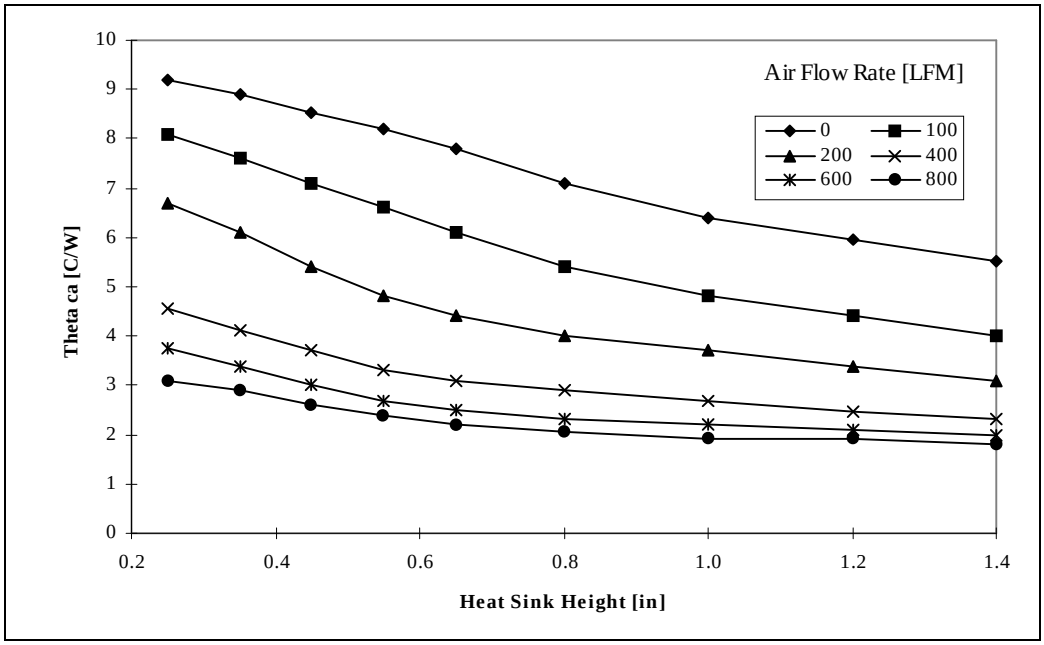


Figure 10-5. Thermal Resistance vs. Heatsink Height — SPGA Packages for the Pentium® Processor with MMX™ Technology

Table 10-5. Thermal Resistances for PPGA Packages for the Pentium® Processor with MMX™ Technology

Heat Sink Height in Inches	θ_{JC} (°C/Watt)	θ_{CA}(°C/Watt) vs. Laminar Airflow (linear ft/min)					
		0	100	200	400	600	800
0.25	0.4	8.9	7.8	6.4	4.3	3.4	2.8
0.35	0.4	8.6	7.3	5.8	3.8	3.1	2.6
0.45	0.4	8.2	6.8	5.1	3.4	2.7	2.3
0.55	0.4	7.9	6.3	4.5	3.0	2.4	2.1
0.65	0.4	7.5	5.8	4.1	2.8	2.2	1.9
0.80	0.4	6.8	5.1	3.7	2.6	2.0	1.8
1.00	0.4	6.1	4.5	3.4	2.4	1.9	1.6
1.20	0.4	5.7	4.1	3.1	2.2	1.8	1.6
1.40	0.4	5.2	3.7	2.8	2.0	1.7	1.5
Without Heat Sink	1.2	12.9	12.2	11.2	7.7	6.3	5.4

NOTES:

Heat sinks are omni directional pin aluminum alloy.

Features were based on standard extrusion practices for a given height

Pin size ranged from 50 to 129 mils

Pin spacing ranged from 93 to 175 mils

Based thickness ranged from 79 to 200 mils

Heat sink attach was 0.005” of thermal grease.

Attach thickness of 0.002” will improve performance approximately 0.1^{o}C/Watt

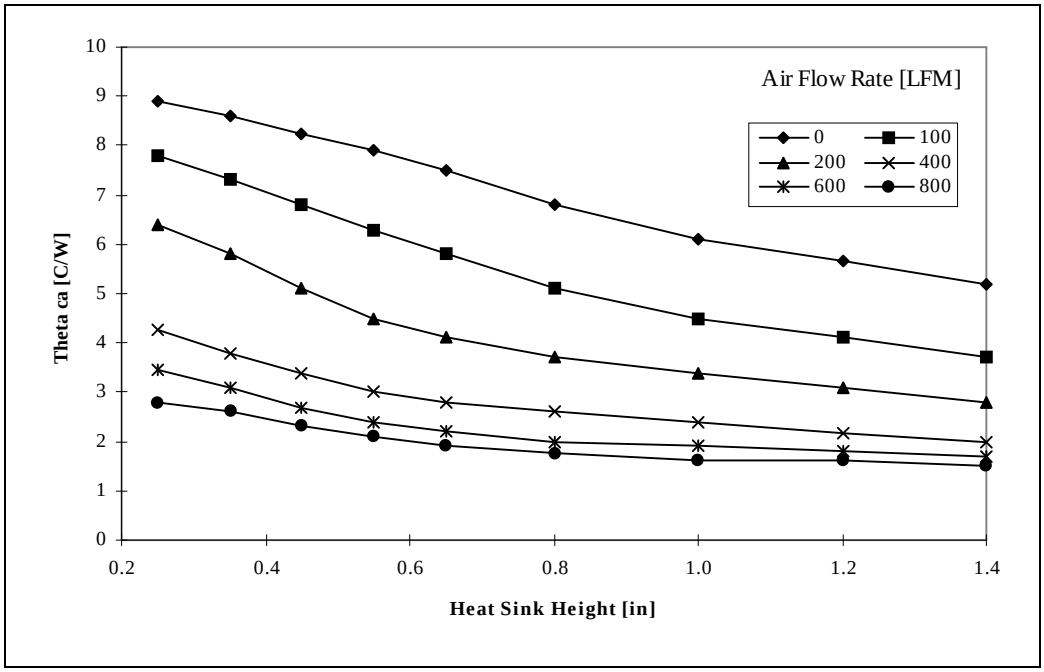


Figure 10-6. Thermal Resistance vs. Heatsink Height — PPGA Packages for the Pentium® Processor with MMX™ Technology

11

Testability

CHAPTER 11
TESTABILITY

This chapter describes the features which are included in the Pentium processor for the purpose of enhancing the testability of the Pentium processor. The capability of the Intel486 CPU test hooks are included in the Pentium processor; however, some are implemented differently. In addition, new test features were added to assure timely testing and production of the system product.

Internal component testing through the Built-InSelf-Test (BIST) feature of the Pentium processor provides 100% single stuck at fault coverage of the microcode ROM and large PLAs. Some testing of the instruction cache, data cache, Translation Lookaside Buffers (TLBs), and Branch Target Buffer (BTB) is also performed. In addition, the constant ROMs are checked.

Tristate test mode and the IEEE 1149.1 “Test Access Port and Boundary Scan” mechanism are included to facilitate testing of board connections.

See Chapter 16 for more information regarding the testing of the on-chip caches, translation lookaside buffers, branch target buffer, second level caches, the superscalar architecture, and internal parity checking through the test registers.

Built-in self-test, tristate test mode, and Boundary Scan are discussed in this chapter.

11.1. BUILT-IN SELF-TEST (BIST)

Self-test is initiated by driving the INIT pin high when RESET transitions from high to low.

No bus cycles are run by the Pentium processor during self-test. The duration of self-test is approximately 2^{19} core clocks. Approximately 70% of the devices in the Pentium processor are tested by BIST.

The Pentium processor BIST consists of two parts: hardware self-test and microcode self-test.

During the hardware portion of BIST, the microcode and all large PLAs are tested. All possible input combinations of the microcode ROM and PLAs are tested.

The constant ROMs, BTB, TLBs and all caches are tested by the microcode portion of BIST. The array tests (caches, TLBs and BTB) have two passes. On the first pass, data patterns are written to arrays, read back and checked for mismatches. The second pass writes the complement of the initial data pattern, reads it back and checks for mismatches. The constant ROMs are tested by using the microcode to add various constants and check the result against a stored value.

Upon completion of BIST, the cumulative result of all tests are stored in the EAX register. If EAX contains 0h, then all checks passed; any non-zero result indicates a faulty unit. Note that if an internal parity error is detected during BIST, the processor will assert the IERR# pin and attempt to shutdown.

11.2. TRISTATE TEST MODE

When the FLUSH# pin is sampled low in the clock prior to the RESET pin going from high to low, the Pentium processor enters tristate test mode. The Pentium processor floats all of its output pins and bi-directional pins including pins which are never floated during normal operation (except TDO). Tristate test mode can be initiated in order to facilitate testing of board connections. The Pentium processor remains in tristate test mode until the RESET pin is toggled again.

In a dual-processor system, the private interface pins are not floated in Tri-state Test mode. These pins are PBREQ#, PBGNT#, PHIT#, and PHITM#.

NOTE

There are several pins that have internal pullups or pulldowns attached that show these pins going high or low, respectively, during Tri-state Test mode. There is one pin, PICD1, that has an internal pulldown attached that shows this pin going low during Tri-state Test mode. The five pins that have pullups are PHIT#, PHITM#, PBREQ#, PBGNT#, and PICD0. There are two other pins that have pullups attached during dual processor mode, HIT# and HITM#. The pullups on these pins (except HIT#) have a value of about 30K Ohms, HIT# is about 2K Ohms.

11.3. IEEE 1149.1 TEST ACCESS PORT AND BOUNDARY SCAN MECHANISM

The IEEE Standard Test Access Port and Boundary Scan Architecture (Standard 1149.1) is implemented in the Pentium processor. This feature allows board manufacturers to test board interconnects by using “boundary scan,” and to test the Pentium processor itself through BIST. All output pins are tristateable through the IEEE 1149.1 mechanism.

11.3.1. Pentium® Processor Test Access Port (TAP)

The Pentium processor Test Access Port (TAP) contains a TAP controller, a Boundary Scan Register, 4 input pins (TDI, TCK, TMS, and TRST#) and one output pin (TDO). The TAP controller consists of an Instruction Register, a Device ID Register, a Bypass Register, a Runbist Register and control logic. See Figure 11-1 for the TAP Block Diagram.

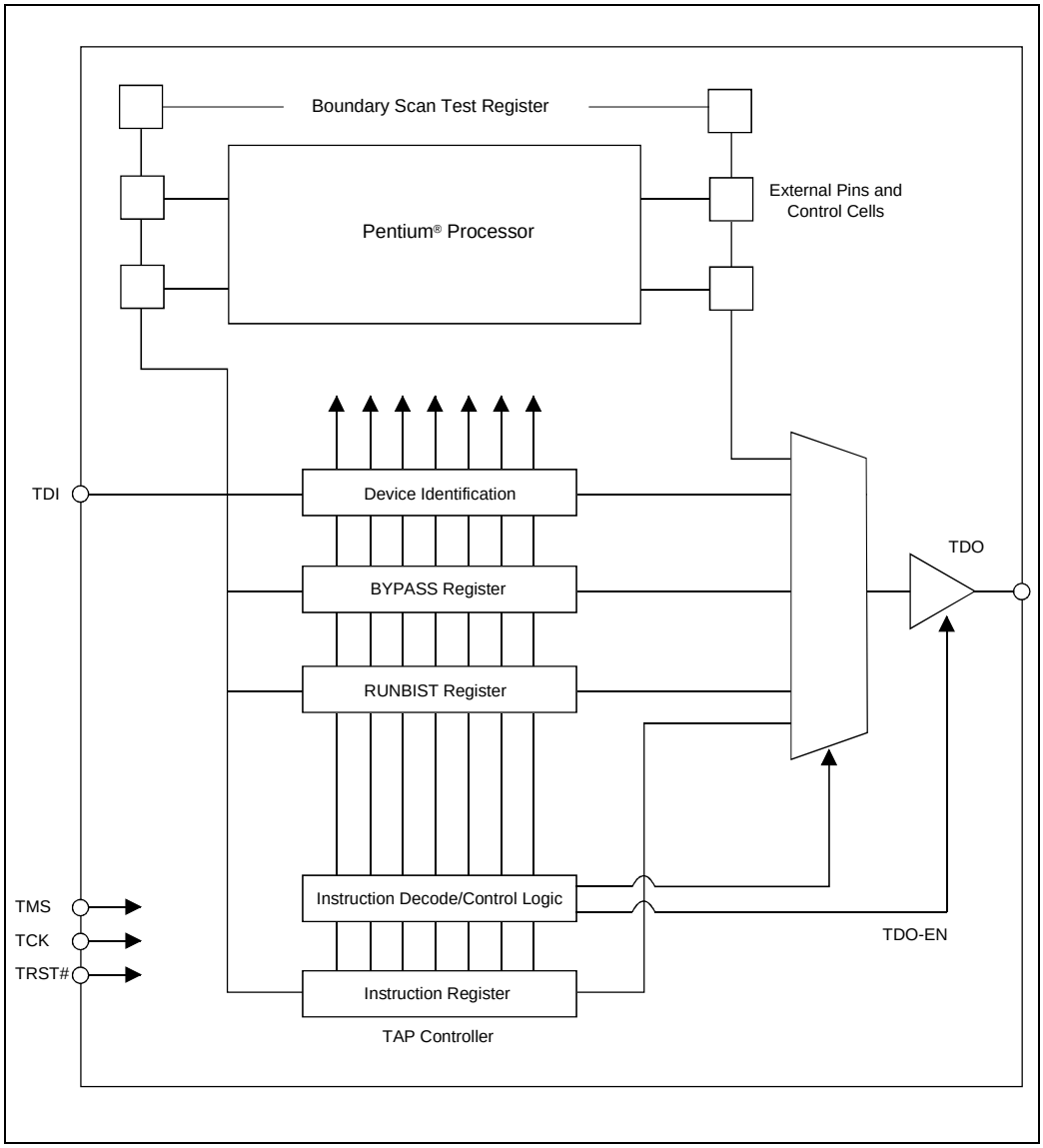


Figure 11-1. Test Access Port Block Diagram

11.3.1.1. TAP PINS

As mentioned in the previous section, the TAP includes 4 input pins and one output pin. TDI (test data in) is used to shift data or instructions into the TAP in a serial manner. TDO (test data out) shifts out the response data. TMS (test mode select) is used to control the state of the TAP controller. TCK is the test clock. The TDI and TMS inputs are sampled on the rising edge of this TCK. Asserting TRST# will force the TAP controller into the Test Logic Reset State (see the TAP controller state diagram, Figure 11-4). The input pins (TDI, TMS, TCK, and TRST#) have pullup resistors.

11.3.1.2. TAP REGISTERS

Boundary Scan Register

The IEEE standard requires that an extra single bit shift register be inserted at each pin on the device (Pentium processor). These single bit shift registers are connected into a long shift register, the Boundary Scan Register. Therefore, the Boundary Scan Register is a single shift register path containing the boundary scan cells that are connected to all input and output pins of the Pentium processor. Figure 11-2 shows the logical structure of the Boundary Scan Register. While output cells determine the value of the signal driven on the corresponding pin, input cells only capture data; they do not affect the normal operation of the device (the INTEST instruction is not supported by the Pentium processor). Data is transferred without inversion from TDI to TDO through the Boundary Scan Register during scanning. The Boundary Scan Register can be operated by the EXTEST and SAMPLE/PRELOAD instructions. The Boundary Scan Register order is defined later in this chapter.

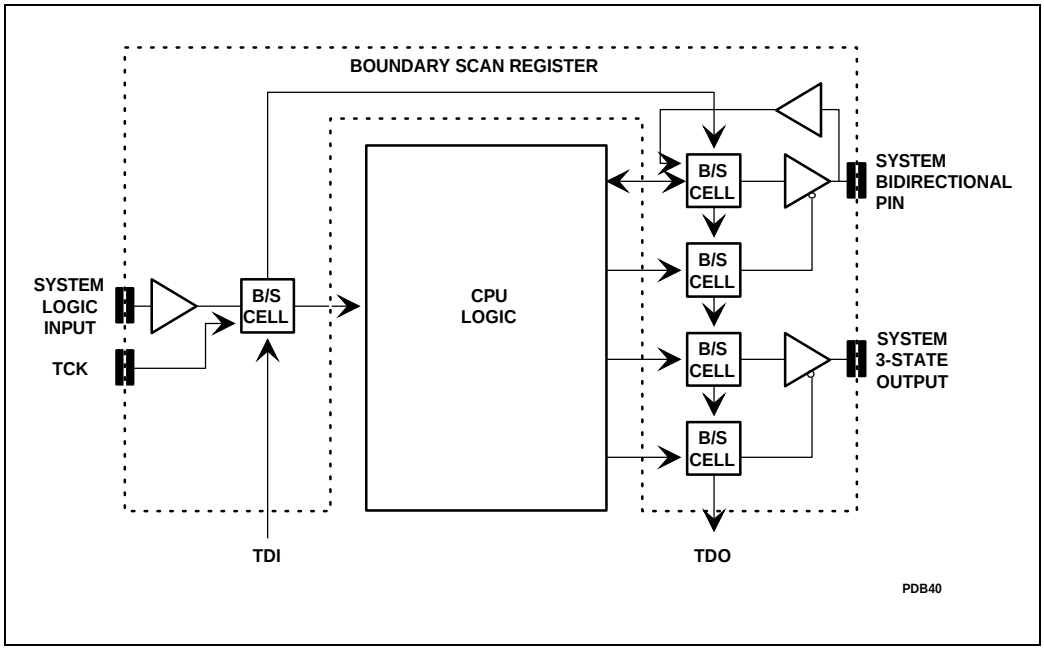


Figure 11-2. Boundary Scan Register

BYPASS Register

The Bypass Register is a one-bit shift register that provides the minimal length path between TDI and TDO. This path can be selected when no test operation is being performed by the component to allow rapid movement of test data to and from other components on the board. While the bypass register is selected data is transferred from TDI to TDO without inversion. The Bypass Register loads a logic 0 at the start of a scan cycle.

Device ID Register

The Device Identification Register contains the manufacturer's identification code, part number code, and version code in the format shown in Figure 11-3. It is selected to be connected between TDI and TDO by using the IDCODE instruction.

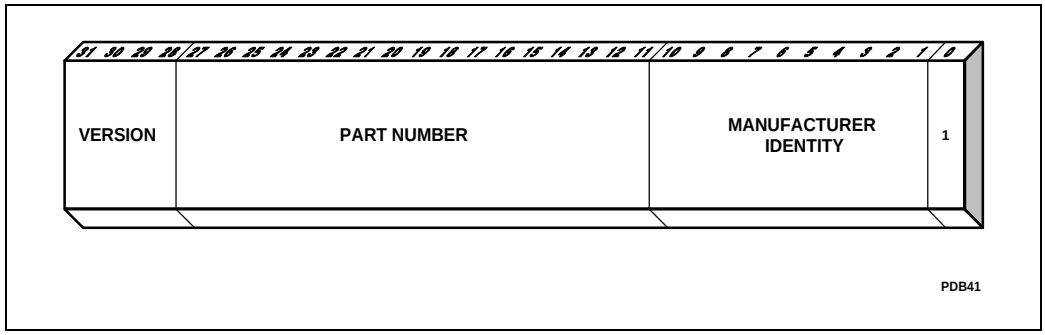


Figure 11-3. Format of the Device ID Register

The Pentium processor has divided up the 16-bit part number into 3 fields. The upper 7 bits are used to define the product type (examples: Cache, CPU architecture). The middle 4 bits are used to represent the generation or family (examples: Intel486 CPU, Pentium processor). The lower 5 bits are used to represent the model (examples: SX, DX). Using this definition, the Pentium processor ID code is shown in Table 11-1.

The version field is used to indicate the stepping ID.

Table 11-1. Device ID Register Values

Processor	Stepping	Version	Part Number			Manufacturing ID	"1"	Entire Code
			Product Type	Generation	Model			
Pentium® processor (75/90/120/ 133/150/ 166/200)	x	xh	01h	05h	08h	09h	1	x82A8013h
	x	xh	01h	05h	04h	09h	1	x82A4013h
Pentium processor with MMX™ technology	x	xh	01h	05h	03h	09h	1	x82A3013h

Runbist Register

The Runbist Register is a one bit register used to report the results of the Pentium processor BIST when it is initiated by the RUNBIST instruction. This register is loaded with "0" upon successful completion of BIST.

Instruction Register

This register is 13 bits wide. The command field (the lower 4 bits of instruction) is used to indicate one of the following instructions: EXTEST, IDCODE, RUNBIST, SAMPLE/PRELOAD and BYPASS. The upper 9 bits are reserved by Intel.

The most significant bit of the Instruction Register is connected to TDI, the least significant to TDO.

11.3.1.3. TAP CONTROLLER STATE DIAGRAM

Figure 11-4 shows the 16-state TAP controller state diagram. A description of each state follows. Note that the state machine contains two main branches to access either data or instruction registers.

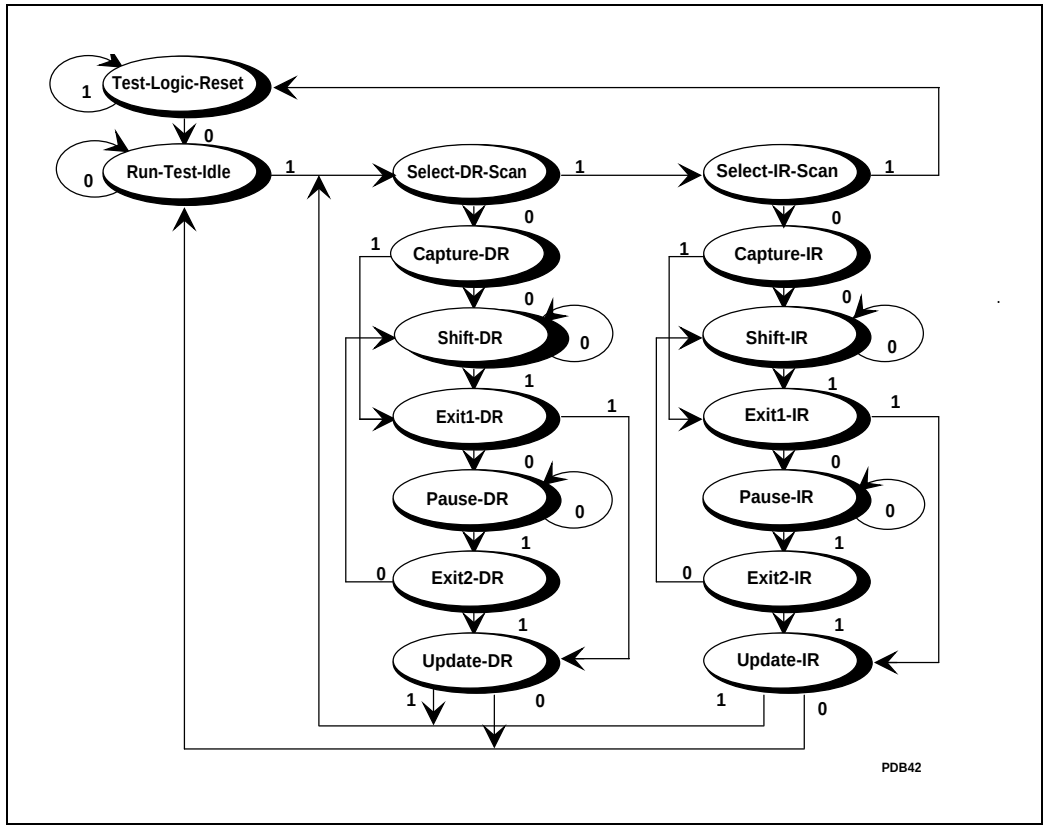


Figure 11-4. TAP Controller State Diagram

Test-Logic-Reset State

In this state, the test logic is disabled so that normal operation of the device can continue unhindered. During initialization, the Pentium processor initializes the instruction register such that the IDCODE instruction is loaded.

No matter what the original state of the controller, the controller enters Test-Logic-Reset state when the TMS input is held high (logic 1) for at least five rising edges of TCK. The controller remains in this state while TMS is high. The TAP controller is forced to enter this state when the TRST# pin is asserted (with TCK toggling or TCK at a high logic value). The Pentium processor automatically enters this state at power-up.

Run-Test/Idle State

This is a controller state between scan operations. Once in this state, the controller remains in this state as long as TMS is held low. In devices supporting the RUNBIST instruction, the BIST is performed during this state and the result is reported in the Runbist Register. For instructions not causing functions to execute during this state, no activity occurs in the test logic. The instruction register and all test data registers retain their previous state. When TMS is high and a rising edge is applied to TCK, the controller moves to the Select-DR state.

Select-DR-Scan State

This is a temporary controller state. The test data register selected by the current instruction retains its previous state. If TMS is held low and a rising edge is applied to TCK when in this state, the controller moves into the Capture-DR state, and a scan sequence for the selected test data register is initiated. If TMS is held high and a rising edge is applied to TCK, the controller moves to the Select-IR-Scan state.

The instruction does not change in this state.

Capture-DR State

In this state, the Boundary Scan Register captures input pin data if the current instruction is EXTEST or SAMPLE/PRELOAD. The other test data registers, which do not have parallel input, are not changed.

The instruction does not change in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-DR state if TMS is high or the Shift-DR state if TMS is low.

Shift-DR State

In this controller state, the test data register connected between TDI and TDO as a result of the current instruction shifts data one stage toward its serial output on each rising edge of TCK.

The instruction does not change in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-DR state if TMS is high or remains in the Shift-DR state if TMS is low.

Exit1-DR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-DR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Pause-DR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

Pause-DR State

The pause state allows the test controller to temporarily halt the shifting of data through the test data register in the serial path between TDI and TDO. An example use of this state could be to allow a tester to reload its pin memory from disk during application of a long test sequence.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

The controller remains in this state as long as TMS is low. When TMS goes high and a rising edge is applied to TCK, the controller moves to the Exit2-DR state.

Exit2-DR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-DR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Shift-DR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

Update-DR State

The Boundary Scan Register is provided with a latched parallel output to prevent changes at the parallel output while data is shifted in response to the EXTEST and SAMPLE/PRELOAD instructions. When the TAP controller is in this state and the Boundary Scan Register is selected, data is latched onto the parallel output of this register from the shift-register path on the falling edge of TCK. The data held at the latched parallel output does not change other than in this state.

All shift-register stages in the test data register selected by the current instruction retains their previous value during this state. The instruction does not change in this state.

Select-IR-Scan State

This is a temporary controller state. The test data register selected by the current instruction retains its previous state. If TMS is held low and a rising edge is applied to TCK when in this

state, the controller moves into the Capture-IR state, and a scan sequence for the instruction register is initiated. If TMS is held high and a rising edge is applied to TCK, the controller moves to the Test-Logic-Reset state. The instruction does not change in this state.

Capture-IR State

In this controller state the shift register contained in the instruction register loads a fixed value on the rising edge of TCK.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

When the controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-IR state if TMS is held high, or the Shift-IR state if TMS is held low.

Shift-IR State

In this state the shift register contained in the instruction register is connected between TDI and TDO and shifts data one stage towards its serial output on each rising edge of TCK.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

When the controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-IR state if TMS is held high, or remains in the Shift-IR state if TMS is held low.

Exit1-IR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-IR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Pause-IR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

Pause-IR State

The pause state allows the test controller to temporarily halt the shifting of data through the instruction register.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

The controller remains in this state as long as TMS is low. When TMS goes high and a rising edge is applied to TCK, the controller moves to the Exit2-IR state.

Exit2-IR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-IR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Shift-IR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

Update-IR State

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of TCK. Once the new instruction has been latched, it becomes the current instruction.

Test data registers selected by the current instruction retain their previous value.

11.3.2. Boundary Scan

The IEEE Standard 1149.1 Boundary Scan is implemented using the Test Access Port and TAP Controller as described above. The Pentium processor implements all of the required boundary scan features as well as some additional features. The required pins (all 3.3V) are: TDI, TDO, TCK and TMS. The required registers are: Boundary Scan, Bypass, and the Instruction Register. Required instructions include: BYPASS, SAMPLE/PRELOAD and EXTEST. The additional pin, registers, and instructions are implemented to add additional test features.

On the board level, the TAP provides a simple serial interface that makes it possible to test all signal traces with only a few probes. The testing is controlled through the TAP Controller State machine that can be implemented with automatic test equipment or a PLD.

On power up the TAP controller is automatically initialized to the test logic reset state (test logic disabled), so normal Pentium processor behavior is the default. The Test Logic Reset State is also entered when TRST# is asserted, or when TMS is high for 5 or more consecutive TCK clocks.

To implement boundary scan, the TDO of one device is connected to TDI of the next in a daisy-chain fashion. This allows all of the I/O of the devices on this chain to be accessed through a long shift register. TMS and TCK are common to all devices.

The Boundary Scan Register for the Pentium processor contains a cell for each pin.

The following is the bit order of the Pentium processor with MMX technology Boundary Scan Register (left to right, top to bottom):

TDI → Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, STPCLK#, Reserved, Reserved, Reserved, Reserved, Reserved, FRCMC#, PEN#, INIT, IGNNE#, SMI#, INTR, RS#, NMI, D/P#, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, Disabus*, RESET, CLK, SCYC, BE7#, BE6#, BE5#, BE4#, BE3#, BE2#, BE1#, BE0#, A20M#, FLUSH#, BUSCHK#, W/R#, HIT#, HITM#, ADS#, EADS#, D/C#, PWT, PCD, ADSC#,

LOCK#, AP, HLDA, BREQ, APCHK#, PCHK#, PRDY, SMIACT#, PBGNT#, PBREQ#, PHIT#, PHITM#, HOLD, WB/WT#, Dismiscf*, Dismisca*, Disua1bus*, Disua2bus*, Dismisc*, Disbusl*, Dismisch*, Disbus*, NA#, BOFF#, BRDY#, BRDYC#, KEN#, AHOLD, INV, EWBE#, CACHE#, M/IO#, BP3, BP2, PM1BP1, PM0BP0, FERR#, IERR#, DP7, D63, D62, D61, D60, D59, D58, D57, D56, DP6, D55, D54, D53, D52, D51, D50, D49, D48, DP5, D47, Diswr*, D46, D45, D44, D43, D42, D41, D40, DP4, D39, D38, D37, D36, D35, D34, D33, D32, DP3, D31, D30, D29, D28, D27, D26, D25, D24, DP2, D23, D22, D21, D20, D19, D18, D17, D16, DP1, D15, D14, D13, D12, D11, D10, D9, D8, DP0, D7, D6, D5, D4, D3, D2, D1, D0, PICCLK, Reserved, PICD0, PICD1, Disapsba* → TDO

The following is the bit order of the Pentium processor (75/90/100/120/133/150/166/200) Boundary Scan Register (left to right, top to bottom):

TDI -> Disapsba*, PICD1, PICD0, Reserved, PICCLK, D0, D1, D2, D3, D4, D5, D6, D7, DP0, D8, D9, D10, D11, D12, D13, D14, D15, DP1, D16, D17, D18, D19, D20, D21, D22, D23, DP2, D24, D25, D26, D27, D28, D29, D30, D31, DP3, D32, D33, D34, D35, D36, D37, D38, D39, DP4, D40, D41, D42, D43, D44, D45, D46, Diswr*, D47, DP5, D48, D49, D50, D51, D52, D53, D54, D55, DP6, D56, D57, D58, D59, D60, D61, D62, D63, DP7, IERR#, FERR#, PM0/BP0, PM1/BP1, BP2, BP3, M/IO#, CACHE#, EWBE#, INV, AHOLD, KEN#, BRDYC#, BRDY#, BOFF#, NA#, Disbus*, Dismisch*, Disbus1*, Dismisc*, Disua2bus*, Disua1bus*, Dismisca*, Dismiscfa*, WB/WT#, HOLD, PHITM#, PHIT#, PBREQ#, PBGNT#, SMIACT#, PRDY, PCHK#, APCHK#, BREQ, HLDA, AP, LOCK#, ADSC#, PCD, PWT, D/C#, EADS#, ADS#, HITM#, HIT#, W/R#, BUSCHK#, FLUSH#, A20M#, BE0#, BE1#, BE2#, BE3#, BE4#, BE5#, BE6#, BE7#, SCYC, CLK, RESET, Disabus*, A20, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A31, A30, A29, A28, A27, A26, A25, A24, A23, A22, A21, D/P#, NMI, R/S#, INTR, SMI#, IGNNE#, INIT, PEN#, FRCMC#, Reserved, Reserved, BF0, BF1, Reserved, STPCLK#, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, CPUTYP -> TDO

“Reserved” includes the no connect “NC” signals on the Pentium processor.

The cells marked with an “*” are control cells that are used to select the direction of bi-directional pins or tristate the output pins. If “1” is loaded into the control cell, the associated pin(s) are tristated or selected as input. The following lists the control cells and their corresponding pins:

For Pentium Processor with MMX Technology:

Disabus:	A31-A3, AP.
Disbus:	BE7-0#, CACHE#, SCYC, M/IO#, D/C#, W/R#, PWT, PCD.
Disbusl:	ADS#, ADSC#, LOCK#.
Dismisc:	APCHK#, PCHK#, PRDY, BP3, BP2, PM1/BP1, PM0/BP0.
Dismiscf:	D/P#.
Dismisch:	FERR#, SMIACT#, BREQ, HLDA, HIT#, HITM#.
Dismisca:	IERR#.
Disua1bus:	PBREQ#, PHIT#, PHITM#.

Disua2bus: PBGNT#.

Diswr: D63-0, DP7-0.

Disapsba: PICD1-0.

For Pentium Processors (75/90/100/120/150/166/200):

Disabus: A31-A3, AP

Dismiscfa: D/P#, FERR#

Dismisca: IERR#

Disua1bus: PBREQ#, PHIT#, PHITM#

Disua2bus: PBGNT#

Dismisc: APCHK#, PHCK#, PRDY#, BP3, BP2, PM1/BP1, PM0/BP0

Disbus1: ADS#, ADSC#, LOCK#

Dismisch: HIT#, HITM#, HLDA, BREQ#, SMIACT#

Disbus: SCYC, BE7#-BE0#, W/R#, D/C#, PWT, PCD, CACHE#, M/IO#

Diswr: DP7-DP0, D63-D0

Disapsba: PICD0, PICD1

11.3.2.1. PENTIUM® PROCESSOR BOUNDARY SCAN TAP INSTRUCTION SET

Table 11-2 shows the Pentium Processor Boundary Scan TAP instructions and their instruction register encoding. A description of each instruction follows. The IDCODE and BYPASS instructions may also be executed concurrent with processor execution. The following instructions are not affected by the assertion of RESET: EXTEST, SAMPLE PRELOAD, BYPASS, and ID CODE.

The instructions should be scanned in to the TAP port least significant bit first (bit 0 of the TAP Command field is the first bit to be scanned in).

Table 11-2. TAP Instruction Set and Instruction Register Encoding

Instruction Name	Instruction Register Bits 12:4	TAP Command Field [Bits 3:0]
EXTEST	XXXXXXXXX	0000
Sample/Preload	XXXXXXXXX	0001
IDCODE	XXXXXXXXX	0010
Private Instruction	XXXXXXXXX	0011
Private Instruction	XXXXXXXXX	0100
Private Instruction	XXXXXXXXX	0101
Private Instruction	XXXXXXXXX	0110
RUNBIST	XXXXXXXXX	0111
Private Instruction	XXXXXXXXX	1000
Private Instruction	XXXXXXXXX	1001
Private Instruction	XXXXXXXXX	1010
HI-Z	XXXXXXXXX	1011
Private Instruction	XXXXXXXXX	1100
BYPASS	XXXXXXXXX	1111

The TAP Command field encodings not listed in Table 11-2 (1101, 1110) are unimplemented and will be interpreted as Bypass instructions.

EXTEST — The EXTEST instruction allows testing of circuitry external to the component package, typically board interconnects. It does so by driving the values loaded into the Pentium processor's Boundary Scan Register out on the output pins corresponding to each boundary scan cell and capturing the values on the Pentium processor input pins to be loaded into their corresponding Boundary Scan Register locations. I/O pins are selected as input or output, depending on the value loaded into their control setting locations in the Boundary Scan Register. Values shifted into input latches in the Boundary Scan Register are never used by the internal logic of the Pentium processor. Note: after using the EXTEST instruction, the Pentium processor must be reset before normal (non-boundary scan) use.

SAMPLE/PRELOAD — The SAMPLE/PRELOAD performs two functions. When the TAP controller is in the Capture-DR state, the SAMPLE/PRELOAD instruction allows a "snap-shot" of the normal operation of the component without interfering with that normal operation. The instruction causes Boundary Scan Register cells associated with outputs to sample the value being driven by the Pentium processor.

It causes the cells associated with inputs to sample the value being driven into the Pentium processor. On both outputs and inputs the sampling occurs on the rising edge of TCK. When the TAP controller is in the Update-DR state, the SAMPLE/PRELOAD instruction preloads data to the device pins to be driven to the board by executing the EXTEST instruction. Data is preloaded to the pins from the Boundary Scan Register on the falling edge of TCK.

IDCODE

The IDCODE instruction selects the device identification register to be connected to TDI and TDO. This allows the device identification code to be shifted out of the device on TDO.

RUNBIST

The RUNBIST instruction selects the one (1) bit Runbist Register, loads a value of "1" into the Runbist Register, and connects it to TDO. It also initiates the built-in self test (BIST) feature of the Pentium processor. After loading the RUNBIST instruction code in the instruction register, the TAP controller must be placed in the Run-Test/Idle state. BIST begins on the first rising edge of TCK after entering the Run-Test/Idle state. The TAP controller must remain in the Run-Test/Idle state until BIST is completed. It requires 2^{19} core clock cycles to complete BIST and report the result to the Runbist Register. After completing BIST, the value in the Runbist Register should be shifted out on TDO during the Shift-DR state. A value of "0" being shifted out on TDO indicates BIST successfully completed. A value of "1" indicates a failure occurred. The CLK clock must be running in order to execute RUNBIST. After executing the RUNBIST instruction, the Pentium processor must be reset prior to normal (non-boundary scan) operation.

HI-Z

The TAP Hi-Z instruction causes all outputs and I/Os of the Pentium processor to go to a high-impedance state (float) immediately. The Hi-Z state is terminated by either resetting the TAP with the TRST# pin, by issuing another TAP instruction, or by entering the Test_Logic_Reset state. The Hi-Z state is enabled or disabled on the first TCK clock after the TAP instruction has entered the UPDATE-IR state of the TAP control state machine. This instruction overrides all other bus cycles. Resetting the Pentium processor will not disable this instruction since CPU RESET does not reset the TAP.

BYPASS

The BYPASS instruction selects the Bypass Register to be connected to TDI and TDO. This effectively bypasses the test logic on the Pentium processor by reducing the shift length of the device to one bit. Note that an open circuit fault in the board level test data path will cause the Bypass Register to be selected following an instruction scan cycle due to a pull-up resistor on the TDI input. This was implemented to prevent any unwanted interference with the proper operation of the system logic.

12

Error Detection

CHAPTER 12
ERROR DETECTION

The Pentium processor incorporates a number of data integrity features that are focused on the detection and limited recovery of errors. The data integrity features in the Pentium processor provide capabilities for error detection of the internal devices and the external interface. The Pentium processor (75/90/100/120/133/150/166/200) also provides the capability to obtain maximum levels of error detection by incorporating Functional Redundancy Checking (FRC) support. Error detecting circuits in the Pentium processor do not limit the operating frequency of the chip.

The data integrity features in the Pentium processor can be categorized as (1) internal error detection, (2) error detection at the bus interface, and (3) FRC support.

12.1. INTERNAL ERROR DETECTION

Detection of errors of a majority of the devices in the Pentium processor is accomplished by employing parity checking in the large memory arrays of the chip. The data and instruction caches (both storage and tag arrays), translation lookaside buffers, and microcode ROM are all parity protected. The following describes the parity checking employed in the major memory arrays in the Pentium processor (MESI status bits are not parity protected):

- Parity bit per byte in the data cache storage array.
- Parity bit per entry in the data cache tag array.
- Four Parity bits: One for each of the even upper, even lower, odd upper, odd lower bits of an instruction cache line.
- Parity bit per entry in the instruction cache tag array.
- Parity bit per entry in both the data and instruction TLBs storage arrays.
- Parity bit per entry in both the data and instruction TLBs tag arrays.
- Parity bit per entry in the microcode ROM.

Parity checking as described above provides error detection coverage of 53% of the on-chip devices. This error detection coverage number also includes the devices in the branch target buffer since branch predictions are always verified.

If a parity error has occurred internally, then the Pentium processor operation can no longer be trusted. Normally, a parity error on a read from an internal array will cause the Pentium processor to assert the IERR# pin and then shutdown. (Shutdown will be entered assuming it is not prevented from doing so by the error.); however, if TR1.NS is set, IERR# will not result in processor shutdown. Execution will continue, but operation will not be reliable. Parity errors on reads during normal instruction execution, reads during a flush operation, reads during BIST and testability cycles, and reads during inquire cycles will cause IERR# to be asserted.

The IERR# pin will be asserted for one clock for each clock a parity error is detected and may be latched by the system. The IERR# pin is a glitch free signal, so no spurious assertions of IERR# will occur.

In general, internal timing constraints of the Pentium processor do not allow the inhibition of writeback cycles caused by inquire cycles, FLUSH# assertion or the WBINVD instruction when a parity error is encountered. In those cases where an internal parity error occurred during the generation of a writeback cycle, and that cycle was not able to be inhibited, the IERR# pin can be used to recognize that the writeback should be ignored. If an internal parity error occurs during a flush operation, the Pentium processor will assert the IERR# pin as stated above, and the internal caches will be left in a partially flushed state. The flush, flush acknowledge, or writeback special cycles will not be run.

12.2. ERROR DETECTION AT PENTIUM® PROCESSOR INTERFACE

The Pentium processor provides parity checking on the external address and data buses. There is one parity bit for each byte of the data bus and one parity bit for bits A31-A5 of the address bus.

12.2.1. Address Parity

A separate and independent mechanism is used for parity checking on the address bus during inquire cycles. Even address parity is driven along with the address bus during all Pentium processor initiated bus cycles and checked during inquire cycles. When the Pentium processor is driving the address bus, even parity is driven on the AP pin. When the address bus is being driven into the Pentium processor during an inquire cycle, this pin is sampled in any clock in which EADS# is sampled asserted. APCHK# is driven with the parity status two clocks after EADS# is sampled active. The APCHK# output (when active) indicates that a parity error has occurred on the address bus during an inquire. Figure 12-1 depicts an address parity error during an inquire cycle. For additional timing diagrams which show address parity, see the Bus Functional Description chapter. The APCHK# pin will be asserted for one clock for each clock a parity error is detected and may be latched by the system. The APCHK# pin is a glitch free signal, so no spurious assertions of APCHK# will occur.

In the event of an address parity error during inquire cycles, the internal snoop will not be inhibited. If the inquire hits a modified line in this situation and an active AHOLD prevents the Pentium processor from driving the address bus, the Pentium processor will potentially writeback a line at an address other than the one intended. If the Pentium processor is not driving the address bus during the writeback cycle, it is possible that memory will be corrupted.

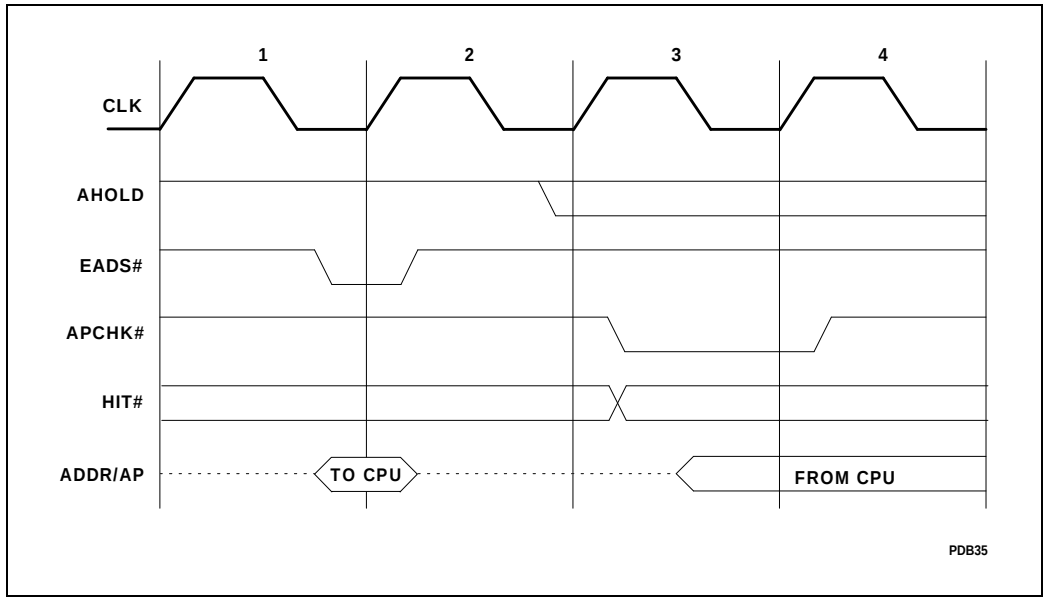


Figure 12-1. Inquire Cycle Address Parity Checking

Driving APCHK# is the only effect that bad address parity has on the Pentium processor. It is the responsibility of the system to take appropriate action if a parity error occurs. If parity checks are not implemented in the system, the APCHK# pin may be ignored.

12.2.2. Data Parity

Even data parity is driven on the DP7-DP0 pins in the same clock as the data bus is driven during all Pentium processor initiated data write cycles. During reads, even parity information may be driven back to the Pentium processor on the data parity pins along with the data being returned. Parity status for data sampled is driven on the PCHK# pin two clocks after the data is returned. PCHK# is driven low if a data parity error was detected, otherwise it is driven high. The PCHK# pin will be asserted for one clock for each clock a parity error is detected and may be latched by the system. The PCHK# pin is a glitch free signal, so no spurious assertions of PCHK# will occur. Figure 12-2 shows when the data parity (DP) pins are driven/sampled and when the PCHK# pin is driven. For additional timing diagrams that show data parity, see the Bus Functional Description chapter (Chapter 6).

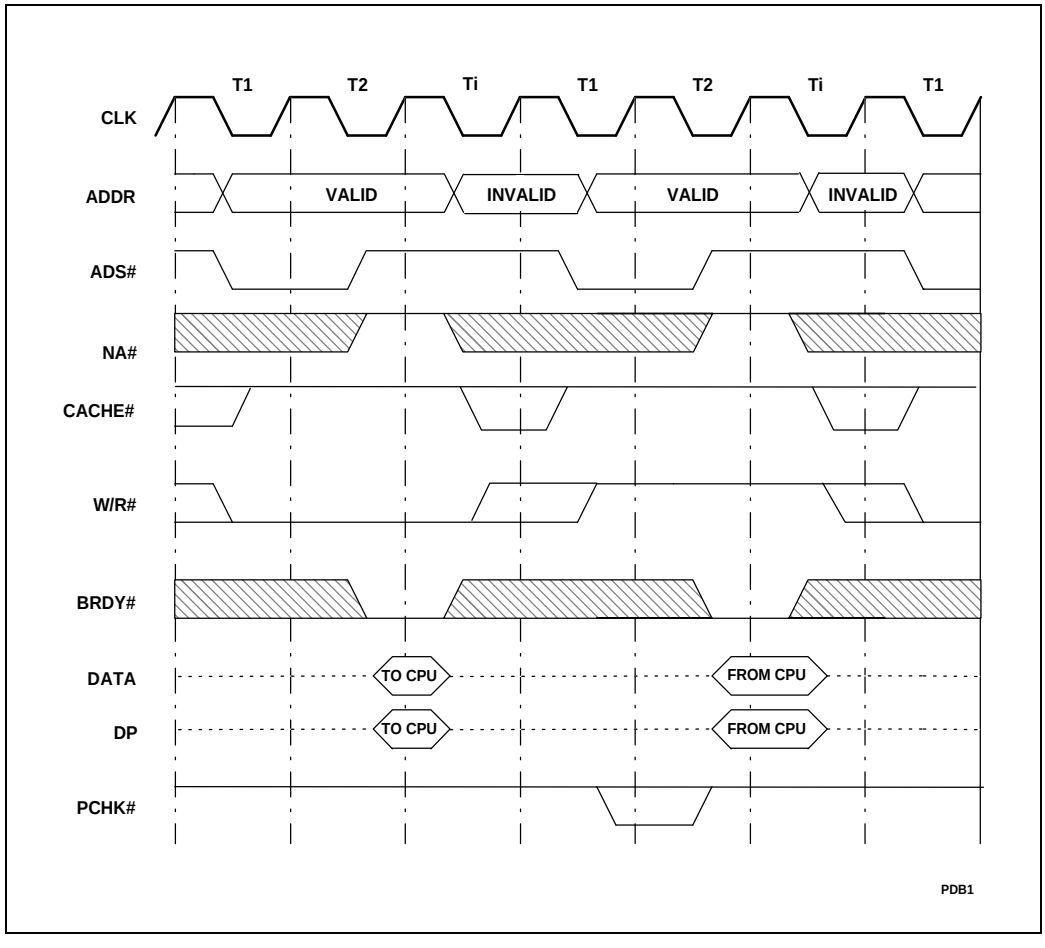


Figure 12-2. Data Parity During a Read and Write Cycle

Driving PCHK# is the only effect that bad data parity has on the Pentium processor. It is the responsibility of the system to take appropriate action if a parity error occurs. If parity checks are not implemented in the system, the PCHK# pin may be ignored.

12.2.2.1. MACHINE CHECK EXCEPTION AS A RESULT OF A DATA PARITY ERROR

The PEN# input determines whether a machine check interrupt will be taken as a result of a data parity error. If a data parity error occurs on a read for which PEN# was asserted, the physical address and cycle information of the cycle causing the parity error will be saved in the Machine Check Address Register and the Machine Check Type Register. If in addition, the

CR4.MCE is set to 1, the machine check exception is taken. The “Machine Check Exception” section provides more information on the machine check exception.

The parity check pin, PCHK#, is driven as a result of read cycles regardless of the state of the PEN# input.

12.2.3. Machine Check Exception

As mentioned in the earlier section, a new exception has been added to the Pentium processor. This is the machine check exception which resides at interrupt vector 18 (decimal). In processors previous to the Pentium processor, interrupt vector 18 was reserved and, therefore, there should be no interrupt routine located at vector 18. For reasons of compatibility, the MCE bit of the CR4 register will act as the machine check enable bit. When set to “1,” this bit will enable the generation of the machine check exception. When reset to “0,” the processor will inhibit generation of the machine check exception. CR4.MCE will be cleared on processor reset. In the event that a system is using the machine check interrupt vector for another purpose and the Machine Check Exception is enabled, the interrupt routine at vector 18 must examine the state of the CHK bit in the Machine Check Type register to determine the cause of its activation. Note that at the time the system software sets CR4.MCE to 1, it must read the Machine Check Type register in order to clear the CHK bit.

The Machine Check Exception is an abort, that is, it is not possible to reliably restart the instruction stream or identify the instruction causing the exception. Therefore, the exception does not allow the restart of the program that caused the exception. The Pentium processor does not generate an error code for this exception. Since the machine check exception is synchronous to a bus cycle and not an instruction, the IP pushed on to the stack may not be pointing to the instruction which caused the failing bus cycle.

The Machine Check Exception can be caused by one of two events: 1) Detection of data parity error during a read when the PEN# input is active, or 2) The BUSHCK# input being sampled active. When either of these events occur, the cycle address and type will be latched into the Machine Check Address (MCA) and Machine Check Type (MCT) registers (independent of the state of the CR4.MCE bit). If in addition, the CR4.MCE is “1,” a machine check exception will occur. When the MCA and MCT registers are latched, the MCT.CHK bit is set to “1” indicating that their contents are valid (Figure 12-3).

The Machine Check Address register, and the Machine Check Type register are model specific, read only registers. The Machine Check Address register is a 64-bit register containing the physical address for the cycle causing the error. The Machine Check Type register is a 64-bit register containing the cycle specification information, as defined in Figure 12-3. These registers are accessed using the RDMSR instruction. When the MCT.CHK is zero, the contents of the MCT and MCA registers are undefined. When the MCT register is read (using the RDMSR instruction), the CHK bit is reset to zero. Therefore, software must read the MCA register before reading the MCT register.

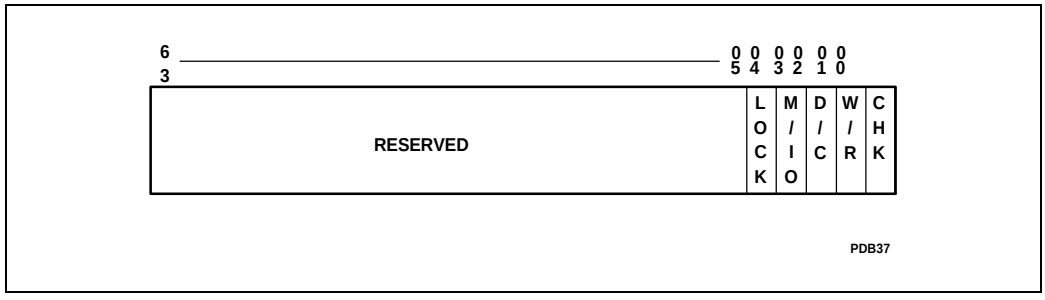


Figure 12-3. Machine Check Type Register

The bits in the Machine Check Type Register are defined as follows:

CHK: This bit is set to 1 when the Machine Check Type register is latched and is reset to 0 after the Machine Check Type register is read via the RDMSR instruction. In the event that the Machine Check Type register is latched in the same clock in which it is read, the CHK bit will be set. The CHK bit is reset to "0" on assertion of RESET. When the CHK bit is "0," the contents of the MCT and MCA registers are undefined.

M/IO#, D/C#, W/R#: These cycle definition pins can be decoded to determine if the cycle in error was a memory or I/O cycle, a data or code fetch, and a read or a write cycle. (See Table 4-2 for Quick pin reference.)

LOCK: Set to "1" if LOCK# is asserted for the cycle

12.2.4. Bus Error

The BUSCHK# input provides the system a means to signal an unsuccessful completion of a bus cycle. This signal is sampled on any edge in which BRDY# is sampled, for reads and writes. If this signal is sampled active, then the cycle address and type will be latched into the Machine Check Address and Machine Check Type registers. If in addition, the CR4.MCE bit is set to 1, the processor will be vectored to the machine check exception.

Even if BUSCHK# is asserted in the middle of a cycle, BRDY# must be asserted the appropriate number of clocks required to complete the bus cycle. The purpose of BUSCHK# is to act as an indication of an error that is synchronous to bus cycles. If the machine check interrupt is not enabled, i.e., the MCE bit in the CR4 register is zero, then an assertion of BUSCHK# will not cause the processor to vector to the machine check exception.

The Pentium processor can remember only one machine check exception at a time. This exception is recognized on an instruction boundary. If BUSCHK# is sampled active while servicing the machine check exception for a previous BUSCHK#, it will be remembered by the processor until the original machine check exception is completed. It is then that the processor will service the machine check exception for the second BUSCHK#. Note that only one

BUSCHK# will be remembered by the processor while the machine exception for the previous one is being serviced.

For use of BUSCHK# with STPCLK#, please refer to Table 4-2.

When the BUSCHK# is sampled active by the processor, the cycle address and cycle type information for the failing bus cycle is latched upon assertion of the last BRDY# of the bus cycle. The information is latched into the Machine Check Address and Machine Check Type registers respectively. However, if the BUSCHK# input is not deasserted before the first BRDY# of the next bus cycle, and the machine check exception for the first bus cycle has not occurred, then new information will be latched into the MCA and MCT registers, over-writing the previous information at the completion of this new bus cycle. Therefore, in order for the MCA and MCT registers to report the correct information for the failing bus cycle when the machine check exception for this cycle is taken at the next instruction boundary, the system must deassert the BUSCHK# input immediately after the completion of the failing bus cycle and before the first BRDY# of the next bus cycle is returned.

12.2.5. Functional Redundancy Checking

Functional Redundancy Checking (FRC) in the Pentium processor will provide maximum error detection (>99%) of on-chip devices and the processor's interface. A "checker" Pentium processor that executes in lock step with the "master" Pentium processor is used to compare output signals every clock. Note, the Pentium processor with MMX technology does not support FRC. Also, FRC is not supported in Dual processor designs.

Two Pentium processors are required to support FRC. Both the master and checker must be of the same stepping and same bus fraction. The Pentium processor configured as a master operates according to bus protocol described in this document. The outputs of the checker Pentium processor are tristated (except IERR#, TDO, PICD0, PICD1—however, these signals are not part of FRC) so the outputs of the master can be sampled. If the sampled value differs from the value computed internally by the checker, the checker asserts the IERR# output to indicate an error. A master-checker pair should have all pins except FRCMC#, IERR#, PICD0, PICD1 and TDO tied together.

The Pentium processors are configured either as a master or a checker by driving the FRCMC# input to the appropriate level while RESET is asserted. If sampled low during reset, the Pentium processor enters checker mode and tristates all outputs except IERR# and TDO (IERR# is driven inactive during reset). This feature is provided to prevent bus contention before reset is completed. The final master/checker configuration is determined when RESET transitions from high to low. The final master/checker configuration may not be changed other than by a subsequent RESET.

The IERR# pin reflects the result of the master-checker comparison. It is asserted for one clock, two clocks after the mismatch. It is asserted for each detected mismatch, so IERR# may be low for more than one consecutive clock. During the assertion of RESET, IERR# will be driven inactive. After RESET is deasserted, IERR# will not be asserted due to a mismatch until two clocks after the ADS# of the first bus cycle (i.e., in the third clock of the first bus cycle). IERR# will reflect pin comparisons thereafter. Note that IERR# may be asserted due to an

internal parity error prior to the first bus cycle. It is possible for FRC mismatches to occur in the event that an undefined processor state is driven off-chip, therefore no processor state should be stored without having been previously initialized.

In order for the master-checker pair to operate correctly, the system must be designed such that the master and the checker sample identical input states in the same clock. All asynchronous inputs should change state in such a manner that both the master and checker sample them in the same state in the same clock. The simplest way to do this is to design all asynchronous inputs to be synchronously controlled.

The TDO pin is not tested by FRC since it operates on a separate clock. Note that it is possible to use boundary scan to verify the connection between the master and checker by scanning into one, latching the outputs of the other and then scanning out. The Stop Clock state feature cannot be used in dual processing or functional redundancy checking modes because there is no way to re-synchronize the internal clocks of the two processors.

Figure 12-4 illustrates the configuration of output pins with respect to FRC. The comparators at each output compare the value of the package pin with the value being driven from the core to that pin, not the value driven by boundary scan to that pin. Therefore, during the use of boundary scan, FRC mismatches (IERR# assertion) can be expected to occur.

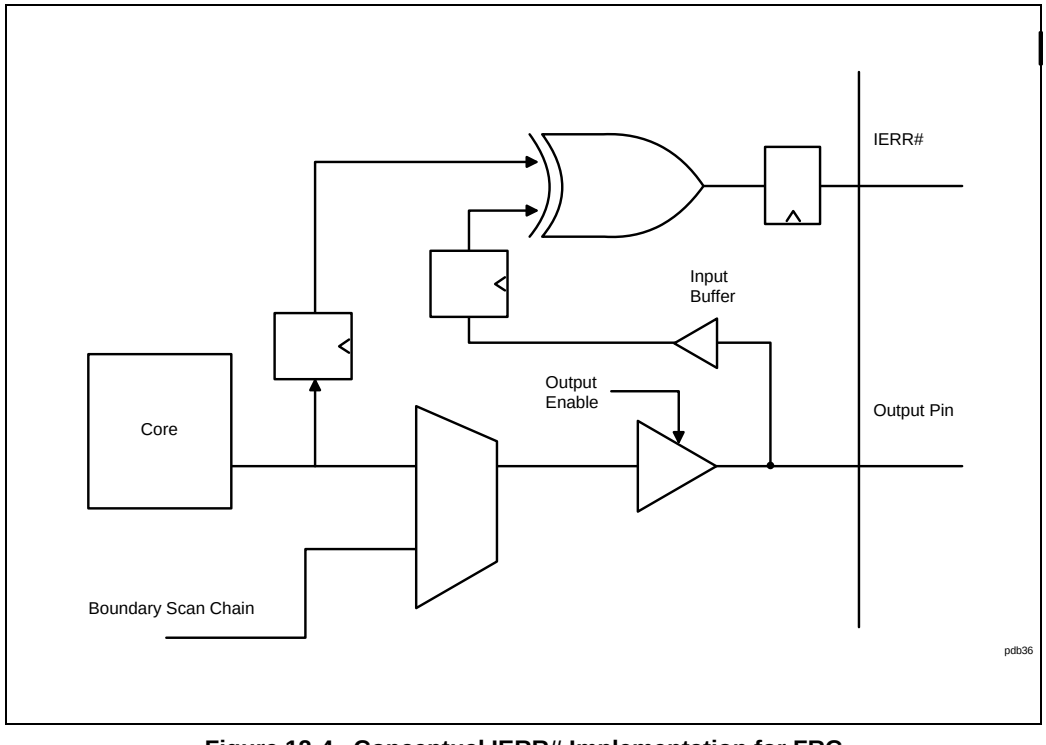


Figure 12-4. Conceptual IERR# Implementation for FRC

13

Execution Tracing

CHAPTER 13
EXECUTION TRACING

13.1. EXECUTION TRACING

The Pentium processor family uses special bus cycles to support execution tracing. These bus cycles, which are optional, have a significant impact on overall performance. Execution tracing allows the external hardware to track the flow of instructions as they execute inside the processor.

The special bus cycles generated by the Pentium processor family are Branch Trace Messages. Due to physical limitations, the maximum number of outstanding taken branches allowed is two. Once the second taken branch reaches the last stage of the pipeline, execution is stalled until the first branch message is sent on the bus.

Branch trace messages may be enabled by setting the Execution Tracing bit, TR, of TR12 (bit 1) to a 1. Once enabled, there are two forms of branch trace messages: normal and fast. Normal messages produce two cycles, one for the branch target linear address, and one for the linear address of the instruction causing the taken branch. Fast messages only produce the second of these two cycles. The second message will always contain the linear address of the instruction executed in the u pipe even if the instruction that caused the branch was executed in the v pipe. For serializing instructions and segment descriptor loads the address field of the first cycle will contain the address of the next sequential instruction after the instruction that caused the BTM. Fast execution tracing is enabled by setting bit 8 of TR12 to 1. Note that switching between the normal and fast formats by using the WRMSR instruction to change bit 8 of TR12, the WRMSR instruction causes a branch trace message when they are enabled. The format for this branch trace message will be the format that was programmed **before** the WRMSR instruction was executed.

Normal and fast branch trace messages may be delayed by 0 or more clocks after the cycle in which the branch was taken depending on the bus activity. Also, higher priority cycles may be run between the first and second cycles of a normal branch trace message. In dual-processor mode, branch trace message cycles may be interleaved with cycles from the other processor. Branch trace message cycles are buffered so they do not normally stall the processor.

Branch trace messages, normal and fast, may be identified by the following special cycle:

M/IO# = 0

D/C# = 0

W/R# = 1

BE [7:0]# = 0DFh

The address and data bus fields for the two bus cycles associated with a branch trace message are defined below:

First Cycle (Normal)

A31 - A4	Bits 31 - 4 of the branch target linear address
A3	“1” if the default operand size is 32 bits
	“0” if the default operand size is 16 bits
D63 - D60	Bits 3 - 0 of the branch target linear address
D59	“0” - indicating the first of the two cycles
D58-D0	Reserved. Driven to a valid state, but must be ignored

Second Cycle (Normal)

A31 - A4	Bits 31 - 4 of the linear address of the instruction causing the taken branch
A3	“1” if the default operand size is 32 bits
	“0” if the default operand size is 16 bits
D63 - D60	Bits 3 - 0 of the linear address of the instruction causing the taken branch
D59	“1” - indicating the second of the two cycles
D58-D0	Reserved. Driven to a valid state, but must be ignored

Fast Cycle

A31 - A4	Bits 31 - 4 of the linear address of the instruction causing the taken branch
A3	“1” if the default operand size is 32 bits
	“0” if the default operand size is 16 bits
D63 - D60	Bits 3 - 0 of the linear address of the instruction causing the taken branch
D59	Driven to a “1”
D58-D0	Reserved. Driven to a valid state, but must be ignored

In addition to conditional branches, jumps, calls, returns, software interrupts, and interrupt returns, the Pentium processor family treats the following operations as causing taken branches:

- Sserializing instructions
- Some segment descriptor loads
- Hardware interrupts
- Masked floating point exceptions and all other exceptions that invoke a trap or fault handler
- Exiting the HALT state

With execution tracing enabled, these operations will also cause a corresponding branch trace message cycle. The Pentium processor data bus is valid during branch trace message special cycles. Instructions which cause masked floating point exceptions may cause one or more branch trace special cycles. This is because execution of an instruction may be aborted and restarted several times due to the exception.

Also note that the WRMSR instruction to enable branch trace messages will cause a BTM to be generated (WRMSR is a serializing instruction and serializing instructions cause BTMs). A WRMSR to disable BTMs will not generate a BTM. Conditions which cause the VERR, VERW, LAR and LSL instruction to clear the ZF bit in EFLAGS will also cause these instructions to be treated as taken branches. However, if these instructions fail the protection checks, no branch trace message will be generated.

Note that if an instruction faults, it does not complete execution but instead is flushed from the pipeline and an exception handler is invoked. This faulting instruction effectively causes a branch; a branch trace message is generated accordingly.

Refer to Chapter 16 for detailed information on model specific registers.

14

Power Management

CHAPTER 14
POWER MANAGEMENT

The Pentium processor family implements Intel's System Management Mode (SMM) architecture. This chapter describes the hardware interface to SMM and Clock Control. For a overall architectural description, refer to Chapter 3 of this manual, and also the *Intel Architecture Software Developer's Manual.*

14.1. PENTIUM® PROCESSOR FAMILY POWER MANAGEMENT FEATURES

- System Management Interrupt can be delivered through the SMI# signal or through the local APIC using the SMI# message, which enhances the SMI interface, and provides for SMI delivery in APIC-based Pentium processor dual processing systems.
- In dual processing systems, SMIACT# from the bus master (MRM) behaves differently than in uniprocessor systems. If the LRM processor is the CPU in SMM mode, SMIACT# will be inactive and remain so until that processor becomes the MRM.
- The Pentium processor is capable of supporting an SMM I/O instruction restart. This feature is automatically disabled following RESET. To enable the I/O instruction restart feature, set bit 9 of the TR12 register to "1".
- The Pentium processor default SMM revision identifier has a value of 2 when the SMM I/O instruction restart feature is enabled.
- SMI# is NOT recognized by the Pentium processor in the shutdown state.

14.2. SYSTEM MANAGEMENT INTERRUPT PROCESSING

The system interrupts the normal program execution and invokes SMM by generating a System Management Interrupt (SMI#) to the processor. The processor will service the SMI# by executing the following sequence. See Figure 14-1.

1. Wait for all pending bus cycles to complete and EWBE# to go active.
2. The CPU asserts the SMIACT# signal while in SMM indicating to the system that it should enable the SMRAM.
3. The CPU saves its state (context) to SMRAM, starting at address location SMBASE + 0FFFFH, proceeding downward in a stack-like fashion.
4. The CPU switches to the System Management Mode processor environment (a pseudo-real mode).

5. The CPU will then jump to the absolute address of SMBASE + 8000H in SMRAM to execute the SMI handler. This SMI handler performs the system management activities.
6. The SMI handler will then execute the RSM instruction which restores the CPU's context from SMRAM, de-asserts the SMIACT# signal, and then returns control to the previously interrupted program execution.

NOTE

The default SMBASE value following RESET is 30000H.

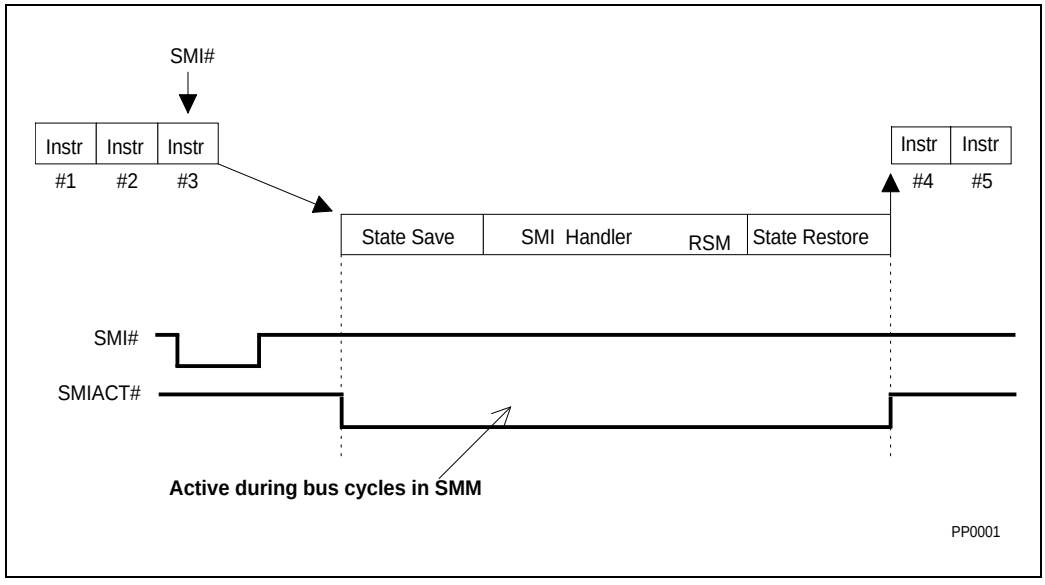


Figure 14-1. Basic SMI# Interrupt Service

Figure 14-2 describes the System Management Interrupt hardware interface which consists of the SMI# interrupt request input and the SMIACT# output used by the system to decode the SMRAM.

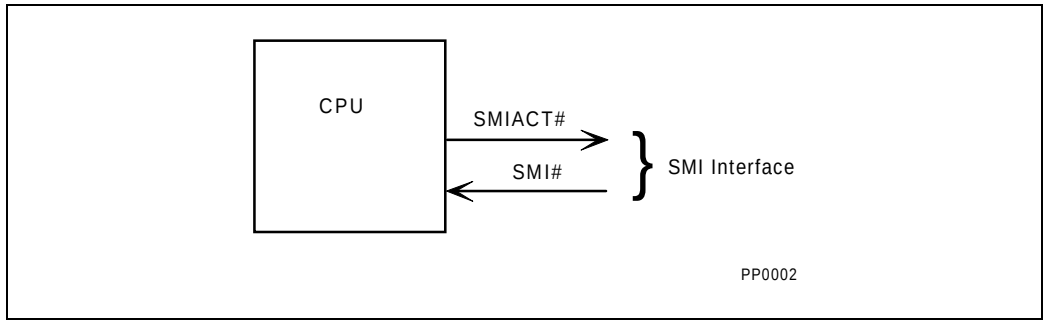


Figure 14-2. Basic SMI# Hardware Interface

14.2.1. System Management Interrupt (SMI#)

SMI# is a falling-edge triggered, non-maskable interrupt request signal. SMI# is an asynchronous signal, but setup and hold times, t28 and t29, (see Chapter 7) must be met in order to guarantee recognition on a specific clock. The SMI# input need not remain active until the interrupt is actually serviced. The SMI# input only needs to remain active for a single clock if the required setup and hold times are met. SMI# will also work correctly if it is held active for an arbitrary number of clocks.

The SMI# signal is synchronized internally and must be asserted at least three (3) CLK periods prior to asserting the BRDY# signal in order to guarantee recognition on a specific instruction boundary. See Figure 14-3.

The SMI# input must be held inactive for at least four clocks after it is asserted to reset the edge triggered logic. A subsequent SMI# might not be recognized if the SMI# input is not held inactive for at least four clocks after being asserted.

SMI#, like NMI, is not affected by the IF bit in the EFLAGS register and is recognized on an instruction boundary. An SMI# will not break locked bus cycles. The SMI# has a higher priority than NMI and is not masked during an NMI.

After the SMI# interrupt is recognized, the SMI# signal will be masked internally until the RSM instruction is executed and the interrupt service routine is complete. Masking the SMI# prevents recursive SMI# calls. If another SMI# occurs while the SMI# is masked, the pending SMI# will be recognized and executed on the next instruction boundary after the current SMI# completes. This instruction boundary occurs before execution of the next instruction in the interrupted application code, resulting in back to back SMM handlers. Only one SMI# can be pending while SMI# is masked.

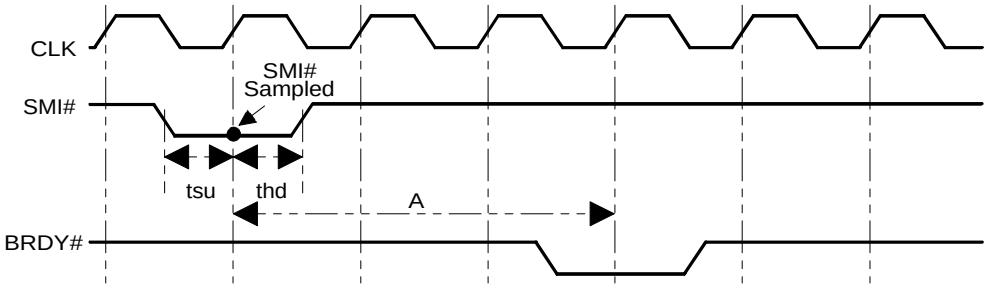


Figure 14-3. SMI# Timing

14.2.1.1. SMI# Synchronization for I/O Instruction Restart

The SMI# signal is synchronized internally and must be asserted at least three (3) CLK periods prior to asserting the BRDY# signal in order to guarantee recognition on a specific I/O instruction boundary. This is important for servicing an I/O trap with an SMI# handler. Due to the asynchronous nature of SMI# delivery with the APIC, it is impossible to synchronize the assertion of BRDY#. As a result, the SMM I/O instruction restart feature cannot be used when an SMI is delivered via the local APIC.

14.2.1.2. DUAL PROCESSING CONSIDERATIONS FOR SMI# DELIVERY

Although the SMM functions the same when the Dual processor is inserted into Socket 7, the dual processor operation of the system must be carefully considered. Table 14-1 shows the four possible options for SMI# delivery depending on the SMM applications (mainly power management) the system has to support. There are implications to system design and the SMM handler. Note that for operation with the Dual processor and upgradability with the future Pentium OverDrive processor, Option #3 is strongly recommended.

Table 14-1. Dual Processing SMI# Delivery Options

	SMI# Pins Tied Together	SMI# Pins NOT Tied Together
SMI# pins delivering SMI	Option #1 Both CPUs enter SMM.	Option #2 One CPU enters SMM.
APIC delivering SMI	**Option #3** **One or Both CPUs enter SMM.**	Option #4 One or Both CPUs enter SMM

NOTE:

The I/O Instruction Restart Power Management feature should not be used when delivering the system management interrupt via the local APIC. Refer to the *Intel Architecture Software Developer's Manual, Volume 3* for additional details on I/O instruction restart.

Implications

1. SMI# pin delivery of SMI with the SMI# pins tied together: Any assertion of the SMI# pin will cause both the Primary and Dual processors to interrupt normal processing, enter SMM mode and start executing SMM code in their respective SMRAM spaces. In this case, using the I/O Instruction restart feature in Dual Processor mode will require additional system hardware (D/P# pin) and software (detection of which processor was the MRM when the SMI# pin was asserted) considerations. This option will work for the future Pentium OverDrive processor.
2. SMI# pin delivery of SMI with the SMI# pins NOT tied together: Only the processor whose SMI# pin is asserted will handle SMM processing. It is possible that both the Primary and Dual processor will be doing SMM processing at the same time, especially if the I/O Instruction restart feature is being used. If I/O instruction restart is not supported, then it is possible to dedicate only one processor for SMM handling at any time. This option is not recommended for future Pentium OverDrive processor compatibility.
3. APIC SMI# delivery of SMI with the SMI# pins tied together: This option is strongly recommended for operation with the Dual processor and upgradability with the future Pentium OverDrive processor. System Management Interrupts should be delivered via the APIC for DP systems, and may be delivered either via the APIC or the SMI# pin for turbo-upgraded systems. Either the Primary or Dual processor can be the assigned target for SMI# delivery and hence SMM handling. The SMM I/O instruction restart feature may be used in a uniprocessor system or in a system with a future Pentium OverDrive processor (with SMI# pin delivery of the interrupt), but the system must not use this feature when operating in dual processing mode (with APIC delivery of the interrupt).
4. APIC SMI# delivery of SMI with the SMI# pins NOT tied together: I/O Instruction Restart feature is not recommended when delivering SMI via the local APIC. Either the Primary or Dual processor can be the assigned target for SMI# delivery and hence SMM handling. This option is not recommended for future Pentium OverDrive processor compatibility.

14.2.2. SYSTEM MANAGEMENT INTERRUPT VIA APIC

When SMI# is asserted (SMI# pin asserted low or APIC SMI# message) it causes the processor to invoke SMM.

14.2.3. SMI Active (SMIACT#)

SMIACT# indicates that the CPU is operating in System Management Mode. The CPU asserts SMIACT# in response to an SMI interrupt request on the SMI# pin or through the APIC message. SMIACT# is driven active for accesses only after the CPU has completed all pending write cycles (including emptying the write buffers — EWBE# returned active by the system). SMIACT# will be asserted for all accesses in SMM beginning with the first access to SMRAM when the CPU saves (writes) its state (or context) to SMRAM. SMIACT# is driven active for every access until the last access to SMRAM when the CPU restores (reads) its state from SMRAM. The SMIACT# signal is used by the system logic to decode SMRAM.

NOTE

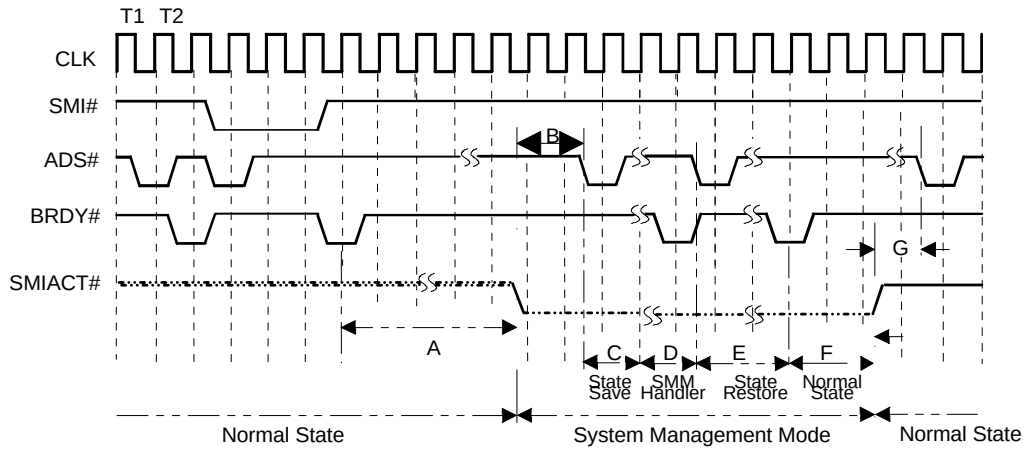
The number of CLKs required to complete the SMM state save and restore is very dependent on system memory performance and the CPU bus frequency.

As shown in Figure 14-4, the approximate time required to enter an SMI handler routine for the Pentium processor (from the completion of the interrupted instruction) is given by:

Latency to beginning of SMI handler = A + B + C = ~184 CLKs

The approximate time required to return to the interrupted application (following the final SMM instruction before RSM) is given by:

Latency to continue interrupted application = E + F + G = ~207 CLKs



	Pentium® Processor
A: Last BRDY# from non-SMM transfer to SMIACT# assertion	2 CLKs minimum
B: SMIACT# assertion to first ADS# for SMM state save	2 CLKs minimum
C: SMM state save (dependent on memory performance)	Approximately 180 CLKs
D: SMM handler	User determined
E: SMM state restore (dependent on memory performance)	Approximately 200 CLKs
F: Last RDY# from SMM transfer to de-assertion of SMIACT#	2 CLKs minimum
G: SMIACT# de-assertion to first non-SMM ADS#	5 CLKs minimum

PP0004

Figure 14-4. SMIACT# Timing

14.2.3.1. Dual Processing Considerations for SMIACT#

When the Pentium processor is the only processor present, then it always drives the D/P# signal low. SMIACT# is asserted when the Pentium processor enters SMM and is de-asserted only when the Pentium processor exits SMM.

When the Dual processor is also present, the D/P# signal toggles depending upon whether the Primary or Dual processor owns the bus (MRM). The SMIACT# pins may be tied together or be used separately to insure SMRAM access by the correct processor.

CAUTION

If SMIACT# is used separately: the SMIACT# signal is only driven by the Primary or Dual processor when it is the MRM, so this signal must be qualified with the D/P# signal.

In a dual socket system, connecting the SMIACT# signals together on the Primary and Dual processor sockets is strongly recommended for both dual processing operation and upgradability with the Pentium OverDrive processor.

In dual processing systems, SMIACT# may not remain low (e.g., may toggle) if both processors are not in SMM mode. The SMIACT# signal is asserted by either the Primary or Dual processor based on two conditions: the processor is in SMM mode and is the bus master (MRM). If one processor is executing in normal address space, the SMIACT# signal will go inactive when that processor is MRM. The LRM processor, even if in SMM mode, will not drive the SMIACT# signal low.

14.3. SMM — SYSTEM DESIGN CONSIDERATIONS

14.3.1. SMRAM Interface

The hardware designed to control the SMRAM space must follow these guidelines:

1. A provision should be made to allow for initialization of SMRAM space during system boot up. This initialization of SMRAM space must happen before the first occurrence of an SMI# interrupt. Initializing the SMRAM space must include installation of an SMM handler, and may include installation of related data structures necessary for particular SMM applications. The memory controller providing the interface to the SMRAM should provide a means for the initialization code to manually open the SMRAM space.
2. A minimum initial SMRAM address space of SMBASE + 8000H to SMBASE + 0FFFFH should be decoded by the memory controller.
3. Alternate bus masters (such as DMA controllers) should not be allowed to access SMRAM space. Only the CPU, either through SMI or during initialization, should be allowed access to SMRAM.
4. In order to implement a zero-volt suspend function, the system must have access to all of normal system memory from within an SMM handler routine. If the SMRAM is going to overlay normal system memory, there must be a method of accessing any system memory that is located underneath SMRAM.
5. Inquire cycles are permitted during SMM, but it is the responsibility of the system to ensure that any snoop writeback completes to the correct memory space, irrespective of the state of the SMIACT# pin. Specifically, if SMM is overlaid, and SMM space is non cacheable, then any snoop writeback cycle occurring during SMM must complete to system memory, even though SMIACT# will remain active.

   If an inquire cycle occurs after assertion of SMI# to the processor, but before SMIACT# is returned, note that SMIACT# could be returned at any point during the snoop writeback cycle. Depending on the timing of SMI# and the inquire cycle, SMIACT# could change states during the writeback cycle. Again, it is the responsibility of the system, if it supports snooping during SMM, to ensure that the snoop writeback cycle completes to the correct memory space, irrespective of the state of the SMIACT# pin.

6. It should also be noted that upon entering SMM, the branch target buffer (BTB) is not flushed and thus it is possible to get a speculative prefetch to an address outside of SMRAM address space due to branch predictions based on code executed prior to entering SMM. If this occurs, the system must still return BRDY# for each code fetch cycle.

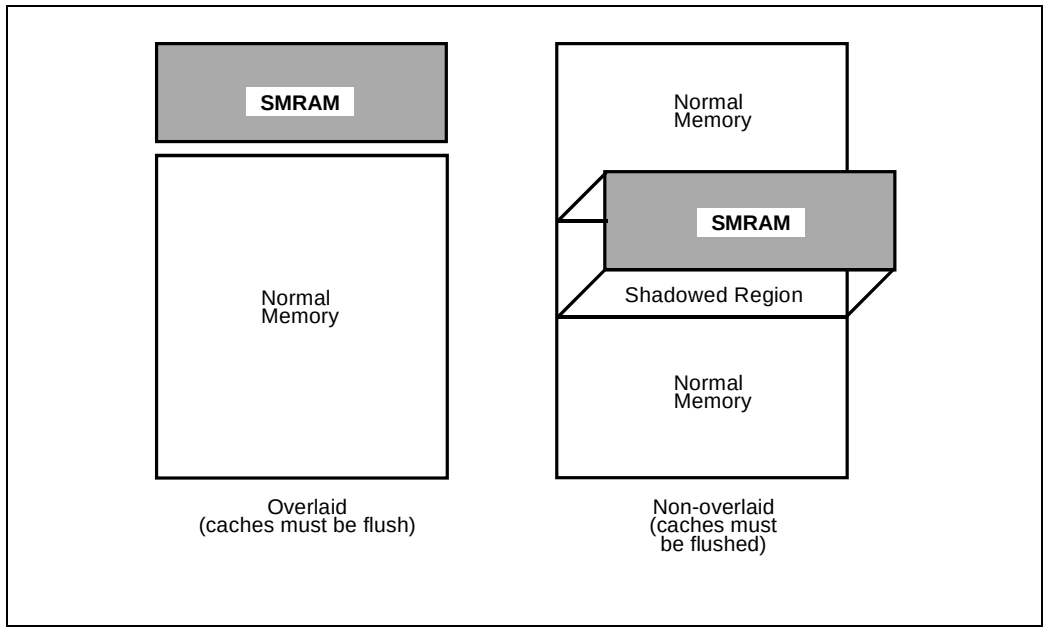


Figure 14-5. SMRAM Location

14.3.2. Cache Flushes

The Pentium processor does not unconditionally writeback and invalidate its cache before entering SMM (this option is left to the system designer). If the SMRAM is in an area that is cacheable and overlaid on top of normal memory that is visible to the application or operating system (default), then it is necessary for the system to flush both the CPU cache and any second level cache upon entering SMM. This may be accomplished by asserting flush the same time as the request to enter SMM (i.e., cache flushing during SMM entry is accomplished by asserting the FLUSH# pin at the same time as the request to enter SMM through SMI#). The priorities of FLUSH# and SMI# are such that the FLUSH# will be serviced first. To guarantee this behavior, the constraints on setup and hold timings on the interaction of FLUSH# and SMI# as specified for a processor should be obeyed. When the default SMRAM location is used, SMRAM is overlaid on top of system main memory (at SMBASE + 8000H to SMBASE + 0FFFFH).

In a system where FLUSH# and SMI# pins are synchronous and setup/hold times are met, then the FLUSH# and SMI# pins may be asserted in the same clock. In asynchronous systems, the

FLUSH# pin must be asserted at least one clock before the SMI# pin to guarantee that the FLUSH# pin is serviced first. Note that in systems that use the FLUSH# pin to write back and invalidate the cache contents before entering SMM, the Pentium processor will prefetch at least one cache line in between the time the Flush Acknowledge special cycle is run and the recognition of SMI# and the driving of SMIACT# for SMRAM accesses. It is the obligation of the system to ensure that these lines are not cached by returning KEN# inactive.

If SMRAM is located in its own distinct memory space, which can be completely decoded with only the CPU address signals, it is said to be non-overlaid. In this case, there is one new requirement for maintaining cache coherency, refer to Table 14-2 below.

Table 14-2. Scenarios for Cache Flushes with Writeback Caches

Is SMRAM overlapped with normal memory?	Is Normal Memory cacheable ?	Is SMRAM cacheable?	Flush required during SMM entry?	Flush required during SMM exit?	Comments
No	No	No	No	No	
	No	WT	No	No	
	WT	No	No	No	
	WB	No	No*	No	*Snoop WB's must always go to normal memory space
	WT	WT	No	No	
	WB	WT	No*	No	*Snoop and Replacement WB's must go to normal memory space.
Yes	No	No	No	No	
	No	WT	No	Yes	
	WT	No	Yes	No	
	WB	No	Yes	No	
	WT	WT	Yes	Yes	
	WB	WT	Yes	Yes	

NOTE:

Writeback cacheable SMRAM is not recommended. When flushing upon SMM exit, SMIACT# will be deasserted and may cause regular memory to be overwritten.

The Pentium processor implements writeback caches. Hence the performance hit due to flushing the cache for SMM execution can be more significant. Due to the writeback nature of the cache, flushing the cache has the following penalties:

1. Before entry into SMM (when SMRAM is cacheable), the cache has to be flushed. Hence, all dirty lines need to be written back. This may cause a large number of bus cycles and increase SMM entry latency.
2. If the cache had to be flushed upon SMM exit, execution starts with cache miss 100%. The cache fill cycles reduce performance.

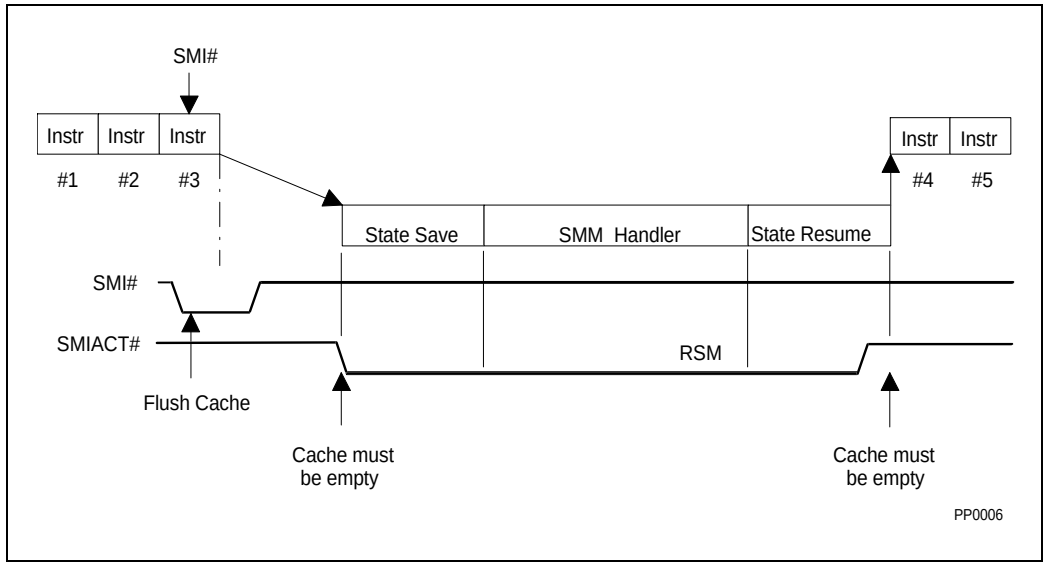


Figure 14-6. FLUSH# Mechanism During SMM with Overlay

The method suggested is shown in Figure 14-7.

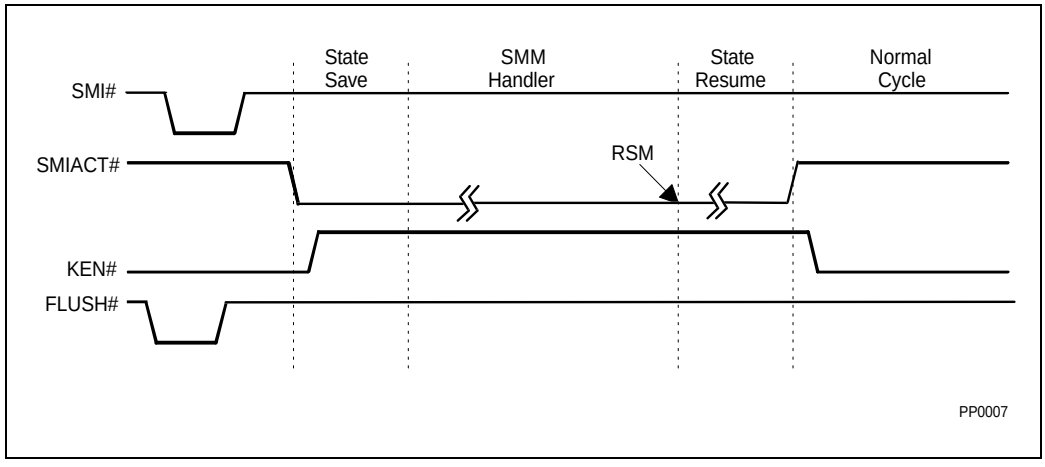


Figure 14-7. Flush with Non-Cached SMM with Overlay

14.3.2.1. Dual Processing Considerations for Cache Flushes

Cache flushing during SMM exit is not possible while both the Primary and Dual processors are present due to the fact that it is not possible to clearly predict when the CPU in SMM has exited. This is because the SMIACT# is not a static status indicator but only a bus cycle indicator for SMRAM accesses.

14.3.3. A20M# Pin

Systems based on the MS-DOS operating system contain a feature that enables the CPU address bit A20 to be forced to 0. This limits physical memory to a maximum of 1 Mbyte, and is provided to ensure compatibility with those programs that relied on the physical address wrap around functionality of the original IBM PC. The A20M# pin on the Pentium processor provides this function. When A20M# is active, all external bus cycles will drive A20 low, and all internal cache accesses will be performed with A20 low.

The A20M# pin is recognized while the CPU is in SMM. The functionality of the A20M# input must be recognized in two instances:

1. If the SMM handler needs to access system memory space above 1 Mbyte (for example, when saving memory to disk for a zero-volt suspend), the A20M# pin must be de-asserted before the memory above 1 Mbyte is addressed.
2. If SMRAM has been relocated to address space above 1 Mbyte, and A20M# is active upon entering SMM, the CPU will attempt to access SMRAM at the relocated address, but with A20 low. This could cause the system to crash, since there would be no valid SMM interrupt handler at the accessed location.

In order to account for the above two situations, the system designer must ensure that A20M# is de-asserted on entry to SMM. A20M# must be driven inactive before the first cycle of the SMM state save, and must be returned to its original level after the last cycle of the SMM state restore. This can be done by blocking the assertion of A20M# whenever SMIACT# is active.

14.3.4. SMM and Second Level Write Buffers

Before the Pentium processor enters SMM, it empties its internal write buffers. This is necessary so that the data in the write buffers is written to normal memory space, not SMM space. Once the CPU is ready to begin writing an SMM state save to SMRAM, it asserts the SMIACT# signal for SMRAM references. SMIACT# may be driven active by the CPU before the system memory controller has had an opportunity to empty the second level write buffers.

To prevent the data from these second level write buffers from being written to the wrong location, the system memory controller needs to direct the memory write cycles to either SMM space or normal memory space. This can be accomplished by saving the status of SMIACT# along with the address for each word in the write buffers.

EWBE# can also be used to prevent the CPU from asserting SMIACT# before write buffers are empty. The processor will wait for an active EWBE# before asserting SMIACT#.

14.4. CLOCK CONTROL

14.4.1. Clock Generation

To understand the additional power management fears of the Pentium processor and how it manipulates the clock to conserve power, it is necessary to understand how the clock operates. The Pentium processor is capable of running internally at frequencies much higher than the bus speed via the various bus frequency settings (see BF[1:0] pin settings in Chapter 4 or Chapter 5). This allows simpler system design by lowering the clock speeds required in the external system. The high frequency internal clock relies on an internal Phase Lock Loop (PLL) to generate the two internal clock phases, “phase one” and “phase two.” Most external timing parameters are specified with respect to the rising edge of CLK. The PLL requires a constant frequency CLK input, and therefore the CLK input cannot be changed dynamically.

On the Pentium processor, CLK provides the fundamental timing reference for the bus interface unit. The internal clock converter enhances all operations functioning out of the internal cache and/or operations not blocked by external bus accesses.

14.4.2. Stop Clock

The Pentium processor provides an interrupt mechanism, STPCLK#, that allows system hardware to control the power consumption of the CPU by stopping the internal clock (output

of the PLL) to the CPU core in a controlled manner. This low-power state is called the Stop Grant state. The target for low-power mode supply current in the Stop Grant state is ~15% of normal I_{CC}.

When the CPU recognizes a STPCLK# interrupt, the processor will stop execution on the next instruction boundary (unless superseded by a higher priority interrupt), stop the pre-fetch unit, complete all outstanding writes, generate a Stop Grant bus cycle, and then stop the internal clock. At this point, the CPU is in the Stop Grant state.

NOTE

If STPCLK# is asserted during RESET and continues to be held active after RESET is deasserted, the processor will execute one instruction before the STPCLK# interrupt is recognized. Execution of instructions will therefore stop on the second instruction boundary after the falling edge of RESET.

The CPU cannot respond to a STPCLK# request from a HLDA state because it cannot generate a Stop Grant cycle.

The rising edge of STPCLK# will tell the CPU that it can return to program execution at the instruction following the interrupted instruction.

Unlike the normal interrupts, INTR and NMI, the STPCLK# interrupt does not initiate interrupt table reads. Among external interrupts, STPCLK# is the lowest priority.

14.4.2.1. STPCLK# PIN

STPCLK# is treated as a level triggered interrupt to the Pentium processor. This interrupt may be asserted asynchronously and is prioritized below all of the external interrupts. If asserted, the Pentium processor will recognize STPCLK# on the next instruction boundary, and then do the following:

1. Flush the instruction pipeline of any instructions waiting to be executed.
2. Wait for all pending bus cycles to complete and EWBE# to go active.
3. Drive a special bus cycle (Stop Grant bus cycle) to indicate that the clock is being stopped.
4. Enter low power mode.

STPCLK# is active LOW. To ensure STPCLK# recognition, the system must keep this signal active until the appropriate special cycle has been issued by the Pentium processor. To guarantee that every STPCLK# assertion, and subsequent de-assertion and re-assertion, is recognized and thus will get a Stop Grant bus cycle response (which will also ensure that each de-assertion of STPCLK# allows execution of at least one instruction), the system must meet the following requirements:

1. Hold STPCLK# active at least until the processor's Stop Grant cycle response has been completed by the system's BRDY# response.

2. STPCLK# must not be re-asserted until 5 clocks after the **last** of the following events:
   a. The processor's Stop Grant cycle has been completed by the system's BRDY# response.
   b. HITM# is de-asserted. (This applies only if HITM# was asserted while waiting for one of the other two events listed here, **or** within 2 bus clocks of their completion.)
   c. EWBE# becomes active after it was sampled inactive at the last relevant BRDY#. A relevant BRDY# is one which ends **either** a stop-grant cycle **or** an external snoop writeback caused by HITM# being asserted as in case b) above.

Events b) and c) can in principle alternate indefinitely, continuing to delay STPCLK# de-assertion recognition, if the system design allows that to happen.

Note that if a system is not relying on either a Stop Grant bus cycle response for every STPCLK# assertion, or for each de-assertion of STPCLK# to allow execution of at least one instruction, these detailed requirements can be ignored. Though STPCLK# is asynchronous, setup and hold times (Refer to Chapter 7) may be met to ensure recognition on a specific clock.

The STPCLK# input must be driven high (not floated) in order to exit the Stop Grant state. Once STPCLK# is deasserted and the Pentium processor resumes execution, the Pentium processor is guaranteed to execute at least one instruction before STPCLK# is recognized again. To return to normal state, external hardware must deassert STPCLK#.

14.4.2.2. DUAL PROCESSING CONSIDERATIONS

The Primary and Dual processors may or may not tie their STPCLK# signals together. The decision is dependent on system specific CPU power conservation needs. Connecting the STPCLK# signals on the Primary and Dual processors together is strongly recommended for operation with the Dual processor and upgradability with the future Pentium OverDrive processor.

Tying the STPCLK# signals together causes both the Primary and Dual processors to eventually enter the Stop Grant state on assertion of STPCLK#. The system ceases processing until the STPCLK# signal is deasserted. In Dual Processor mode with the STPCLK# pins tied together, independent STPCLK# control of each processor is not possible. Both the Primary processor and Dual processor will go into the Stop Grant state independently, and will each generate a Stop Grant special bus cycle.

NOTE

In a dual processing system where STPCLK# is tied to both the primary and dual processors, the system expects to see two Stop Grant Bus Cycles after STPCLK# is asserted. FLUSH# should not be asserted between the time STPCLK# is asserted and the completion of the second Stop Grant Bus Cycle. If FLUSH# is asserted during this interval, the system may not see the second Stop Grant Bus Cycle until after STPCLK# is deasserted.

Not tying the STPCLK# signals together gives the flexibility to control either or both the processors' power consumption based on the system performance required. External logic would be required to control this signal to each processor in a DP system. In order to be upgradable with the future Pentium OverDrive processor, system-level logic would be required (and be end-user invisible) to allow the STPCLK# signal to operate properly with both the Pentium processor (in a non-upgraded system) and with the future Pentium OverDrive processor (in an upgraded system).

14.4.3. Stop Grant Bus Cycle

A special Stop Grant bus cycle will be driven to the bus after the processor recognizes the STPCLK# interrupt. The definition of this bus cycle is the same as the HALT cycle definition for the standard Intel486 microprocessor architecture, with the exception that the Stop Grant bus cycle drives the value 0000 0010H on the address pins. In a Dual Processor system, with both STPCLK# signals tied together, two stop grant cycles will occur in a row. The system hardware must acknowledge the Stop Grant cycle by returning BRDY#. The processor will not enter the Stop Grant state until BRDY# has been returned.

The Stop Grant Bus cycle consists of the following signal states: M/IO# = 0, D/C# = 0, W/R# = 1, Address Bus = 0000 0010H (A4 = 1), BE7#-BE0# = 1111 1011, Data bus = undefined.

NOTE

When operating in dual processing mode, and the STPCLK# signals are tied together, both the Primary processor and Dual processor will go into the Stop Grant state independently, and will each generate a Stop Grant special bus cycle. The system must return BRDY# for both of the special bus cycles.

The latency between a STPCLK# request and the Stop Grant bus cycle is dependent on the current instruction, the amount of data in the CPU write buffers, and the system memory performance. Refer to Figure 14-8.

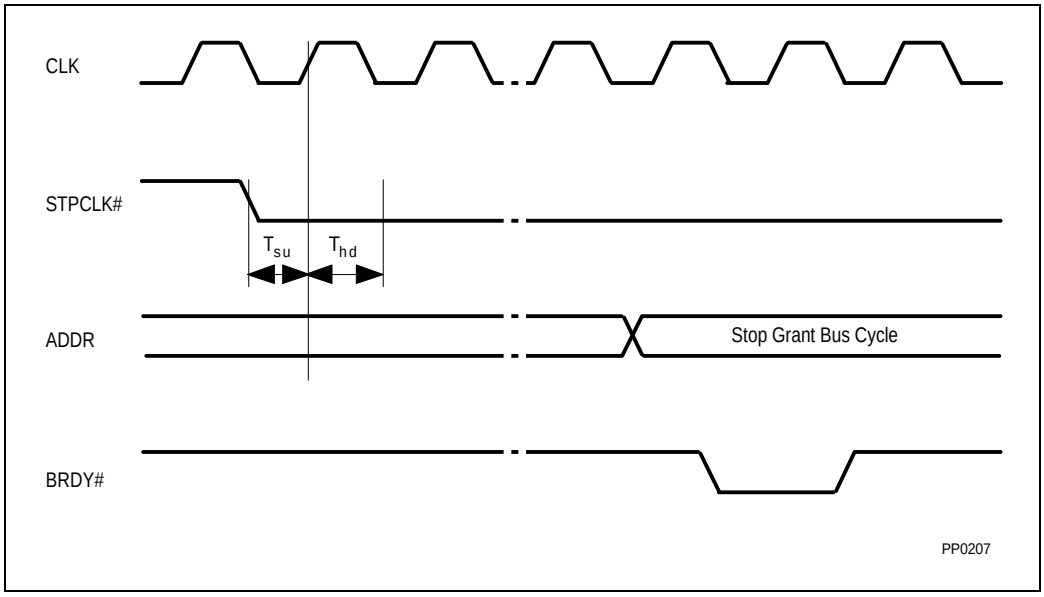


Figure 14-8. Entering Stop Grant State

14.4.4. Pin State during Stop Grant

During the Stop Grant state, most output and input/output signals of the microprocessor will be held at their previous states (the level they held when entering the Stop Grant state). See Table 14-3. However, the data bus and data parity pins will be floated. In response to HOLD being driven active during the Stop Grant state (when the CLK input is running), the CPU will generate HLDA and tri-state all output and input/output signals that are tri-stated during the HOLD/HLDA state. After HOLD is de-asserted, all signals will return to their states prior to the HOLD/HLDA sequence.

Table 14-3. Pin State During Stop Grant Bus State

Signal	Type	State
A31-A3	I/O	Previous State
D63-D0	I/O	Floated
BE7# - BE0#	O	Previous State
DP7 - DP0	I/O	Floated
W/R#, D/C#, M/IO#	O	Previous State
ADS#, ADSC#	O	Inactive
LOCK#	O	Inactive
BREQ	O	Previous State
HLDA	O	As per HOLD
FERR#	O	Previous State
PCHK#	O	Previous State
PWT, PCD	O	Previous State
SMIACT#	O	Previous State

In order to achieve the lowest possible power consumption during the Stop Grant state, the system designer must ensure the input signals with pull-up resistors are not driven LOW and the input signals with pull-down resistors are not driven HIGH. (Refer to Table 4-4 to Table 4-7 in this document for signals with internal pull-up and pull-down resistors).

All inputs, except data bus pins, must be driven to the power supply rails to ensure the lowest possible current consumption during Stop Grant or Stop Clock modes. Data pins should be driven low to achieve the lowest power consumption. Pull down resistors or bus keepers are needed to minimize the leakage current.

14.4.5. CLOCK CONTROL STATE DIAGRAM

Figure 14-9 shows the state descriptions and the state transitions for the clock control architecture.

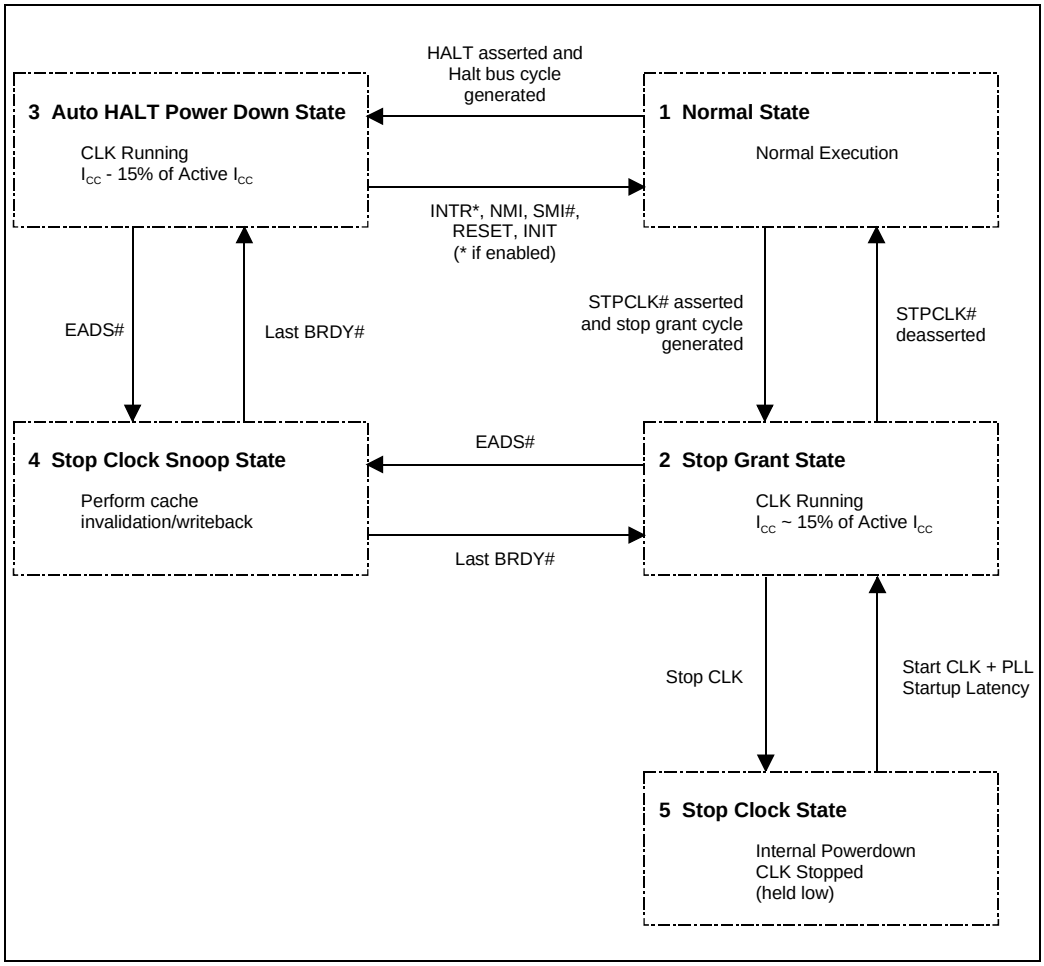


Figure 14-9. Stop Clock State Machine

A Flush State can be entered from states 1, 2 and 3 by asserting the FLUSH# input signal. The flush state is exited (e.g., the CPU returns to the state from which it came) when the Flush Acknowledge Special Bus Cycle is issued by the CPU.

14.4.5.1. NORMAL STATE — STATE 1

This is the normal operating state of the CPU.

14.4.5.2. STOP GRANT STATE — STATE 2

The Stop Grant state (~15% of normal state I_{CC}) provides a fast wake-up state that can be entered by simply asserting the external STPCLK# interrupt pin. Once the Stop Grant bus cycle has been placed on the bus, and BRDY# is returned, the CPU is in this state. The CPU returns to the normal execution state in approximately 10 clock periods after STPCLK# has been deasserted.

For minimum CPU power consumption, all other input pins should be driven to their inactive level while the CPU is in the Stop Grant state. A RESET will bring the CPU from the Stop Grant state to the normal state (note: unless STPCLK# is also deasserted, an active RESET will only bring the CPU out of the Stop Grant state for a few cycles). The CPU will recognize the inputs required for maintaining cache coherency (e.g., HOLD, AHOLD, BOFF#, and EADS# for cache invalidations and snoops) as explained later in this section. The CPU will not recognize any other inputs while in the Stop Grant state. Input signals to the CPU will not be recognized until 1 CLK after STPCLK# is de-asserted.

While in the Stop Grant state, the CPU will latch transitions on the external interrupt signals (SMI#, NMI, INTR, FLUSH#, R/S#, and INIT). All of these interrupts are taken after the deassertion of STPCLK# (e.g., upon re-entering the normal state). The Pentium processor requires INTR to be held active until the CPU issues an interrupt acknowledge cycle in order to guarantee recognition.

The CPU will generate a Stop Grant bus cycle only when entering that state from the normal state. When the CPU enters the Stop Grant state from the Stop Clock Snoop state, the CPU will not generate a Stop Grant bus cycle.

14.4.5.3. AUTO HALT POWERDOWN STATE — STATE 3

The execution of a HLT instruction will also cause the Pentium processor to automatically enter the Auto HALT Power Down state where I_{CC} will be ~15% of I_{CC} in the Normal state. The CPU will issue a normal HALT bus cycle when entering this state. The CPU will transition to the normal state upon the occurrence of INTR, NMI, SMI#, RESET, or INIT.

A FLUSH# event during the Auto HALT power down state will be latched and acted upon while in this state.

STPCLK# is not recognized by the CPU while in the Auto HALT Powerdown state. The system can generate a STPCLK# while the CPU is in the Auto HALT Powerdown state, but the Pentium processor will only service this interrupt if the STPCLK# pin is still asserted when the Pentium processor returns to the normal state.

While in Auto HALT Powerdown state, the CPU will only recognize the inputs required for maintaining cache coherency (e.g., HOLD, AHOLD, BOFF#, and EADS# for cache invalidations and snoops) as explained later in this section.

14.4.5.4. STOP CLOCK SNOOP STATE (CACHE INVALIDATIONS) — STATE 4

When the CPU is in the Stop Grant state or the Auto HALT Powerdown state, the CPU will recognize HOLD, AHOLD, BOFF# and EADS# for cache invalidation/writebacks. When the system asserts HOLD, AHOLD, or BOFF#, the CPU will float the bus accordingly. When the system then asserts EADS#, the CPU will transparently enter the Stop Clock Snoop state and perform the required cache snoop cycle. It will then re-freeze the clock to the CPU core and return to the previous state. The CPU does not generate the Stop Grant bus cycle or HALT special cycle when it returns to the previous state.

14.4.5.5. STOP CLOCK STATE — STATE 5

Stop Clock state (~ 1% of normal state I_{CC}) is entered from the Stop Grant state by stopping the CLK input. Note: for the Pentium processor, the CLK must be held at a logic low while stopped. None of the CPU input signals should change state while the CLK input is stopped. Any transition on an input signal (with the exception of INTR) before the CPU has returned to the Stop Grant state will result in unpredictable behavior. If INTR is driven active while the CLK input is stopped, and held active until the CPU issues an interrupt acknowledge bus cycle, it will be serviced in the normal manner once the clock has been restarted. The system design must ensure the CPU is in the correct state prior to asserting cache invalidation or interrupt signals to the CPU.

While the processor is in the Stop Clock state, all pins with static pullups or pulldowns must be driven to their appropriate values as specified in Table 4-4 to Table 4-7.

During the Stop Clock state the CPU input frequency may be changed to any frequency between the minimum and maximum frequency listed in the AC timing specifications found in Chapter 7. To exit out of the Stop Clock state, the CLK input must be restarted and remain at a constant frequency for a minimum of 1 ms. The PLL requires this amount of time to properly stabilize. After the PLL stabilizes, the CPU will return to Stop Grant state and the STPCLK# signal may be deasserted to take the CPU out of Stop Grant state and back to the Normal state.

In order to realize the maximum power reduction while in the Stop Clock state, PICCLK and TCK should also be stopped. These clock inputs have the same restarting restrictions as CLK. The local APIC cannot be used while in the Stop Clock state since it also uses the system clock, CLK.

WARNING

The Stop Clock state feature cannot be used in dual processing or functional redundancy checking modes because there is no way to re-synchronize the internal clocks of the two processors.

15

Pentium® Processor Debugging

CHAPTER 15
PENTIUM® PROCESSOR DEBUGGING

15.1. INTRODUCTION

Pentium processor-based system designers intending to use integration tools to debug their prototype systems can interface to the processor using two methods: (1) insert an emulator probe into the CPU socket, or (2) include some simple logic on their board that implements a debug port connection. Inserting an emulator probe into the CPU socket will allow access to all bus signals, but capacitive loading issues may affect high speed operation. In contrast, the debug port connection will allow debugger access to Pentium processor's registers and signals without affecting any high speed operation. This allows the system to operate at full speed with the debugger attached. Therefore, Intel recommends that all Pentium processor-based system designs include a debug port .

15.2. TWO LEVELS OF SUPPORT

Two levels of support are defined for the Pentium processor-based debug port, the second level being a superset of first. The system designer should choose the level of support that is appropriate for the particular system design and implement that level. Samples of each level of implementation are given in section 15.6 of this document.

15.2.1. Level 1 Debug Port (L1)

The Level 1 debug port supports systems with a single Pentium processor-based CPU. L1 uses a 20-pin connector to allow a debugger access to the processor's registers and signals.

15.2.2. Level 2 Debug Port (L2)

L2 extends the 20-pin debug port connector to 30 pins. The extra ten pins include a second set of boundary scan signals as well as additional R/S# and PRDY signals. The additional R/S# and PRDY signals are added to support the Pentium processor in the dual-processor configuration. This enables a debugger to provide separate control over the two CPUs during debug.

Signals on pins 1 through 20 of the L2 debug port are identical to the signals on the L1 debug port.

15.3. DEBUG PORT CONNECTOR DESCRIPTIONS

A debugger for Pentium processor-based designs can have a 30-pin connector on its probe that supports both levels of the debug port (as described previously, L1 or L2). Two cables can be provided, each cable having a 30-pin connector at one end (to mate with the debugger's probe connector) and the appropriate size connector at the other end to mate with the debug port in the system under debug. (For example, the L1 debug port Cable can be a 20-conductor cable with a 20-pin connector at one end and a 30-pin connector at the other end, leaving pins 21 to 30 unconnected.)

Intel recommended connectors to mate with debug port Cables are available in either a vertical or right-angle configuration. Use the one that fits best in your design. The connectors are manufactured by AMP Incorporated and are in their AMPMODU System 50 line. Table 15-1 shows the AMP part numbers for the various connectors:

Table 15-1. Recommended Connectors

	Vertical	Right-Angle
20-pin shrouded header	104068-1	104069-1
30-pin shrouded header	104068-3	104069-5

NOTE:

These are high density through hole connectors with pins on 0.050" by 0.100" centers. Do not confuse these with the more common 0.100" by 0.100" center headers.

Figure 15-1 is an example of the pinout of the connector footprint as viewed from the connector side of the circuit board. This is just an example. Contact your tools representative to determine the correct implementation for the tool you will use. Note that the 30-pin connector is a logical extension of the 20-pin connector with the key aligned with pin 15.

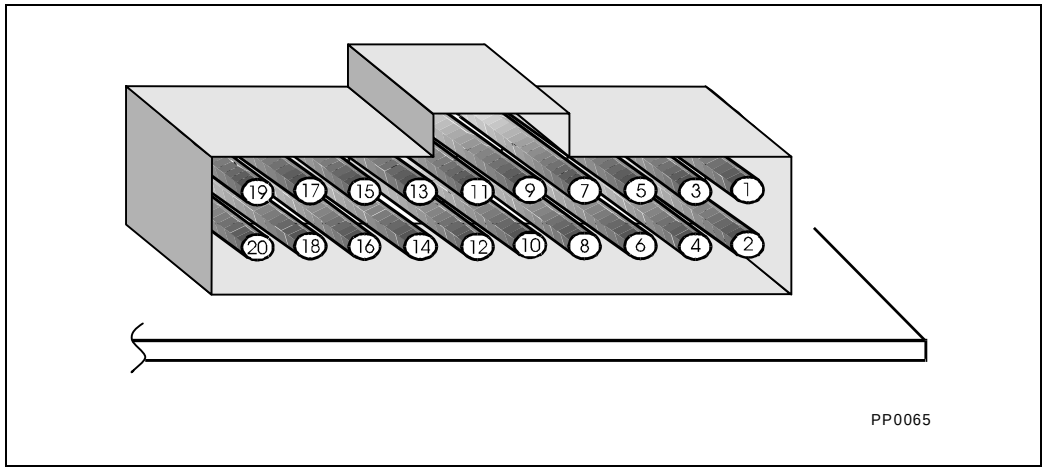


Figure 15-1. Debug Port Connector

15.4. SIGNAL DESCRIPTIONS

Table 15-2 shows the debug port signals. Direction is given as follows: O = output from the Pentium processor-based board to a debugger; I = input to the Pentium processor-based board from a debugger. These are 3.3V signals, compatible with the Pentium processor DC specifications. For the L1 debug port, ignore signals on pins 21 through 30.

NOTE

Target systems should be sure to provide a way for debugging tools like emulators, in-target probes and logic analyzers to reset the entire system, including upgrade processor, chip sets, etc. For example, if you follow the debug port implementation described below, the DBRESET signal provides this functionality. If you are not implementing the debug port, make sure that your system has a test point connected into the system reset logic to which a debug tool can connect.

Table 15-2. Debug Port Signals

Signal Name	Dir	Pin	Description
INIT	O	1	(Pentium® processor signal). A debugger may use INIT to support emulating through the CPU INIT sequence while maintaining breakpoints or breaking on INIT.
DBRESET	I	2	Debugger Reset output. A debugger may assert DBRESET (high) while performing the “RESET ALL” and “RESET TARGET” debugger commands. DBRESET should be connected to the system reset circuitry such that the system and processor(s) are reset when DBRESET is asserted. This is useful for recovering from conditions like a “ready hang.” This signal is asynchronous.
RESET	O	3	(Pentium processor signal). A debugger may use RESET to support emulating through the reset while maintaining breaking on RESET.
GND		4	Signal ground.
NC		5	No connect. Leave this pin unconnected.
V_{CC}		6	V_{CC} from the Pentium processor system. A debugger uses this signal to sense that system power is on. Connect this signal to V_{CC} through a 1 K Ohm (or smaller) resistor.
R/S#	I	7	Connect to the R/S# pin of the Pentium processor.
GND		8	Signal ground.
NC		9	No connect. Leave this pin unconnected.
GND		10	Signal ground.
PRDY	O	11	From the PRDY pin of the Pentium processor.
TDI	I	12	Boundary scan data input (Pentium processor signal). This signal connects to TDI of the Pentium processor. For dual processor operation, TDI of the Dual Pentium processor would connect to TDO of the Pentium processor.
TDO	O	13	Boundary scan data output (Pentium processor signal). This signal connects to TDO from the Pentium processor for a single processor design, or to TDO from the Dual Pentium Pentium processor for dual processor operation.
TMS	I	14	Boundary scan mode select (Pentium processor signal).
GND		15	Signal ground.

Table 15-2. Debug Port Signals (Contd.)

Signal Name	Dir	Pin	Description
TCK	I	16	Boundary scan clock (Pentium processor signal).
GND		17	Signal ground.
TRST#	I	18	Boundary scan reset (Pentium processor signal).
DBINST#	I	19	DBINST# is asserted (connected to GND) while the debugger is connected to the debug port . DBINST# can be used to control the isolation of signals while the debugger is installed.
BSEN#	I	20	Boundary scan enable. This signal can be used by the Pentium processor system to control multiplexing of the boundary scan input pins (TDI, TMS, TCK, and TRST# signals) between the debugger and other boundary scan circuitry in the Pentium processor system. The debugger asserts (low) BSEN# when it is driving the boundary scan input pins. Otherwise, the debugger drivers are high impedance. If the boundary scan pins are actively driven by the Pentium processor system, then BSEN# should control the system drivers/multiplexers on the boundary scan input pins. See example 2 in section 15.6.
PRDY2	O	21	From the PRDY pin of the Dual Pentium processor (for dual processor operation).
GND		22	Signal ground.
R/S#2	I	23	Connect to the R/S# pin of the Dual Pentium processor (for dual processor operation).
NC		24	
NC		25	
NC		26	
NC		27	
NC		28	
GND		29	Signal ground.
NC		30	

15.5. SIGNAL QUALITY NOTES

Since debuggers can connect to the Pentium processor-based system via cables of significant length (e.g., 18 inches), care must be taken in Pentium processor-based system design with regard to the signals going to the debug port. If system outputs to the debug port (i.e., TDO, PRDY, INIT and RESET) are used elsewhere in the system they should have dedicated drivers to the debug port. This will isolate them from the reflections from the end of the debugger cable. Series termination is recommended at the driver output. If the Pentium processor boundary scan signals are used elsewhere in the system, then the TDI, TMS, TCK, and TRST# signals from the debug port should be isolated from the system signals with multiplexers.

15.6. IMPLEMENTATION EXAMPLES

15.6.1. Example 1: Single CPU, Boundary Scan Not Used by System

Figure 15-2 shows a schematic of a minimal Level 1 debug port implementation for a Pentium processor, single-processor system in which the boundary scan pins of the Pentium processor are not used in the system.

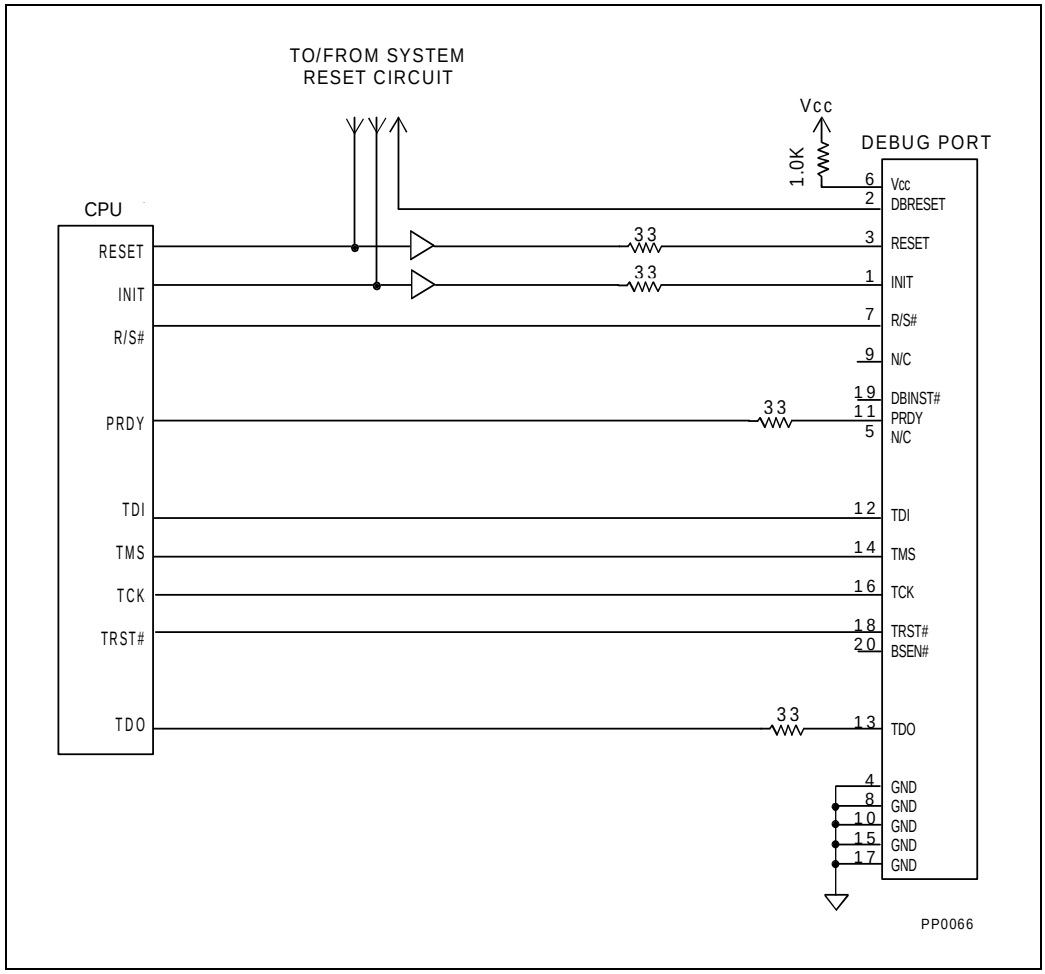


Figure 15-2. Single CPU — Boundary Scan Not Used

15.6.2. Example 2: Single CPU, Boundary Scan Used by System

Figure 15-3 shows a schematic of a Level 1 debug port implementation for a Pentium processor, single-processor system in which the boundary scan pins of the Pentium processor are used in the system. Note that the BSEN# signal controls the multiplexing of the boundary scan signals. With this implementation, the Pentium processor-based system could use the boundary scan (through the Pentium processor) while the debugger is "emulating," but could not while the debugger is "halted" (because the chain is broken).

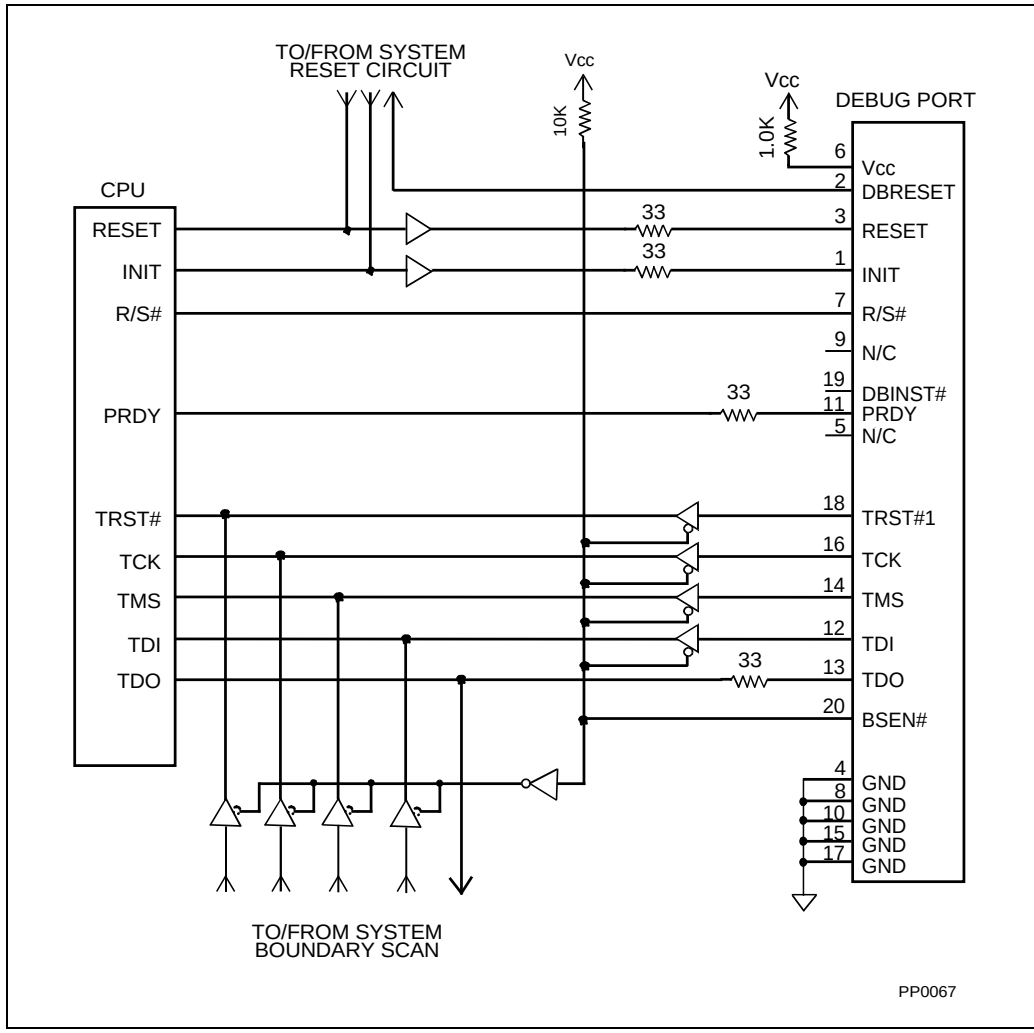


Figure 15-3. Single CPU — Boundary Scan Used

15.6.3. Example 3: Dual CPUs, Boundary Scan Not Used by System

Figure 15-4 shows a schematic of a typical Level 2 debug port implementation for a Pentium processor, dual-processor system in which the boundary scan pins of the Pentium processor are not used in the system. The multiplexer circuit for use with the "upgrade socket" concept is shown, but could be replaced with a jumper.

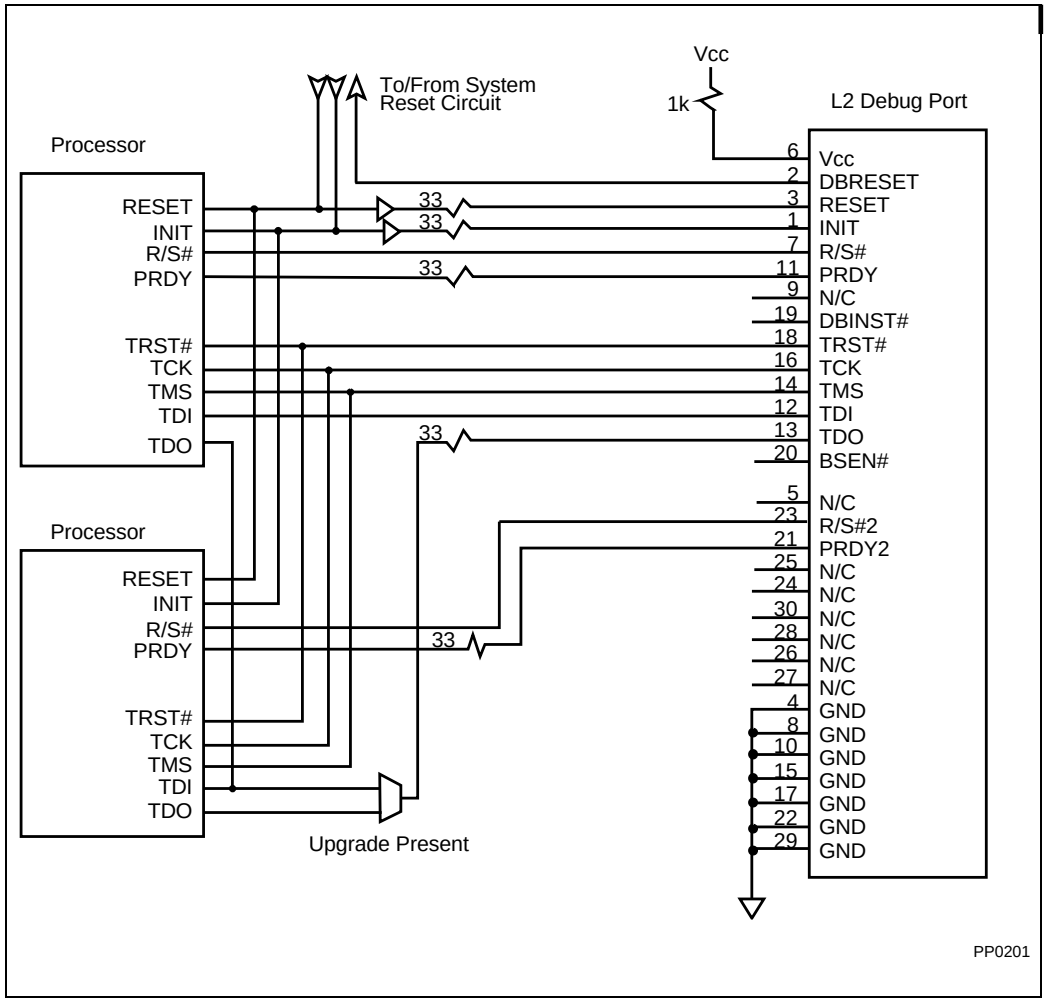


Figure 15-4. Dual CPUs — Boundary Scan Not Used

15.6.4. Example 4: Dual CPUs, Boundary Scan Used by System

Figure 15-5 shows a schematic of a Level 2 debug port implementation for a Pentium processor, dual-processor system that uses boundary scan. Note that the BSEN# signal controls the multiplexing of the boundary scan signals. With this implementation, the Pentium processor-based system could use the boundary scan (through the Pentium processor) while the debugger is "emulating," but could not while the debugger is "halted" (because the chain is broken).

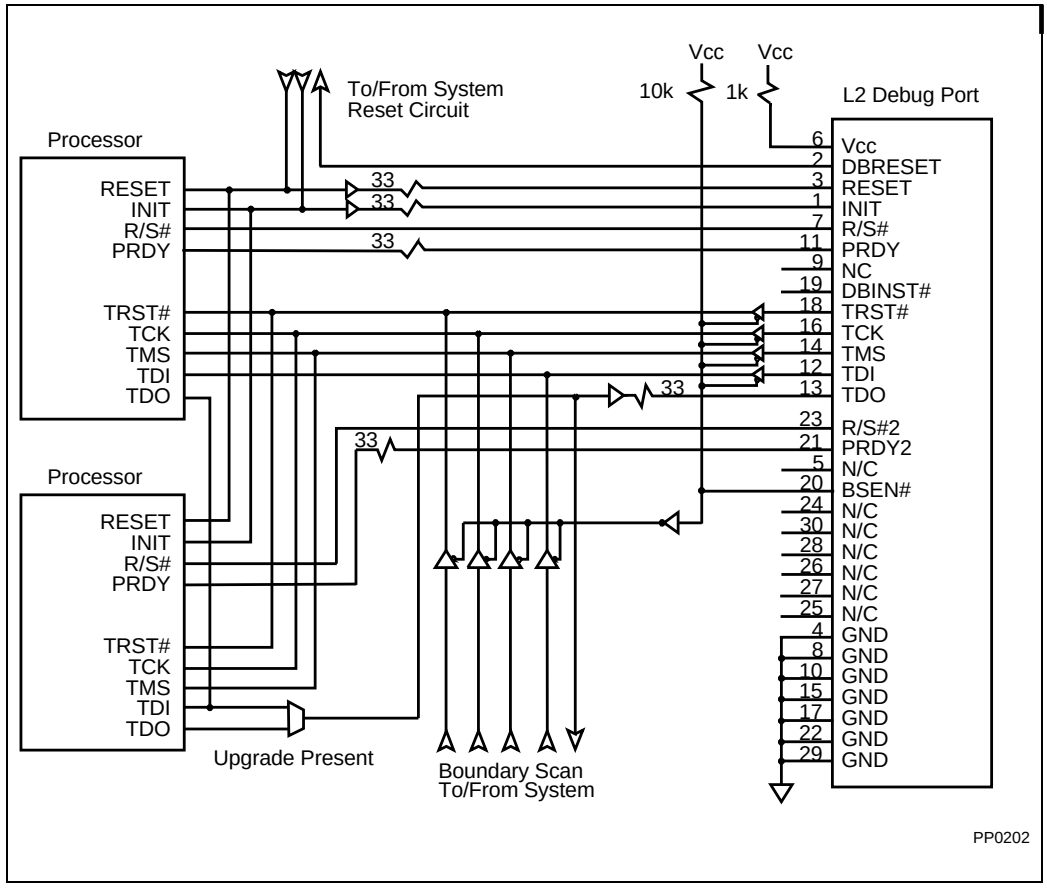


Figure 15-5. Dual CPUs — Boundary Scan Used

15.7. IMPLEMENTATION DETAILS

15.7.1. Signal Routing Note

The debugger software communicates with the CPU through the debug port using the boundary scan signals listed above. Typically, the debugger expects the CPU to be the first and only component in the scan chain (from the perspective of the debug port). That is, it expects TDI to go directly from the debug port to the TDI pin of the CPU, and the TDO pin to go directly from the CPU to the debug port (see Figure 15-6). If you have designed your system so that this is not the case (for instance, see Figure 15-7), you will need to provide the debugger software with the following information: (1) position of the CPU in the scan chain, (2) the length of the scan chain, (3) instruction register length of each device in the scan chain. Without this information the debugger will not be able to establish communication with the CPU.

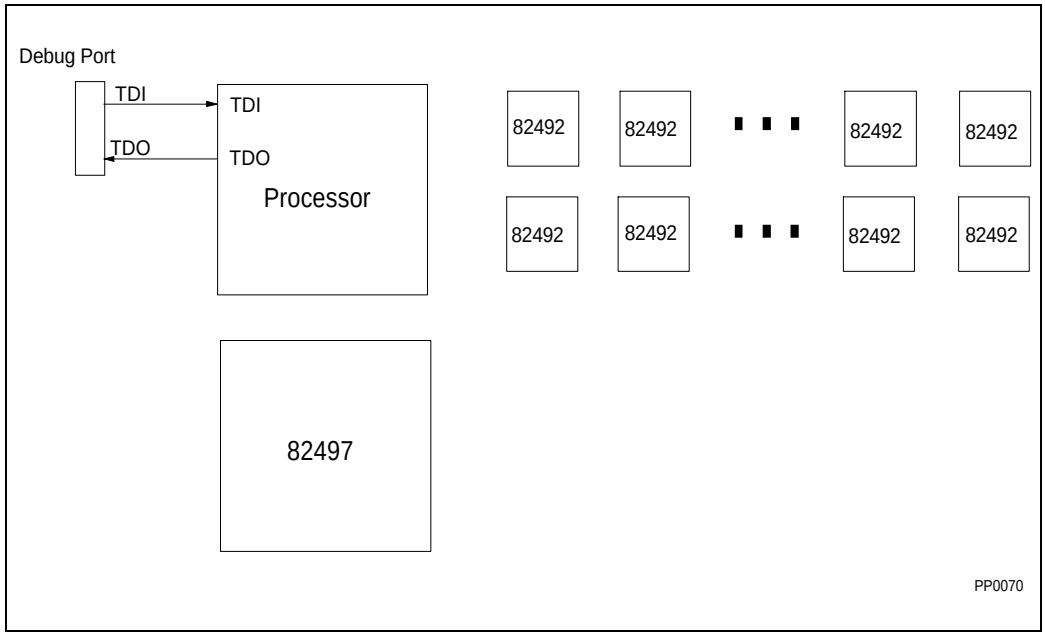


Figure 15-6. Example of CPU Only in Scan Chain

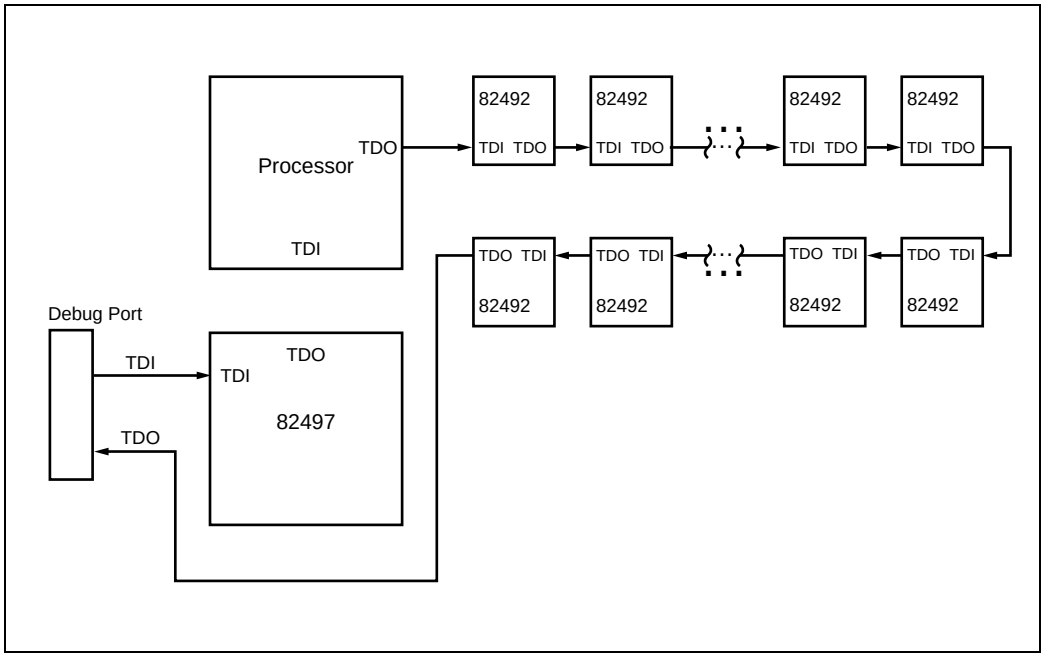


Figure 15-7. Example of Multiple Components in Scan Chain

15.7.2. Special Adapter Descriptions

For those designs where board real estate is a concern or where the design is finished and it is too late to implement the debug port, it may be possible to use a special “debug port adapter” to replace the on-board debug port described in the previous sections. The purpose of the adapter is to provide easy access to the boundary scan signals of the CPU(s). For simplicity, the adapter should make the boundary scan signals accessible to the debug tool while at the same time preventing the target system from accessing them. Two debug port adapters are described: (1) for uni-processor debug, (2) for dual-processor debug.

15.7.2.1. UNI-PROCESSOR DEBUG

A debug port adapter for use in uni-processor systems, or dual-processor systems where only one processor will be debugged at a time, can be built by reworking two Pentium processor SPGA sockets (see Figure 15-8).

NOTE

This adapter can be used only when the CPU is NOT included in the target system boundary scan string. In addition, when used in dual-processor systems you will only be able to debug the CPU to which the adapter is connected.

Table 15-3 show which pins to connect lines of appropriate 20- or 30- wire cable to on the top socket.

Table 15-3. SPGA Socket

Cable Wire #	SPGA Pin#	Signal
1	AA33	INIT
2	NC	DRESET
3	AK20	RESET
4	AD36(V_{SS})	GND
5	NC	NC
6	U37	V_{CC}
7	AC35	R/S#
8	AB36(V_{SS})	GND
9	NC	NC
10	Z36(V_{SS})	GND
11	AC05	PRDY
12	N35	TDI
13	N33	TDO
14	P34	TMS
15	X36(V_{SS})	GND
16	M34	TCK
17	R36(V_{SS})	GND
18	Q33	TRST#
19	NC	DBINST#
20	NC	BSEN#

NOTE

You may connect the GND pins to any pin marked V_{SS} on the SPGA pinout diagram. The NC pins are no connects. You may simply cut those wires.

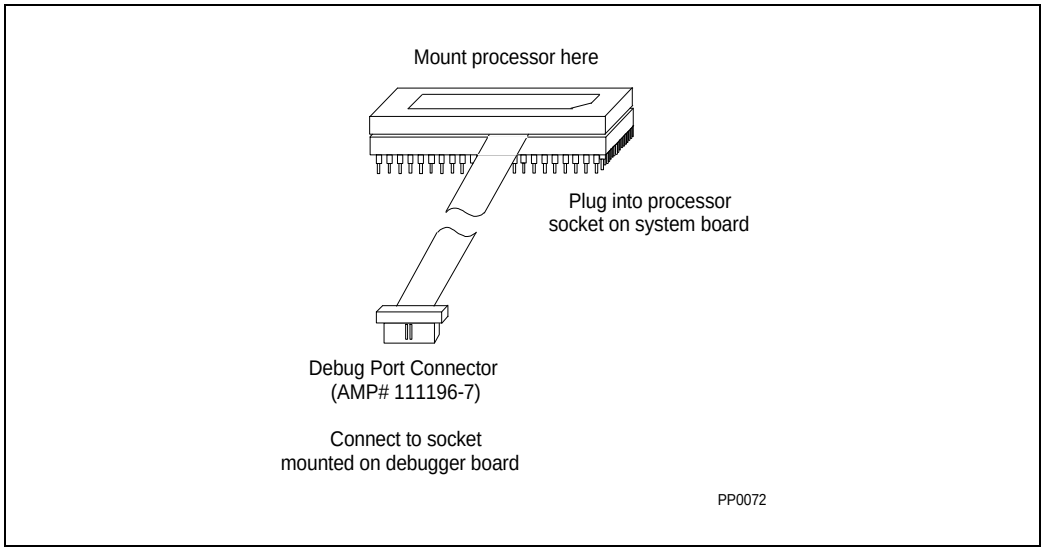


Figure 15-8. Uni-Processor Debug

Connect a double-row receptacle (AMP# 111196-7) to the debug port connector end of the cable. This is a 30-pin connector, so that it fits into the socket on the debugger buffer board.

Remove the following pins from the bottom socket:

R/S#	AC35
PRDY	AC05
TDI	N35
TDO	N33
TMS	P34
TCK	M34
TRST#	Q33

Connect the two sockets together. Make sure not to crush the wires between the pins.

15.7.2.2. DUAL-PROCESSOR DEBUG

A debug port adapter for use in dual processor debugging can be built by reworking four Pentium processor-based SPGA sockets. (See Figure 15-9).

NOTE

This adapter can be used only when the CPUs are NOT included in the target system boundary scan string.

You will need to use two SPGA sockets per processor location. For this discussion, assume that the startup processor is called processor 1 and that the upgrade processor is called processor 2. Thus, you will use two SPGA sockets to connect to processor 1 and two SPGA sockets to connect to processor 2. Certain debug port signals must be shared by Processor 1 and Processor 2. These signals must be connected from the debug port connector end of the cable (on which you will place a double-row receptacle: AMP# 111196-7) to both double SPGA sockets.

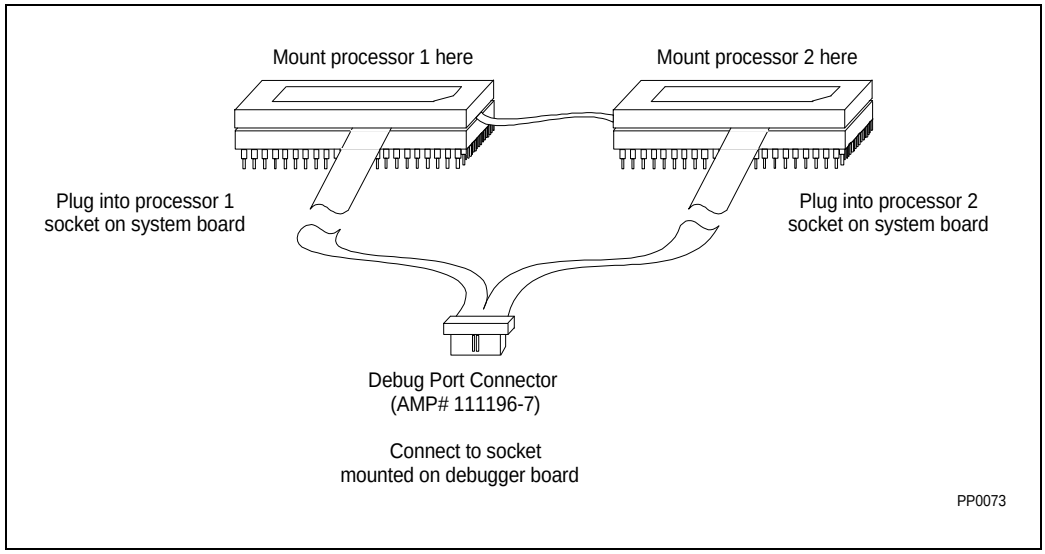


Figure 15-9. Dual-Processor Debug Port Adapter

Connect lines of 30-wire cable to the pins on the top SPGA sockets for both processor 1 and 2. Following are the signals which should be connected to each processor socket. Make sure to connect the shared lines to both top sockets.

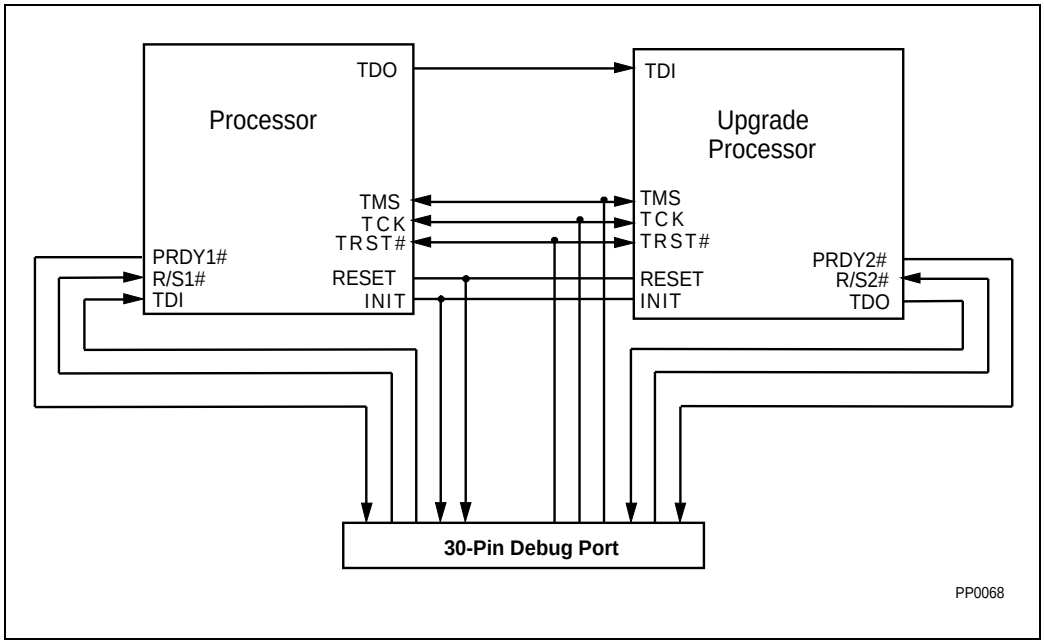


Figure 15-10. Shared Pins for Dual-Processor Adapter

Table 15-4. Debug Port Connector Pinout

Cable wire #	SPGA Pin#	Processor Socket	Signal
1	AA33	1,2	INIT
2	NC		DBRESET
3	AK20	1,2	RESET
4	V_{SS}	1	GND
5	NC		NC
6	V_{CC}	1	V_{CC}
7	AC35	1	R/S1#
8	V_{SS}	1	GND
9	NC		NC
10	V_{SS}	1	GND
11	AC05	1	PRDY1
12	N35	1	TDI
13	N33	2	TDO
14	P34	1,2	TMS
15	V_{SS}	1	GND
16	M34	1,2	TCK
17	V_{SS}	1	GND
18	Q33	1,2	TRST#
19	NC		DBINST#
20	NC		BSEN#
21	AC05	2	PRDY2
22	V_{SS}	2	GND
23	AC35	2	R/S2#
24	NC		NC
25	NC		NC
26	NC		NC
27	NC		NC
28	NC		NC
29	V_{SS}	2	GND
30	NC		NC

NOTE:

You may connect the V_{CC} and GND pins to any convenient power or ground pin.

Connect a double-row receptacle (AMP# 111196-7) to the debug port end of the cable. This is a 30-pin connector, so that it fits into the socket on the debugger buffer board.

Remove the following pins from the bottom of both double sockets:

R/S#	AC35
PRDY	AC05
TDI	N35
TDO	N33
TMS	P34
TCK	M34
TRST#	Q33

Connect each set of two sockets together. Make sure not to crush the wires between the pins.

16

Model Specific Registers and Functions

CHAPTER 16
MODEL SPECIFIC REGISTERS AND FUNCTIONS

This chapter introduces the model specific registers (MSRs) as they are implemented on the Pentium processor family. Model specific registers are used to provide access to features that are generally tied to implementation dependent aspects of a particular processor. For example, testability features that provide test access to physical structures such as caches, and branch target buffers are inherently model specific. Features to measure the performance of the processor or particular components within the processor are also model specific.

The features provided by the model specific registers are expected to change from processor generation to processor generation and may even change from model to model within the same generation. Because these features are implementation dependent, they are not recommended for use in portable software. Specifically, software developers should not expect that the features implemented within the MSRs will be supported in an upward or downward compatible manner across generations or even across different models within the same generation.

The Pentium processor with MMX technology MSRs are different than the Pentium processor (75/90/100/120/133/150/166/200) MSRs. When possible, fields were preserved between the two processors. Differences between the MSRs are noted throughout this chapter.

16.1. MODEL SPECIFIC REGISTERS

The Pentium processor family implements the RDMSR and WRMSR instructions to read and write the MSR's respectively. A feature bit in EDX (bit 5), reported by the CPUID instruction, indicates whether the processor supports the RDMSR and WRMSR instructions. The Pentium processor with MMX technology implements a new instruction called RDPMC (Read Performance Monitoring Counter). This instruction enables the user to read the performance monitoring counters in "Current Privilege Level = 3" given bit 8 is set in CR4 (CR4.PCE).

16.1.1. Model Specific Register Usage Restrictions

Proper use of the MSR features described in this chapter requires that the CPUID instruction be used not only to validate that the FAMILY reported in the EAX register is equal to "5", but also to validate the specific MODEL number within that FAMILY. Note that this requirement is significantly more restrictive than is required for new architectural features where it is sufficient to validate that the FAMILY is equal to or greater than that of the first family to implement the new feature. For more information regarding the use of the CPUID instruction, refer to the *Intel Architecture Software Developer's Manual*.

16.1.2. Model Specific Registers

Access to the model specific registers is provided through the RDMSR and WRMSR instructions. Access to a particular MSR is achieved by loading the ECX register with the appropriate ECX value from Table 16-1 below, and then executing either RDMSR or WRMSR. For more information regarding the use of these instructions, refer to the *Intel Architecture Software Developer's Manual*.

Table 16-1. Model Specific Registers

ECX Value (in Hex)	Register Name	Description
00	Machine Check Address [1]	Stores address of cycle causing the exception
01	Machine Check Type [1]	Stores cycle type of cycle causing the exception
02	Test Register 1	Parity Reversal Register
03	RESERVED	
04	Test Register 2 [2]	Instruction Cache End Bit
05	Test Register 3	Cache Test Data
06	Test Register 4	Cache Test Tag
07	Test Register 5	Cache Test Control
08	Test Register 6	TLB Test Linear Address
09	Test Register 7	TLB Test Control & Physical Address 31–12
0A	RESERVED	
0B	Test Register 9	BTB Test Tag
0C	Test Register 10	BTB Test Target
0D	Test Register 11	BTB Test Control
0E	Test Register 12	New Feature Control
0F	RESERVED	
10	Time Stamp Counter	Performance Monitor
11	Control and Event Select	Performance Monitor
12	Counter 0	Performance Monitor
13	Counter 1	Performance Monitor
14+	RESERVED	

NOTES:

1. CR4.MCE must be 1 in order to utilize the machine check exception feature.
2. Reserved on the Pentium® processor with MMX™ technology.

16.2. TESTABILITY AND TEST REGISTERS

The Pentium processor provides testability access to the on-chip caches, TLBs, BTB and internal parity checking features through model specific test registers. The RDMSR/WRMSR instructions may be utilized by the Pentium processor to access the test registers.

16.2.1. Cache, TLB and BTB Test Registers

The Pentium processor and Pentium processor with MMX technology contain several test registers. The purpose of these test registers is to provide direct access to the Pentium processor's caches, Translation Look-aside Buffers (TLB), and Branch Target Buffer (BTB) so test programs can easily exercise these structures. Because the architecture of the caches, TLBs, and BTB is different, a different set of test registers (along with a different test mechanism) is required for each. Most test registers are shared between the code and data caches.

The test registers should be written to for testability purposes only. Writing to the test registers during normal operation causes unpredictable behavior. Note that when the test registers are used to read or write lines directly to or from the cache, external inquire cycles must be inhibited to guarantee predictable results when testing. This is done by setting both CR0.CD and CR0.NW to "1". In addition, the INVD, WBINVD and INVLPG instructions may be executed before and after but not during testing.

Since the on-board caches, TLBs, and BTB implemented in Pentium processor with MMX technology differ than those in Pentium processor (75/90/100/120/133/150/166/200), the test register interface differs.

NOTE

If a memory data access occurs during a code cache testability operation using the test registers, the data cache is checked before the external memory operation in initiated. If the access is a miss in the data cache, then if the accessed line is valid in the code cache, it is invalidated through the internal snooping mechanism. In addition, the same cache line fill buffer is used for cache testability writes and to temporarily store data from memory data reads. For this reason, memory data reads should be done with care or avoided to ensure data from the memory read does not overwrite data from the testability write in the cache line fill buffer.

Similarly, if a code access occurs during a data cache testability operation using the test registers, the code cache is checked before the external memory operation is initiated. If the access is a miss in the code cache, then the accessed line if valid in the data cache is invalidated (or written back and then invalidated if in the M state) through the internal snooping mechanism.

16.2.1.1. CACHE TEST REGISTERS

The registers in Figure 16-1 provide direct access to the Pentium processor's code and data caches.

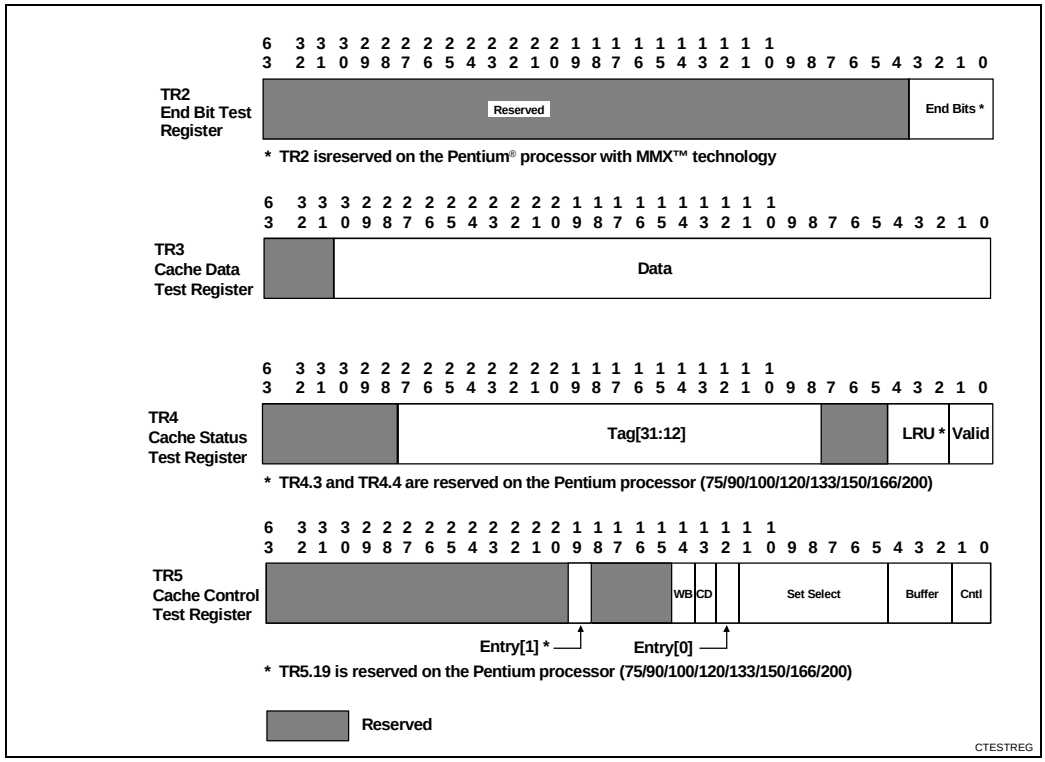


Figure 16-1. Cache Test Registers

On the Pentium processor (75/90/100/120/133/150/166/200), TR2 is the End Bit Test Register for the code cache. It contains four end bits. Each end bit corresponds to one byte of instruction in TR3 during code cache testability access. Since a cache line 32 bytes, 8 access are needed to read or write the end bits for the entire cache line. TR2 is used for accesses to the code cache only. TR2 is reserved on the Pentium processor with MMX technology.

End Bits are used to indicate instruction boundaries on the Pentium processor (75/90/100/120/133/150/166/200). The end bit mechanism aids the decode of two variable length instructions per clock by providing information on where the boundary between instruction is. If a given byte is the last byte in an instruction, the corresponding end bit is set to one. When a line is written into the code cache after a miss, all end bits corresponding to the line are initialized to one. As instructions are decoded, the end bits are checked for correctness and modified if incorrect. In order for two instructions to be issued in a single clock, the end bits of the u-pipe instruction must have the correct values, otherwise only one instruction will

be issued. This does have the effect that instructions are usually not paired the first time that they are put in the code cache.

TR3 is the Cache Data Test Register. This is where the data is held on its way into or out of the cache. Prior to a cache testability write, software must load an entire cache line into the 32-byte fill buffer using TR3, 4 bytes at a time. Similarly, during a cache testability read, the Pentium processor extracts a specified 4-byte data quantity from a cache line and places the data in TR3. A 32-byte cache line may be written to or read from TR3 as eight 4-byte accesses.

TR4 is the Cache Status Test Register. It contains the tag, LRU and valid bits to be written to or read from the cache. Like TR3, TR4 must be loaded with the tag/LRU/valid bits prior to a testability write, and gets updated with the tag/LRU/valid bits as a result of a testability read. **Note that TR4[31:28] are reserved and always return a zero as a result of a testability read.** The two valid bits are interpreted differently by the code and data caches, depending upon the setting of TR5.CD bit. The encodings for TR4.valid are shown in Table 16-2. The encodings for the LRU bits are shown in Table 16-3 for the Pentium processor (75/90/100/120/133/150/166/200) and the Pentium processor with MMX technology.

Table 16-2. Encoding for Valid Bits in TR4

TR5.CD=1 (Data Cache)	valid[1]	valid[0]	Meaning
	0	0	Cache line in I state
	0	1	Cache line in S state
	1	0	Cache line in E state
	1	1	Cache line in M state
TR5.CD=0 (Code Cache)	**valid[1]**	**valid[0]**	**Meaning**
	X	0	Cache line invalid
	X	1	Cache line valid

Table 16-3. Encoding of the LRU Bit in TR4

Pentium® Processor (75/90/100/120/133/150/166/200)			
LRU[0]		Points to WAY	
0		0	
1		1	
Pentium Processor with MMX™ Technology			
LRU[2]	**LRU[1]**	**LRU[0]**	**Points to WAY**
X	0	0	0
X	1	0	1
0	X	1	2
1	X	1	3

NOTE

The LRU bits for the instruction cache change state when an entry is read using the test registers, with CR0.CD=1. The LRU bits for the data cache, however, do not change their state during testability reads with CR0.CD=1.

TR5 is the Cache Control Test Register. It contains the writeback bit, the CD bit, the cache entry, the set address, the buffer select, and a two-bit control field, cntl.

The writeback bit determines whether that particular line is configured for writethrough or allows the possibility of writeback. It is used by the data cache only (i.e., if the writeback bit is set and a flush occurs (TR5.cntl=11), then if the addressed line in the data cache is modified, it will be invalidated and written back to the bus). The CD bit distinguishes between the code and data cache. The entry field selects one of the four ways in the Pentium processor with MMX technology (two ways in the Pentium processor (75/90/100/120/133/150/166/200)) in the cache. The set address field selects one of 128 sets within the cache to be accessed. The buffer field selects one of the eight portions of a cache line to be visible through TR3. The control field selects one of the four possible operation modes. The encodings for the TR5 fields are shown in Table 16-4, Table 16-5, Table 16-6 and Table 16-7.

Table 16-4. Encoding of the WB Bit in TR5

WB	Writeback or Writethrough
0	Writethrough
1	Writeback

Table 16-5. Encoding of the Code/Data Cache Bit in TR5

CD	Cache
0	Code cache
1	Data cache

Table 16-6. Encoding of the Entry Bit in TR5

Entry[1]	Entry[0]	Way
0	0	0
0	1	1
1	0	2
1	1	3

NOTE: The Entry[1] bit, Way 2 and Way 3 are specific to the Pentium® processor with MMX™ technology .

Table 16-7. Encoding of the Control Bits in TR5

Cntl1	Cntl0	Command
0	0	Normal operation
0	1	Testability write
1	0	Testability read
1	1	Flush

16.2.1.1.1. Direct Cache Access

To access the cache for testing, the programmer specifies a set address and entry and requests a testability read or write. No tag comparison is done; the programmer can directly read/write a particular entry in a particular set. Note that since TR2 is reserved for Pentium processor with MMX technology, there is no TR2 access when reading an entry from the cache.

To write down an entry into the cache:

- Disable replacements by setting CR0.CD=1.
- For each 4-byte access:
    1) Write address into TR5.buffer. Here, TR5.cntl=00.
    2) Write data into TR3.
    3) Write end bits into TR2 (for instruction cache only).
- Write the desired tag, LRU and valid bits into TR4. Note that the contents of TR4 completely overwrites the previous tag, LRU and valid bits in the cache.
- Perform a testability write by loading TR5 with the appropriate CD, entry, set address, and cntl fields. Here, TR5.cntl=01.

To read an entry from the cache:

- For each 4-byte access:
    1) Write the appropriate CD, entry, set address, buffer and cntl fields into TR5. Here, TR5.cntl=10.
    2) Read data from TR3.
    3) Read end bits from TR2[3:0] (for instruction cache only).
    4) Read the tag, LRU, and valid bits from TR4. No hit/comparison is performed. Whatever was in that entry in that set is read into TR4, TR3, and TR2.

To invalidate the cache or invalidate an entry:

- When TR5.cntl=11 (flush), and CD=0 (code cache), the entire code cache is invalidated. However, if TR5.cntl=11 and CD=1 (data cache), the user can specify through the TR5.WB bit whether to invalidate the entire data cache, or invalidate and writeback only the cache line specified by TR5 (see Table 16-8).

Table 16-8. Definition of the WB Bit in TR5

TR5.cntl=11	TR5.WB	Meaning
CD=0	X	Invalidate the entire code cache.
CD=1	0	Invalidate entire data cache. Modified lines are not written back.
CD=1	1	Invalidate line. Writeback if modified.

Note that TR2, TR3, and TR4 permit both reads and writes, whereas TR5 is a write-only register. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation may cause unpredictable behavior. For example, inadvertent cache hits can be created.

During cache testability operations, the internal snooping mechanism functions similar to that described in section 6.5.3. If a memory data access occurs during a code cache testability operation using the test registers, the date cache is checked before the external memory operation is initiated. If the access is a miss in the data cache, then the accessed line if valid in the code cache is invalidated through the internal snooping mechanism. In addition, the same cache line fill buffer is used for cache testability writes and to temporarily store data from memory data reads. For this reason, memory data reads should be done with care or avoided to ensure data from the memory read does not overwrite data from the testability write in the cache line fill buffer.

Similarly, if a code access occurs during a data cache testability operation using the test registers, the code cache is checked before the external memory operation is initiated. If the access is a miss in the code cache, then the accessed line if valid in the data cache is invalidated (or written back and then invalidated if in the M-state) through the internal snooping mechanism.

When the FLUSH# pin is asserted, it is treated as an interrupt, and when serviced at the next instruction boundary, it causes a writeback of the data cache and then invalidation of the internal caches. The cache test registers TR2, TR3, TR4 and TR5 are used in this process, and thus their values after FLUSH# has been serviced are unpredictable. Therefore FLUSH# should not be asserted while code is being executed which uses these test registers.

16.2.1.2. TLB TEST REGISTERS

The registers in Figure 16-2 provide access to the Pentium processor's code and data cache translation lookaside buffers (TLBs). Note that the data cache has two TLBs: a 64-entry TLB for 4-Kbyte data pages and an 8-entry TLB for 4-Mbyte data pages. The code cache contains only one 32-entry TLB for both 4-Kbyte code pages and 4-Mbyte code pages. The 4-Mbyte code pages are cached in 4-Kbyte increments (the PS bit in TR6 is ignored). The code cache contains one fully associative 32-entry TLB which is also integrated for both 4-Kbyte and 4-Mbyte pages. Note that, unlike the Pentium processor (75/90/100/120/133/150/166/200), the Pentium processor with MMX technology data cache contains one fully associative 64-entry TLB which is integrated for both 4-Kbyte and 4-Mbyte pages.

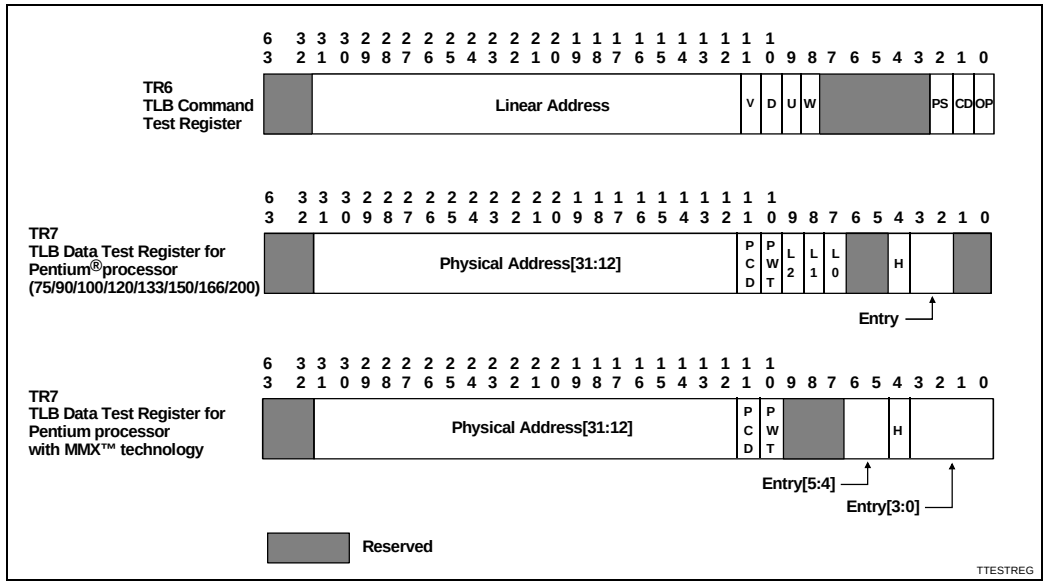


Figure 16-2. TLB Test Registers

TR6 is the TLB Command Test Register. It contains the linear address, code/data TLB select (CD), operation (Op) bits and the following status bits: valid (V), dirty (D), user (U), writeable (W), and page size (PS) bits.

The status bits are inputs to the TLB entry during testability writes, and outputs from the TLB entry during testability reads. The V bit indicates whether a TLB entry is valid or invalid during testability writes. The D bit indicates whether or not a write access was made to the page. The U bit indicates the privilege level that the Pentium processor must be in to access the page. The W bit is one of the factors in determining the read/write protection of the page. The PS (page size) bit specifies the page size for the TLB entry. The CD bit determines if the code or data TLB is being accessed. The Op bit distinguished between a read and write cycle.

The W-bit, D-bit, and PS-bit are defined only for the data TLB.

Table 16-9, Table 16-10, Table 16-11, Table 16-12, Table 16-13, Table 16-14 and Table 16-15 list the encodings for the fields in the TR6 register.

Table 16-9. Encoding for the Valid Bit in TR6

Valid	Valid/Invalid TLB Entry
0	Invalid
1	Valid

Table 16-10. Encoding for the Dirty Bit in TR6

D-bit	Write access made to page?
0	Write access was not made
1	Write access was made

Table 16-11. Encoding for the User Bit in TR6

U-bit	Privilege Level Access Allowed
0	PL=0,1,2,3
1	PL=0

Table 16-12. Encoding for the Writeable Bit in TR6

W-bit	Writes Allowed?
0	No writes, read only
1	Allows writes

Table 16-13. Encoding for the Page Size Bit in TR6

PS-bit	Page Size
0	4 KByte
1	4 MByte

NOTE

Normally the user should not allocate a page entry in both the TLBs; during testability however if a match is found in both, then the processor reports that it found it for the 4-Mbyte page size (PS=1).

Table 16-14. Encoding for the Operation Bit TR6

Op	Command
0	TLB write
1	TLB read

Table 16-15. Encoding for the Code/Data TLB Bit in TR6

CD	Cache
0	Code TLB
1	Data TLB

TR7 is the TLB Data Test Register. In the Pentium processor (75/90/100/ 120/133/150/166/200) it contains bits 31:12 of the physical address, the hit indicator H, a two-bit entry pointer, and the status bits. The status bits of the Pentium processor include the two paging attribute bits PCD and PWT, and three LRU bits (L0, L1, and L2). PCD is the page level cache disable bit. PWT is the page level write through bit. The LRU bits determine which entry is to be replaced according to the pseudo-LRU algorithm. TLB reads which result in hits and TLB writes can change the LRU bits. The LRU bits reported for a test read are the value before the TLB read. The LRU bits are then changed according to the pseudo-LRU replacement algorithm. The two entry bits determine which one of the four ways to write to in the code or data TLB during testability writes.

In the Pentium processor with MMX technology, the entry pointer has been extended from two bits to six bits. The six entry bits determine which one of the 64 entries to write to in the data TLB during testability writes. The lower five entry bits determine which one of the 32 entries to write to in the code TLB during testability writes. Also, the L0, L1 and L2 bits are reserved in the Pentium processor with MMX technology.

The H is the hit indicator. This bit needs to be set to 1 during testability writes. During testability reads, if the input linear address matches a valid entry in the TLB, the H bit is set to 1. The two entry bits determine in which one of the four ways to write to the TLB during testability writes. During testability reads, they indicate the way that resulted in a read bit.

TR6, and TR7 are read/write registers. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

When reading from the code cache TLB (TR5.CD = 0), the TR6 register zeros out bits [31:12] (corresponding to the linear address) at the end of the TLB testability read cycle. This does not mean that an incorrect linear address was used. All operations happen normally (with whatever linear address was written into TR6 before the testability read operation).

16.2.1.2.1. TLB Access

Unlike the caches, the TLB is structured as a CAM cell and, thus, can only be searched (rather than directly read). In other words, the programmer can directly read/write a particular entry in a particular set of the code or data caches, however the TLB only reports a hit or a miss in the Hit bit in TR7. Dumping the TLB requires the programmer to step through the entire linear address space one page at a time. Also, please note the following changes which apply to the Pentium processor with MMX technology:

- LRU bits of TR7 (bits 9:7) are reserved on the Pentium processor with MMX technology.

- The entry pointer in TR7 has been extended from two bits to six bits in the Pentium processor with MMX technology.
- To assure correct functioning, software MUST flush the TLB after testability writes and prior to return to normal operation mode by writing to CR3.
- It is recommended that users do not use testability reads to load the TLB with overlapping 4 Kbyte and 4 Mbyte pages.

To write an entry into the TLB:

- Write the physical address bits [31:12], attribute bits, LRU bits and replacement entry into TR7, setting TR7.H=1.
- Write the linear address, protection bits, and page size bit into TR6, setting TR6.Op=0.

To read an entry from the TLB:

- Write the linear address, CD, and OP bits into TR6, setting TR6.Op=1.
- If TR7.H is set to 1, the read resulted in a hit. Read the translated physical address, attribute bits, and entry from TR7. Read the V, D, U, and W bits from TR6. If TR7.H is cleared to 0, the read was a miss and the physical address is undefined.

Note that when reading from the TLB, the PS bit in the TR6 register does not have to be set; the PS bit is actually written by the processor at the end of the TLB (testability) lookup. Based on the PS bit the user is supposed to infer whether the linear address found in the TLB corresponds to the 4-Kbyte or 4-Mbbyte page size. Normally the user should not allocate a page entry in both the TLBs; during testability however if a match is found in both, then the processor reports that it found it for the 4-Mbyte page size (PS=1).

Also note that when reading from the code cache TLB (TR5.CD=0), the TR6 register zeros out bits 12-31 (corresponding to the linear address) at the end of the TLB testability read cycle. This does not mean that an incorrect linear address was used. All operations happen normally (with whatever linear address was written into TR6 before the testability read operation).

16.2.1.3. BTB TEST REGISTERS

The test registers in Figure 16-3 provide direct access to the branch target buffer. Note that the branch prediction mechanism should be disabled through test register 12 before doing any BTB testability access.

TR9 is the BTB Tag Test Register. Before writing any entry into the BTB, software must first load TR9 with the appropriate information. After reading any entry in the BTB, the processor places the retrieved information in TR9.

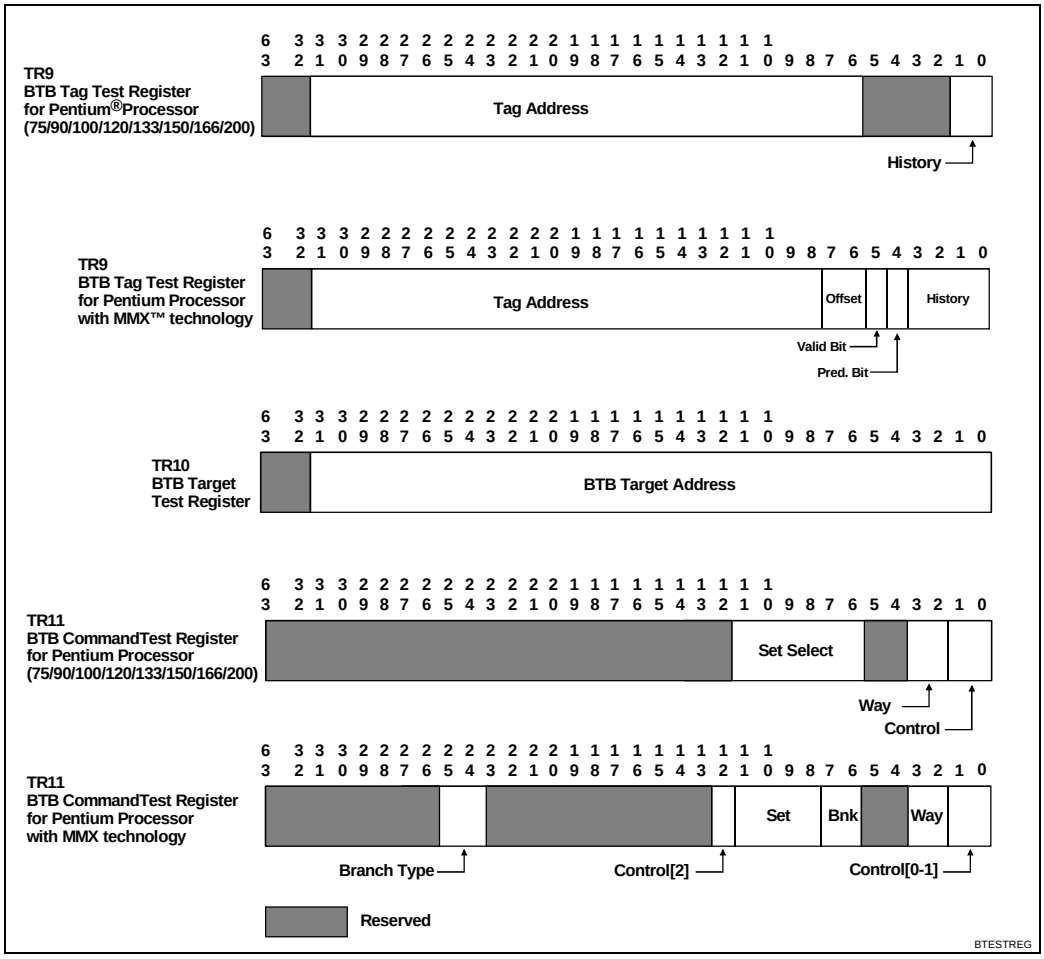


Figure 16-3. BTB Test Registers

Table 16-16. TR9 Register Description (BTB Test Register)

Bits in the Pentium® Processor (75/90/100/120/133/ 150/166/200)	Bits in the Pentium Processor with MMX™ Technology	TR9 Register Description (BTB Test Register)
63:32	63:32	Reserved
31:6	31:8	Tag Address: Bits 31:6 or 31:8 of the address of the last byte of the branch
N/A	7:6	Offset: Bits 1:0 of the address of the last byte of the branch
N/A	5	Valid bit: If set, the entry is allocated in the BTB
N/A	4	Prediction bit: Defines if this branch is predicted taken or not taken by the BTB
1:0	3:0	History: Contains the previous history for this branch

Table 16-17. TR10 Register Description (BTB Target Test Register)

Bits	TR10 Register Description (BTB Target Test Register)
63:32	Reserved
31:0	BTB Target Address: Linear address of the branch's target

Table 16-18. TR11 Register Description (BTB Command Test Register)

Bits in the Pentium® Processor (75/90/100/120/133/ 150/166/200)	Bits in the Pentium Processor with MMX™ Technology	TR11 Register Description (BTB Command Test Register)
63:32	63:32	Reserved
31:12	31:26	Reserved
N/A	25:24	Branch type: 00 JCC (Jump if condition is met), 01 unconditional jump, 10 call, 11 return
N/A	23:13	Reserved
N/A	12	Control: Selects either Normal operation, or Testability Read/Write, Flush and Testability Read Tag
11:6	11:8	Set : Selects one of 64 sets to access in the Pentium® processor (75/90/100/120/133/150/166/200) or 16 sets in the Pentium processor with MMX™ technology.
N/A	7:6	Bank: Selects one of the 4 banks per BTB cache line. The bank number corresponds to bits 3:2 of the branch address
5:4	5:4	Reserved
3:2	3:2	Way: Selects one of four ways within the Set (i.e. 00 = Way1, 01 = Way2, 10 = Way3 and 11 = Way4)
1:0	1:0	Control: Selects either Normal operation, or Testability Read/Write, Flush and Testability Read Tag

NOTE: The format for the control field is shown in Table 16-19.

TR10 is the BTB Target Test Register. Like TR9, TR10 must be loaded with the target address before a testability write. After a BTB testability read, the target address is placed in this register.

TR11 is the BTB Command Test Register. This register is used to issue read and write commands to the BTB. The set address field selects one of 16 sets (64 sets in the Pentium processor 75/90/100/120/133/150/166/200)) to access. The entry field selects one of four ways within the set on the Pentium processor (75/90/100/120/133/150/166/200). A BTB testability cycle is initiated by loading TR11 controls bits with the appropriate values. The format for the control field is shown in Table 16-19.

Table 16-19. Format for TR11 Control Field

Cntl2 [1]	Cntl1	Cntl0	Command
0	0	0	Normal operation
0	0	1	Testability write data
0	1	0	Testability read data
0	1	1	Testability BTB flush
1	0	1	Testability read TAG [1]

NOTES: Other combiniations are reserved.

1. Applies to the Pentium processor with MMX technology only.

TR9, TR10 and TR11 are all read/write registers. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

The following BTB testability cycles exist:

1. Testability read data. Reads the Target, branch type, offset, history, and prediction bit of a BTB line defined by a set, way and bank into the corresponding testability register field.
2. Testability read TAG and valid bit (Pentium processor with MMX technology only). Reads the Tag defined by the testability registers set, way and bank into the corresponding testability register field.
3. Testability BTB flush. Clear all BTB valid bits.
4. Testability Write Data. Writes all the BTB fields from the corresponding test registers. If there is an entry on the same bank and set, with the same TAG, the write overwrites this entry even if the way choosen in TR11 is different from the existing entry's way. (This is done to avoid having two entries in the same bank and same set, but different ways, with the same TAG.)

TR9, TR10, TR11 are all read/write registers. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

16.2.1.3.1. Direct BTB Access

The BTB contents are directly accessible, in a manner similar to the code/data caches. Note that the branch prediction mechanism should be disabled before doing any BTB testability access.

To write an entry into the BTB for the Pentium processor (75/90/100/120/133/150/166/200):

- Disable BTB entry allocation by setting TR12.NBP=1 (see Feature Control section)
- Write the tag address and history information in TR9
- Write the target address in TR10
- Write the appropriate set address, entry fields and control bits in TR11.

To write an entry into the BTB for the Pentium processor with MMX technology:

- Disable BTB entry allocation by setting TR12.NBP=1 (see Feature Control section)
- Write the tag address and history, offset, valid and prediction information in TR9
- Write the target address in TR10
- Write the appropriate set address, way, bank, branch type and control bits in TR11.

To read an entry from the BTB for Pentium processor (75/90/100/120/133/150/166/200):

- Perform a testability read by writing to TR11 with the appropriate set address entry fields.
- Read the tag address and history information from TR9.
- Read the target address from TR10.

To read an entry from the BTB for Pentium processor with MMX technology:

- Disable BTB entry allocation by setting TR12.NBP=1 (see Feature Control section)
- Perform a testability read by writing to TR11 with the appropriate set address, way, bank and control bits.
- Read the tag address, history information, offset, prediction and valid bits from TR9.
- Read the target address from TR10.
- Read the branch type from TR11.
- Perform a testability read tag by writing to TR11 with the appropriate set address, way, bank and control bits.
- Read the branch tag from TR9

NOTE

Read Tag and Read data does not destroy the other's cycle fields in TR9. This means that the read from TR9 can be done only once after both cycles were executed.

16.2.1.4. TEST PARITY CHECK (TR1)

A model specific register, TR1, the Parity Reversal Register (PRR), allows the parity check mechanism to be tested. Figure 16-4 shows the format of the PRR.

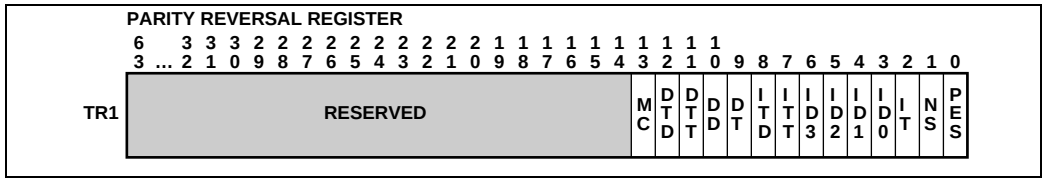


Figure 16-4. Parity Reversal Register

Table 16-20 lists each of the bits in the parity reversal register and their function.

Table 16-20. Parity Reversal Register Bit Definition

Bit Name	Description
PES	Parity Error Summary, set on any parity error
NS	0 = set PRR.PES, assert IERR#, and shutdown on parity error 1 = set PRR.PES, and assert IERR# on parity error
IT	code (instruction) cache tag
ID0	code cache data even bits 126, 124 ... 2,0
ID1	code cache data odd bits 127, 125 ... 3,1
ID2	code cache data even bits 254, 252 ... 130,128
ID3	code cache data odd bits 255, 253 ... 131, 129
ITT	code TLB tag
ITD	code TLB data
DT	data cache tag
DD	data cache data, use byte writes for individual access
DTT	data TLB tag
DTD	data TLB data
MC	microcode, reverse parity on read

Writing a one into bits 2-12 reverses the sense of the parity generation for any write into the corresponding array. This includes both normal cache replacements as well as testability writes and data writes. Parity is checked during both normal reads and testability reads.

To test parity error detection, software should write a one into the appropriate bit of the parity reversal register (PRR), perform a testability write into the array, and then perform a testability read. Upon successful detection of the parity error, the Pentium processor asserts the IERR# pin and may shutdown. Alternatively, after writing a one into the appropriate bit of the PRR, software may perform a normal write and read of the array by creating a cache miss and doing a read.

As an option, software may mask the shutdown by setting PRR.NS to 1 if the system is unable to recover from a shutdown. To determine if a parity error has occurred, software may read the parity error summary bit, PRR.PES. Hardware sets this bit on any parity error, and it remains set until cleared by software.

For the microcode, bad parity may be forced on a read by setting the PRR.MC bit to 1.

Bit 0 of TR1 is read/write. The remaining bits are write only. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

16.3. NEW FEATURE CONTROL (TR12)

The new features of branch prediction, execution tracing, and instruction pairing in the Pentium processor can be selectively enabled or disabled through individual bits in test register TR12 (Figure 16-5). The branch prediction, execution tracing, and instruction pairing features of the Pentium processor family can be selectively enabled or disabled through individual bits in test register TR12. In addition, level 1 caching can be disabled without affecting the PCD output to allow testing of a second level cache.

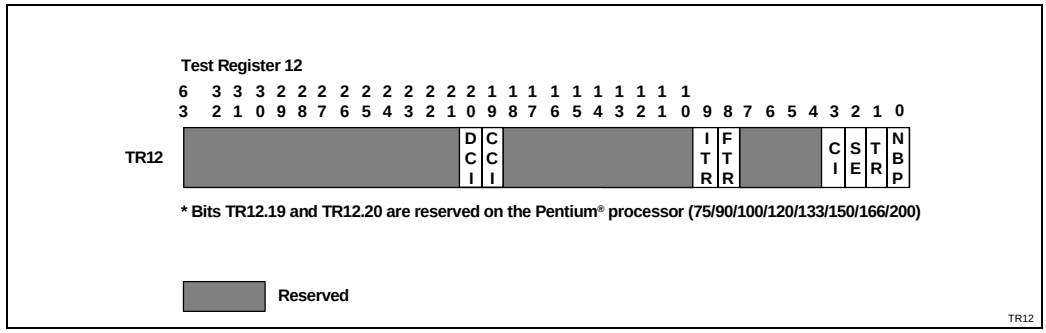


Figure 16-5. Test Register (TR12)

Table 16-21. New Feature Controls

Name	Position	Function
NBP	0	No Branch Prediction controls the allocation of new entries in the BTB. When TR12.NBP is clear, the code cache allocates entries in the BTB. When TR12.NBP is set, no new entry is allocated in the BTB, however, entries already in the BTB may continue to cause a BTB hit and result in the pipeline being reloaded from the predicted branch target. To completely disable branch prediction, first set TR12.NBP to 1 and then flush the entire BTB by loading CR3.
TR	1	Execution Tracing controls the Branch Trace message Special Cycle. When the TR12.TR bit is set to 1, a branch trace message special cycle is generated whenever a taken branch is executed. Two cycles are produced: one for the linear address of the instruction causing the taken branch, and one for the branch target linear address.
SE	2	Single Pipe Execution controls instruction pairing. When TR12.SE is cleared to zero, instructions are issued to both the u and v pipes contingent on pairing restrictions. When TR12.SE is set to one, the v pipe is disabled and instructions are issued only to the u pipe. Microcoded instructions are designed to utilize both pipes concurrently, independent of the state of TR12.SE. Note that all instructions requiring microcode are not pairable.
CI	3	Cache Inhibit controls line fill behavior. When TR12.CI is reset to 0, the on-chip data and instruction caches operate normally. When TR12.CI is set to 1, all cache line fills are inhibited and all bus cycles due to cache misses are run as single transfer cycles (CACHE# is not asserted). Unlike CR0.CD, TR12.CI does not affect the state of the PCD output pin. This allows the first level cache to be disabled while the second level cache is still active and can be tested. Note that the contents of the instruction and data caches are not affected by the state of TR12.CI, e.g., they are not flushed. The second level cache test sequence should be: set TR12.CI to 1, flush the internal caches, run the second level cache tests.
	4-7	Reserved
FTR	8	Fast Execution Tracing is similar to Execution Tracing (TR12.TR). If TR12.FTR is set to 1 while execution tracing is enabled (TR12.TR = 1), only one branch trace message special cycle is produced containing the linear address of the instruction causing the taken branch.
ITR	9	IO Trap Restart enables proper interrupt prioritization to support restarting IO accesses trapped by System Management Mode.
	10-18	Reserved
CCI	19 [1]	Code Cache Inhibit is the same instruction as Cache Inhibit (CI), but only applies to the code cache.
DCI	20 [1]	Data Cache Inhibit is the same instruction as Cache Inhibit (CI), but only applies to the data cache.
	21-63	Reserved

NOTES:

1. These bits are reserved on the Pentium processor (75/90/100/120/133/150/166/200).

TR12.NBP, TR12.TR, TR12.SE, and TR12.CI are initialized to zero on reset. This register is write only and the reserved bits should be written with zeros.

16.4. PERFORMANCE MONITORING

The Pentium processor includes features to measure and monitor various parameters that contribute to the performance of the processor. This information can be then used for compiler and memory system tuning. For memory system tuning, it is possible to measure data and instruction cache hit rates, and time spent waiting for the external bus. The performance monitor allows compiler writers to gauge the effectiveness of instruction scheduling algorithms by measuring address generation interlocks and parallelism.

While the performance monitoring features that are provided by the Pentium processor are generally model specific and available only to privileged software, the Pentium processor also provides an *architectural* Time Stamp Counter that is available to the user. With this notable exception, the performance monitor features and the events they monitor are otherwise implementation dependent, and consequently, they are not considered part of the Pentium processor *architecture*. The performance monitor features are expected to change in future implementations. *It is essential that software abide by the usage restrictions for accessing model specific registers as discussed in section 16.1.1.*

16.4.1. Performance Monitoring Feature Overview

Pentium processor performance monitoring features include:

Table 16-22. Architectural Performance Monitoring Features

RDTSC	Read Time Stamp Counter - a user level instruction to provide read access to a 64-bit free-running counter
RDPMC	Read Performance Monitoring Counter - this instruction enables reading of the performance monitoring counters (in CPL = 3) provided bit 8 of CR4 (CR4.PCE) is set. Note: The RDPMC instruction is only defined on the Pentium® processor with MMX™ technology. Execution of the RDPMC instruction in a Pentium processor (75/90/100/120/133/150/166/200) will result in an invalid opcode exception.
CPUID (EDX.TSC)	Time Stamp Counter Feature Bit - Bit 4 of EDX is set to 1 to indicate that the processor implements the TSC and RDTSC instruction
CR4.TSD	Time Stamp Disable - A method for a supervisor program to disable user access to the time stamp counter in secure systems. When bit 2 of CR4 is set to 1, an attempt to execute the RDTSC instruction generates an general protection exception (#GP).

Table 16-23. Model Specific Performance Monitoring Features

CTR0, CTR1	Counter 0, Counter 1 - two programmable counters
CESR	Control and Event Select Register - programs CTR0, CTR1
TSC	Time Stamp Counter - provides read and write access to the architectural 64-bit counter in a manner that is model specific.
PM0/BP0, PM1/BP1	Event Monitoring Pins - These pins allow external hardware to monitor the activity in CTR0 and CTR1.

16.4.2. Time Stamp Counter - TSC

A dedicated, free-running, 64-bit time stamp counter is provided on chip. Note that on the Pentium processor, this counter increments on every clock cycle, although it is not guaranteed that this will be true on future processors. As a time stamp counter, the RDTSC instruction reports values that are guaranteed to be unique and monotonically increasing. Portable software should not expect that the counter reports absolute time or clock counts. The user level RDTSC (Read Time Stamp Counter) instruction is provided to allow a program of any privilege level to sample its value. A bit in CR4, TSD (Time Stamp Disable) is provided to disable this instruction in secure environments. Supervisor mode programs may sample this counter using the RDMSR instruction or reset/preset this counter with a WRMSR instruction. The counter is cleared after reset.

While the user level RDTSC instruction and a corresponding 64-bit time stamp counter will be provided in all future Pentium CPU compatible processors, access to this counter via the RDMSR/WRMSR instructions is dependent upon the particular implementation.

16.4.3. Programmable Event Counters - CTR0, CTR1

Two programmable 40-bit counters CTR0 and CTR1 are provided. The implementation of these two counters is slightly different between the Pentium processor with MMX technology and the Pentium processor (75/90/100/120/133/150/166/200). In the Pentium processor (75/90/100/120/133/150/166/200) each counter may be programmed to count any event from a pre-determined list of events. These events, which are described in the *Events* section of this chapter, are selected by programming the Control and Event Select Register (CESR). In the Pentium processor with MMX technology some additional events were added and cannot be assigned to either of the two counters independently. These new events are paired, so when one event is assigned to counter 0, a second related event is automatically assigned to counter 1. The counters are not affected by writes to CESR and must be cleared or pre-set when switching to a new event. The counters are undefined after RESET.

Associated with each counter is an event pin (PM1/BP1, PM0/BP0) which externally signals the occurrence of the selected event.

Note that neither the CTR0/CTR1 nor CESR are part of the processor state that is automatically saved and restored during a context switch. If it is desired to coordinate the use of the programmable counters in a multiprocessing system, it is the software's responsibility to share or restrict the use of these counters through a semaphore or other appropriate mechanism.

16.4.4. Control and Event Select Register - CESR

A 32-bit Control and Event Select Register (CESR) is used to control operation of the programmable counters and their associated pins. Figure 16-6 depicts the CESR. For each counter, the CESR contains a 6-bit Event Select field (ES), a Pin Control bit (PC), and a three bit control field (CC). It is not possible to selectively write a subset of the CESR. If only one event needs to be changed, the CESR must first be read, the appropriate bits modified, and all bits must be written back. At reset, all bits in the Control and Event Select Register are cleared.

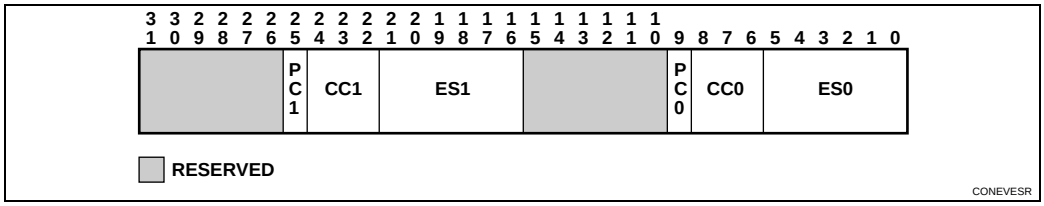


Figure 16-6. Control and Event Select Register

16.4.4.1. EVENT SELECT - ES0, ES1

Up to two events may be monitored by placing the appropriate event code in the Event Select field. The events and codes are listed in the Events section of this chapter.

16.4.4.2. COUNTER CONTROL - CC0, CC1

A three bit field is used to control the operation of the counter. the highest order bit selects between counting events and counting clocks. The middle bit enables counting when the CPL=3. The low order bit enables counting when the CPL=0, 1 or 2.

CC	Meaning
000	Count Nothing (Disable Counter)
001	Count the selected Event while the CPL=0, 1 or 2
010	Count the selected Event while the CPL=3
011	Count the selected Event regardless of the CPL

100	Count Nothing (Disable Counter)
101	Count Clocks while the CPL=0, 1 or 2
110	Count Clocks while the CPL=3
111	Count Clocks regardless of the CPL

While a counter need not be stopped to sample its contents, it must be stopped and cleared or pre-set before switching to a new event.

16.4.4.3. PIN CONTROL - PC0, PC1

Associated with CTR0 and CTR1 are two pins, PM0 and PM1 (PM0/BP0, PM1/BP1), and two bits which control their operation, PC0 and PC1. These pins may be programmed by the PC0/PC1 bits in the CESR to indicate either that the associated counter has incremented or that it has overflowed. Note that the external signalling of the event on the pins will lag the internal event by a "few" clocks as the signals are latched and buffered.

PC	PM pin signals when the corresponding counter:
0	has incremented
1	has overflowed

When the pins are configured to signal that a counter has incremented, it should be noted that although the counters may increment by 1 or 2 in a single clock, the pins can only indicate that the event occurred. Moreover, since the internal clock frequency may be higher than the external clock frequency, a single external clock may correspond to multiple internal clocks.

A "count up to" function may be provided when the event pin is programmed to signal an overflow of the counter. Because the counters are 40 bits, a carry out of bit 39 indicates an overflow. A counter may be preset to a specific value less than $2^{40} - 1$. After the counter has been enabled and the prescribed number of events has transpired, the counter will overflow. Approximately 5 clocks later, the overflow is indicated externally and appropriate action, such as signaling an interrupt, may then be taken.

When the performance monitor pins are configured to indicate when the performance monitor counter has incremented and an "occurrence event" is being counted, the associated PM pin is asserted (high) each time the event occurs. When a "duration event" is being counted the associated PM pin is asserted for the entire duration of the event. When the performance monitor pins are configured to indicate when the counter has overflowed, the associated PM pin is not asserted until the counter has overflowed.

The PM0/BP0, PM1/BP1 pins also serve to indicate breakpoint matches during in Circuit Emulation, during which time the counter increment or overflow function of these pins is not available. After RESET, the PM0/BP0, PM1/BP1 pins are configured for performance monitoring, however a hardware debugger may re-configure these pins to indicate breakpoint matches.

16.4.5. Performance Monitoring Events

Events may be considered to be of two types: those that count OCCURRENCES, and those that count DURATION. Each of the events listed below is classified accordingly.

Occurrences events are counted each time the event takes place. If the PM0 or PM1 pins are configured to indicate when a counter increments, they are asserted each clock the counter increments. Note that if an event can happen twice in one clock the counter increments by 2, however the PM0/1 pins are asserted only once.

For Duration events, the counter counts the total number of clocks that the condition is true. When configured to indicate when a counter increments, the PM0 and PM1 pins are asserted for the duration of the event.

Table 16-24 lists the events that can be counted, and their encodings for the Control and Event Select Register.

The performance monitoring features present in the Pentium processor (75/90/100/120/133/150/166/200) have been extended in the Pentium processor with MMX technology. The event list is longer, and there is a new instruction defined to facilitate use of the instruction monitoring. To leave room for future additions all new Pentium processor with MMX technology events are assigned to just one of the two events counters (CTR0, CTR1). It is not possible to assign these events to any of the two counters at will. “Twin events” (such as “D1 starvation and FIFO is empty”) are assigned to different counters to allow their concurrent measurement.

The Read Performance Monitoring Counter (RDPMC) is implemented in Pentium processor with MMX technology. See the *Intel Architecture Software Developer’s Manual* for more information about the RDPMC instruction.

Table 16-24. Performance Monitoring Events

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
0	000000	Yes	Yes	Data Read	Occurrence
1	000001	Yes	Yes	Data Write	Occurrence
2	000010	Yes	Yes	Data TLB Miss	Occurrence
3	000011	Yes	Yes	Data Read Miss	Occurrence
4	000100	Yes	Yes	Data Write Miss	Occurrence
5	000101	Yes	Yes	Write (hit) to M- or E-state lines	Occurrence
6	000110	Yes	Yes	Data Cache Lines Written Back	Occurrence
7	000111	Yes	Yes	External Snoops	Occurrence
8	001000	Yes	Yes	External Data Cache Snoop Hits	Occurrence
9	001001	Yes	Yes	Memory Accesses in Both Pipes	Occurrence
10	001010	Yes	Yes	Bank Conflicts	Occurrence
11	001011	Yes	Yes	Misaligned Data Memory or I/O References	Occurrence
12	001100	Yes	Yes	Code Read	Occurrence
13	001101	Yes	Yes	Code TLB Miss	Occurrence
14	001110	Yes	Yes	Code Cache Miss	Occurrence
15	001111	Yes	Yes	Any Segment Register Loaded	Occurrence
16	010000	Yes	Yes	Reserved	
17	010001	Yes	Yes	Reserved	
18	010010	Yes	Yes	Branches	Occurrence
19	010011	Yes	Yes	BTB Hits	Occurrence
20	010100	Yes	Yes	Taken Branch or BTB hit	Occurrence
21	010101	Yes	Yes	Pipeline Flushes	Occurrence
22	010110	Yes	Yes	Instructions Executed	Occurrence
23	010111	Yes	Yes	Instructions Executed in the v pipe e.g. parallelism/pairing	Occurrence
24	011000	Yes	Yes	Clocks while a bus cycle is in progress (bus utilization)	Duration
25	011001	Yes	Yes	Number of clocks stalled due to full write buffers	Duration

Table 16-24. Performance Monitoring Events (Contd.)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
26	011010	Yes	Yes	Pipeline stalled waiting for data memory read	Duration
27	011011	Yes	Yes	Stall on write to an E- or M-state line	Duration
28	011100	Yes	Yes	Locked Bus Cycle	Occurrence
29	011101	Yes	Yes	I/O Read or Write Cycle	Occurrence
30	011110	Yes	Yes	Non-cacheable memory reads	Occurrence
31	011111	Yes	Yes	Pipeline stalled because of an address generation interlock	Duration
32	100000	Yes	Yes	Reserved	
33	100001	Yes	Yes	Reserved	
34	100010	Yes	Yes	FLOPs	Occurrence
35	100011	Yes	Yes	Breakpoint match on DR0 Register	Occurrence
36	100100	Yes	Yes	Breakpoint match on DR1 Register	Occurrence
37	100101	Yes	Yes	Breakpoint match on DR2 Register	Occurrence
38	100110	Yes	Yes	Breakpoint match on DR3 Register	Occurrence
39	100111	Yes	Yes	Hardware Interrupts	Occurrence
40	101000	Yes	Yes	Data Read or Data Write	Occurrence
41	101001	Yes	Yes	Data Read Miss or Data Write Miss	Occurrence
42	101010	Yes	No	Bus Ownership Latency	Duration
42	101010	No	Yes	Bus Ownership Transfers	Occurrence
43	101011	Yes	No	MMX instructions executed in u pipe	Occurrence
43	101011	No	Yes	MMX instructions executed in v pipe	Occurrence
44	101100	Yes	No	Cache M-State Line Sharing	Occurrence
44	101100	No	Yes	Cache Line Sharing	Occurrence
45	101101	Yes	No	EMMS instructions executed	Occurrence
45	101101	No	Yes	Transition between MMX™ instructions and FP instructions	Occurrence
46	101110	Yes	No	Bus Utilization Due to processor Activity	Duration
46	101110	No	Yes	Writes to Non-Cacheable Memory	Occurrence

Table 16-24. Performance Monitoring Events (Contd.)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
47	101111	Yes	No	Saturating MMX instructions executed	Occurrence
47	101111	No	Yes	Saturations performed	Occurrence
48	110000	Yes	No	Number of Cycles Not in HLT State	Duration
48	110000	No	Yes	Number of Cycles Not in HLT State	Duration
49	110001	Yes	No	MMX instruction data reads	Occurrence
49	110001	No	Yes	MMX instruction data read misses	Occurrence
50	110010	Yes	No	Floating Point Stalls	Duration
50	110010	No	Yes	Taken Branches	Occurrence
51	110011	Yes	No	D1 Starvation and FIFO is empty	Occurrence
51	110011	No	Yes	D1 Starvation and only one instruction in FIFO	Occurrence
52	110100	Yes	No	MMX instruction data writes	Occurrence
52	110100	No	Yes	MMX instruction data write misses	Occurrence
53	110101	Yes	No	Pipeline flushes due to wrong branch prediction	Occurrence
53	110101	No	Yes	Pipeline flushes due to wrong branch predictions resolved in WB-stage	Occurrence
54	110110	Yes	No	Misaligned data memory reference on MMX instruction	Occurrence
54	110110	No	Yes	Pipeline stalled waiting for MMX instruction data memory read	Duration
55	110111	Yes	No	Returns Predicted Incorrectly or not predicted at all	Occurrence
55	110111	No	Yes	Returns Predicted (Correctly and Incorrectly)	Occurrence
56	111000	Yes	No	MMX multiply unit interlock	Duration
56	111000	No	Yes	MOVD/MOVQ store stall due to previous operation	Duration
57	111001	Yes	No	Returns	Occurrence
57	111001	No	Yes	Reserved	
58	111010	Yes	No	BTB false entries	Occurrence
58	111010	No	Yes	BTB miss prediction on a Not-Taken Branch	Occurrence

Table 16-24. Performance Monitoring Events (Contd.)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
59	111011	Yes	No	Number of clocks stalled due to full write buffers while executing MMX instructions	Duration
59	111011	No	Yes	Stall on MMX instruction write to E- or M-state line	Duration

NOTE:
Shaded areas only apply to the Pentium® processor with MMX™ technology.

16.4.6. Description of Events

The following descriptions clarify the events. The event codes are provided in parenthesis.

Data Read (0, 000000), Data Write (1, 000001), Data Read or Data Write (40, 101000):

These are memory data reads and/or writes (internal data cache hit and miss combined), I/O is not included. The individual component reads and writes for split cycles are counted individually. Data Memory Reads that are part of TLB miss processing are not included. These events may occur at a maximum of two per clock.

Data TLB Miss (2, 000010):

This event counts the number of misses to the data cache translation look-aside buffer.

Data Read Miss (3, 000011), Data Write Miss (4, 000100), Data Read Miss or Data Write Miss (41, 101001):

These are memory read and/or write accesses that miss the internal data cache whether or not the access is cacheable or non-cacheable. Additional reads to the same cache line after the first BRDY# of the burst linefill is returned but before the final (fourth) BRDY# has been returned, will not cause the Data Read Miss counter to be incremented additional times. Data accesses that are part of TLB miss processing are not included. Accesses directed to I/O space are not included.

Write (hit) to M- or E-state lines (5, 000101):

This measures the number of write hits to exclusive or modified lines in the data cache. (These are the writes which may be held up if EWBE# is inactive.) This event may occur at a maximum of two per clock.

Data Cache Lines Written Back (6, 000110):

This counts ALL Dirty lines that are written back, regardless of the cause. Replacements and internal and external snoops can all cause writeback and are counted.

External Snoops (7, 000111), Data Cache Snoop Hits (8, 001000):

The first event counts accepted external snoops whether they hit in the code cache or data cache or neither. Assertions of EADS# outside of the sampling interval are not counted. No internal snoops are counted. The second event applies to the data cache only. Snoop hits to a valid line in either the data cache, the data line fill buffer, or one of the write back buffers are all counted as hits.

Memory Accesses in Both Pipes (9, 001001):

Data memory reads or writes which are paired in the pipeline. Note that these accesses are not necessarily run in parallel due to cache misses, bank conflicts, etc.

Bank Conflicts (10, 001010):

These are the number of actual bank conflicts.

Misaligned Data Memory or I/O References (11, 001011):

Memory or I/O reads or writes that are misaligned. A two or four byte access is misaligned when it crosses a four byte boundary; an eight byte access is misaligned when it crosses an eight byte boundary. Ten byte accesses are treated as two separate accesses of eight and two bytes each.

Code Read (12, 001100), Code TLB Miss (13, 001101), Code Cache Miss (14, 001110):

Total instruction reads and reads that miss the code TLB or miss the internal code cache whether or not the read is cacheable or non-cacheable. Individual eight byte non-cacheable instruction reads are counted.

Any Segment Register Loaded (15, 001111):

Writes into any segment register in real or protected mode including the LDTR, GDTR, IDTR, and TR. Segment loads are caused by explicit segment register load instructions, far control transfers, and task switches. Far control transfers and task switches causing a privilege level change will signal this event twice. Note that interrupts and exceptions may initiate a far control transfer.

Branches (18, 010010):

In addition to taken conditional branches, jumps, calls, returns, software interrupts, and interrupt returns, the Pentium processor treats the following operations as causing taken branches: serializing instructions, VERR and VERW instructions, some segment descriptor loads, hardware interrupts (including FLUSH#), and programmatic exceptions that invoke a trap or fault handler. Both Taken and Not Taken Branches are counted. The pipe is not necessarily flushed. The number of branches actually executed is measured, not the number of predicted branches.

BTB Hits (19, 010011):

Hits are counted only for those instructions that are actually executed.

Taken Branch or BTB Hit (20, 010100):

This is a logical OR of taken branches and BTB hits (defined above). It represents an event that may cause a hit in the BTB. Specifically, it is either a candidate for a space in the BTB, or it is already in the BTB.

Pipeline Flushes (21, 010101):

BTB Misses on taken branches, mis-predictions, exceptions, interrupts, and some segment descriptor loads all cause pipeline flushes. This event counter will not be incremented for serializing instructions (serializing instructions cause the prefetch queue to be flushed but will not trigger the Pipeline Flushed event counter) and software interrups (software interrupts do not flush the pipeline).

Instructions Executed (22, 010110):

Up to two per clock. Invocations of a fault handler are considered instructions. All hardware and software interrupts and exceptions will also cause the count to be incremented. Repeat prefixed string instructions will only increment this counter once despite the fact that the repeat loop executes the same instruction multiple times until the loop criteria is satisfied. This applies to all the Repeat string instruction prefixes (i.e., REP, REPE, REPZ, REPNE, and REPNZ). This counter will also only increment once per each HLT instruction executed regardless of how many cycles the processor remains in the HALT state.

Instructions Executed in the v pipe e.g. parallelism/pairing (23, 010111):

Same as the Instructions exectued counter except it only counts the number of instructions actually executed in the v pipe. It indicates the number of instructions that were paired.

Clocks while a bus is in progress (bus utilization) (24, 011000):

Including HLDA, AHOLD, BOFF# clocks.

Number of clocks stalled due to full write buffers (25, 011001):

This event counts the number of clocks that the internal pipeline is stalled due to full write buffers. Full write buffers stall data memory read misses, data memory write misses, and data memory write hits to S state lines. Stalls on I/O accesses are not included.

Pipeline stalled waiting for data memory read (26, 011010):

Data TLB Miss processing is also included. The pipeline stalls while a data memory read is in progress including attempts to read that are not bypassed while a line is being filled.

Locked Bus Cycle (28, 011100):

LOCK prefix or LOCK instruction, Page Table Updates, and Descriptor Table Updates. Only the Read portion of the Locked Read-Modify-Write is counted. Split Locked cycles (SCYC active) count as two separate accesses. Cycles restarted due to BOFF# are not re-counted.

I/O Read or Write Cycle (29, 011101):

Bus cycles directed to I/O space. Misaligned I/O accesses will generate two bus cycles. Bus cycles restarted due to BOFF# are not re-counted.

Non-cacheable memory reads (30, 011110):

Non-cacheable instruction or data memory read bus cycles. Includes read cycles caused by TLB misses; does not include read cycles to I/O space. Cycles restarted due to BOFF# are not re-counted.

Pipeline stalled because of an address generation interlock (31, 011111):

Number of address generation interlocks (AGIs). An AGI occurring in both the u- and v-pipelines in the same clock signals this event twice. An AGI occurs when the instruction in the execute stage of either of u- or v-pipelines is writing to either the index or base address register of an instruction in the D2 (address generation) stage of either the u- or v-pipelines.

FLOPs (34, 100010);

Number of floating point adds, subtracts, multiplies, divides, remainders, and square roots. The transcendental instructions consist of multiple adds and multiplies and will signal this event multiple times. Instructions generating the divide by zero, negative square root, special operand, or stack exceptions will not be counted. Instructions generating all other floating point exceptions will be counted. The integer multiply instructions and other instructions which use the floating point arithmetic circuitry will be counted.

Breakpoint match on DR0 Register (35, 100011),

Breakpoint match on DR1 Register (36, 100100),

Breakpoint match on DR2 Register (37, 100101),

Breakpoint match on DR3 Register (38, 100110):

If programmed for one of these breakpoint match events, the performance monitor counters will be incremented in the event of a breakpoint match whether or not breakpoints are enabled. However, if breakpoints are not enabled, code breakpoint matches will not be checked for instructions executed in the v-pipe and will not cause this counter to be incremented (they are checked on instruction executed in the u-pipe only when breakpoints are not enabled). These events correspond to the signals driven on the BP[3:0] pins. Please refer to the Debugging chapter of this volume for more information.

Hardware Interrupts (39, 100111):

Number of taken INTR and NMI only.

Bus ownership latency (42, 101010/0), Bus ownership transfers (42, 101010/1):

The first event measures the time from LRM bus ownership request to bus ownership granted, the time from the earlier of PBREQ (0), PHITM# or HITM# to PBGNT. The second event is count of the number of PBREQ (0). The ratio of these two events is the average stall time due to bus ownership conflict.

MMX instructions executed in U pipe (43, 101011/0):

Total number of MMX instructions executed in U-pipe.

MMX instructions executed in V pipe (43, 101011/1):

Total number of MMX instructions executed in V-pipe.

Cache M-state line sharing (44, 101100/0):

Counts the number of times a processor identified a hit to a modified line due to a memory access in the other processor (PHITM (O)). If the average memory latencies of the system are known, this event enables the user to count the **Write Backs on PHITM(O)** penalty and the **Latency on Hit Modified(I)** penalty.

Cache line sharing (44, 101100/1):

Counts the number of shared data lines in the L1 cache (PHIT (O)).

EMMS instructions executed (45, 101101/0):

Counts number of EMMS instructions executed.

Transition between MMX instructions and FP instructions (45, 101101/1):

Counts first floating point instruction following any MMX instruction or first MMX instruction following a floating point instruction. May be used to estimate the penalty in transitions between FP state and MMX state. An even count indicates the processor is in MMX state. an odd count indicates it is in FP state.

Bus utilization due to processor activity (46, 101110/0):

Counts the number of clocks the bus is busy due to the processor's own activity, i.e. the bus activity which is caused by the CPU.

Writes to non-cacheable memory (46, 101110/1):

Counts the number of write accesses to non-cacheable memory. It includes write cycles caused by TLB misses and I/O write cycles. Cycles restarted due to BOFF# are not re-counted.

Saturating MMX instructions executed (47, 101111/0):

Counts saturating MMX instructions executed, independently of whether or not they actually saturated. Saturating MMX instructions may perform either add, subtract or pack operations.

Saturations performed (47, 101111/1):

Counts number of MMX instructions that used saturating arithmetic and that at least one of its results actually saturated; i.e., if an MMX instruction operating on four dwords saturated in three out of the four results, the counter will be incremented by one only.

Number of cycles not in HLT state (48, 110000/0):

This event counts the number of cycles the processor is not idle due to HLT instruction. This event will enable the user to calculate "net CPI". Note that during the time that the processor is executing the HLT instruction, the Time Stamp Counter is not disabled. Since this event is controlled by the Counter Controls CC0, CC1 it can be used to calculate the CPI at CPL=3 which the TSC cannot provide.

Clocks stalled on Data cache TLB miss (48, 110000/1):

Counts the number of clocks the pipeline is stalled due to a data cache translation look-aside buffer (TLB) miss. This is the same as the event with encoding 011010 (pipeline stalled waiting for data memory read), but only for TLB miss.

MMX instruction data reads (49, 110001/0):

Analogous to "Data reads," counting only MMX instruction accesses.

MMX instruction data read misses (49, 110001/1):

Analogous to "Data read misses," counting only MMX instruction accesses.

Floating Point stalls (50, 110010/0):

This event counts the number of clocks while pipe is stalled due to a floating-point freeze.

Taken Branches (50, 110010/1):

This event counts the number of taken branches.

D1 starvation and FIFO is empty (51, 110011/0), D1 starvation and only one instruction in FIFO (51, 110011/1):

The D1 stage can issue 0, 1, or 2 instructions per clock if those are available in an instructions FIFO buffer. The first event counts how many times D1 cannot issue ANY instructions since the FIFO buffer is empty. The second event counts how many times the D1-stage issues just a single instruction since the FIFO buffer had just one instruction ready. Combined with previously defined events, Instruction Executed (010110) and Instruction Executed in the V-pipe (010110), the second event enables the user to calculate the numbers of time pairing rules prevented issuing of two instructions.

MMX instruction data writes (52, 110001/1):

Analogous to "Data writes," counting only MMX instruction accesses.

MMX instruction data write misses (52, 110100/1):

Analogous to "Data write misses," counting only MMX instruction accesses.

Pipeline flushes due to wrong branch prediction (53, 110101/0), Pipeline flushes due to wrong branch prediction resolved in WB-stage(53, 110101/1):

Counts any pipeline flush due to a branch which the pipeline did not follow correctly. It includes cases where a branch was not in the BTB, cases where a branch was in the BTB but was mispredicted, and cases where a branch was correctly predicted but to the wrong address. Branches are resolved in either the Execute stage (E-stage) or the Writeback stage (WB-stage). In the later case, the misprediction penalty is larger by one clock. The two events count the number of pipeline flushes due to wrong branch predictions. The first event counts the number of wrong branch predictions resolved in either the E-stage or the WB-stage. The second event counts the number of wrong branch prediction resolved in the WB-stage. The difference between these two counts is the number of E-stage resolved branches.

Misaligned data memory reference on MMX instruction (54, 110110/0):

Analogous to "Misaligned data memory reference," counting only MMX instruction accesses.

Pipeline stalled waiting for MMX instruction data memory read (54, 110110/1):

Analogous to "Pipeline stalled waiting for data memory read," counting only MMX instruction accesses.

Returns predicted incorrectly or not predicted at all (55, 110111/0):

These are the actual number of Returns that were either incorrectly predicted or were not predicted at all. It is the difference between the total number of executed returns and the number of returns that were correctly predicted. Only RET instructions are counted (e.g., IRET instructions are not counted.).

Returns predicted (correctly and incorrectly) (55, 110111/1):

This is the actual number of Returns for which a prediction was made. Only RET instructions are counted (e.g. IRET instructions are not counted).

MMX multiply unit interlock (56, 111000/0):

This is the number of clocks the pipe is stalled since the destination of previous MMX multiply instruction is not ready yet. The counter will not be incremented if there is another cause for a stall. For each occurrence of a multiply interlock this event will be counted twice (if the stalled instruction comes on the next clock after the multiply) or by one (if the stalled instruction comes two clocks after the multiply).

MOVD/MOVQ store stall due to previous operation (56, 111000/1):

Number of clocks a MOVD/MOVQ store is stalled in D2 stage due to a previous MMX operation with a destination to be used in the store instruction.

Returns (57, 111001/0):

This is the actual number of Returns executed. Only RET instructions are counted (e.g., IRET instructions are not counted). Any exception taken on a RET instruction and any interrupt recognized by the processor on the instruction boundary prior to the execution of the RET instruction will also cause this counter to be incremented.

BTB false entries (58, 111010/0):

Counts the number of false entries in the Branch Target Buffer. False entries are causes for misprediction other than a wrong prediction.

BTB miss prediction on a Not-Taken Branch (58, 111010/1):

Counts the number of times the BTB predicted a Not-Taken branch as Taken.

Number of clocks stalled due to full write buffers while executing MMX instructions (59, 111011/0):

Analogous to "Number of clocks stalled due to full write buffers," counting only MMX instruction accesses.

Stall on MMX instruction write to an E- or M-state line (59, 111011/1):

Analogous to "Stall on write to an E- or M-state line," counting only MMX instruction accesses.

17

Pentium® OverDrive® Processor Socket Specification

CHAPTER 17
PENTIUM® OverDrive® PROCESSOR SOCKET SPECIFICATION

17.1. INTRODUCTION

This chapter includes the Socket 7 design specifications to support Pentium OverDrive processor upgrade products for 3.3V Pentium processor-based systems. Pentium OverDrive processors (125/150/166) are available today for upgradable Pentium processor-based systems (75/90/100). Future Pentium OverDrive processors with MMX technology are planned for upgradable Pentium processor-based systems (75/90/100/120/133/150/166/200[1]).

For the remainder of this chapter, the Pentium processor (75/90/100/120/133/150/166/200) will be referred to as the Pentium processor or 3.3V Pentium processor. When a specific core frequency of the Pentium is discussed, the core speed will be included. For example, the Pentium processor at 133 MHz will be described as the Pentium processor (133). The Pentium processor with MMX technology will always be so noted. The future Pentium OverDrive processor with MMX technology will also be distinguished from the existing Pentium OverDrive processor by stating that it has MMX technology.

Two upgrade sockets have been defined for Pentium processor-based systems as part of the processor architecture. Socket 5 has been defined for the Pentium processor-based systems with core frequencies from 75 MHz to 120 MHz. When upgraded, the Pentium processor is simply removed from the Socket 5 Zero Insertion Force (ZIF) socket and replaced by the Pentium OverDrive processor.

Socket 7 has been defined as the upgrade socket for Pentium processor (75/90/100/120/133/15/166/200)-based systems. The flexibility of the Socket 7 definition makes it backward compatible with Socket 5. Socket 7 should be used for all new Pentium processor-based system designs. Socket 7 supports the Pentium processor family, the Pentium OverDrive processor, and the future Pentium OverDrive processor with MMX technology. To support the future Pentium OverDrive processor with MMX technology, systems should be designed using the electrical specifications described in this chapter. Note that in some cases, the electrical specifications of the Pentium processor with MMX technology differ from the Socket 7 specifications. Systems designed to support the Pentium processor with MMX technology must adhere to the specifications outlined in Chapter 7 of this manual.

[1] The future 200-MHz Pentium OverDrive processor with MMX technology will provide the benefit of MMX technology in addition to some integer and floating point performance increase compared to the 200-MHz Pentium processor.

Socket 7 support requires minor changes from Socket 5 designs: additional power, 3.3V clocks, additional decoupling, etc. Pentium processor (133/150/166/200)-based systems must use Socket 7 in order to accept the future Pentium OverDrive processor with MMX technology.

The inclusion of Socket 7 in Pentium processor-based systems provides the end-user with a flexible and cost-effective way to increase system performance. The majority of upgrade installations will be performed by end-users and resellers; therefore, it is important that the design be "end-user easy," and that the amount of training and technical expertise required to install the upgrade processors be minimal. Upgrade installation instructions should be clearly described in the system user's manual. Three main characteristics of end-user friendly designs are:

- Accessible socket location
- Clear indication of upgrade component orientation
- Minimization of insertion force

The future Pentium OverDrive processor with MMX technology will support the 82430 PCIsets and other common chipsets that are supported by the Pentium processor. The 82497/82492 cache controller, 82498/82493 cache controller, and chipsets with 5 volt signal levels are not supported by the Pentium OverDrive processor with MMX technology.

The rest of this chapter describes the Socket 7 specification.

17.2. SOCKET 7 DIFFERENCES FROM SOCKET 5

This section contains general information concerning the upgrade socket for the future Pentium OverDrive processor with MMX technology (Socket 7). Socket 7 is an enhancement to the Socket 5 definition that allows for future upgradability and installation of the future Pentium OverDrive processor with MMX technology. The socket is a 321-pin ZIF socket compatible to the 320-pin Socket 5 pinout with minimal system design changes. Throughout this document, any significant differences that exist between Socket 5 and Socket 7 that would impact a system design will be detailed.

The major differences are summarized below:

- Socket 7 requires 5.0 amps at 3.3 volts to support all Pentium processors (75/90/100/120/133/150/166/200) and the future Pentium OverDrive processor with MMX technology. The Socket 5 specification only requires 4.33 amps.
- Socket 7 requires that CLK and PICCLK be driven at 3.3 volt levels. CLK and PICCLK are not 5 volt tolerant on Pentium processors with MMX technology.
- The maximum power dissipation for Socket 7 is 17 watts, two watts higher than the 15 watt Socket 5 definition. The future Pentium OverDrive processor with MMX technology supports a maximum ambient temperature of 45°C at the fan intake.
- The future Pentium OverDrive processor with MMX technology requires a slightly larger heatsink for proper cooling. This increase in heatsink height raises the required vertical clearance to 1.75" compared to 1.35" for Socket 5. Socket 7 also enables the use of

standard heatsink clips. The location of these clips will be consistent on all qualified sockets and are available for OEM and upgrade processor use.

- The pinout of Socket 7 is compatible with Socket 5 with the addition of a mechanical key pin at location AH32 and VCC2DET# at AL01. AH32 is defined as an internal no-connect for Socket 7 compatible processors.
- Socket 7 specification provides the capability to support dual voltage supply processors. See the specifications for the Pentium processor with MMX technology for more information.

Upgradability for Socket 7 is implemented through a single socket, processor replacement approach. The OEM processor is always installed in the Socket 7 at the factory. When this system configuration is upgraded, the end user removes the original CPU and installs the Pentium OverDrive processor.

17.3. SOCKET 7 PINOUT

The upgrade socket configuration consists of a 321-pin Socket 7. Socket 7 is a superset of Socket 5 with one additional pin at AH32. This position can be used to key the socket and prevent components designed only for Socket 7 from being installed in systems with Socket 5 configuration. This additional pin is a mechanical key pin and does not need to be routed on the motherboard.

To support flexible designs for both unified and split plane processors, Socket 7 has defined two sets of power pins, $V_{CC}2$ and $V_{CC}3$. (For information on $V_{CC}2$ and $V_{CC}3$ pins, refer to section 17.3.2 of this chapter.) Socket 7 also defines the pin at AL01 as VCC2DET#. For information on the use of VCC2DET# refer to section 17.3.3 of this chapter.

17.3.1. Socket 7 Pin Diagrams

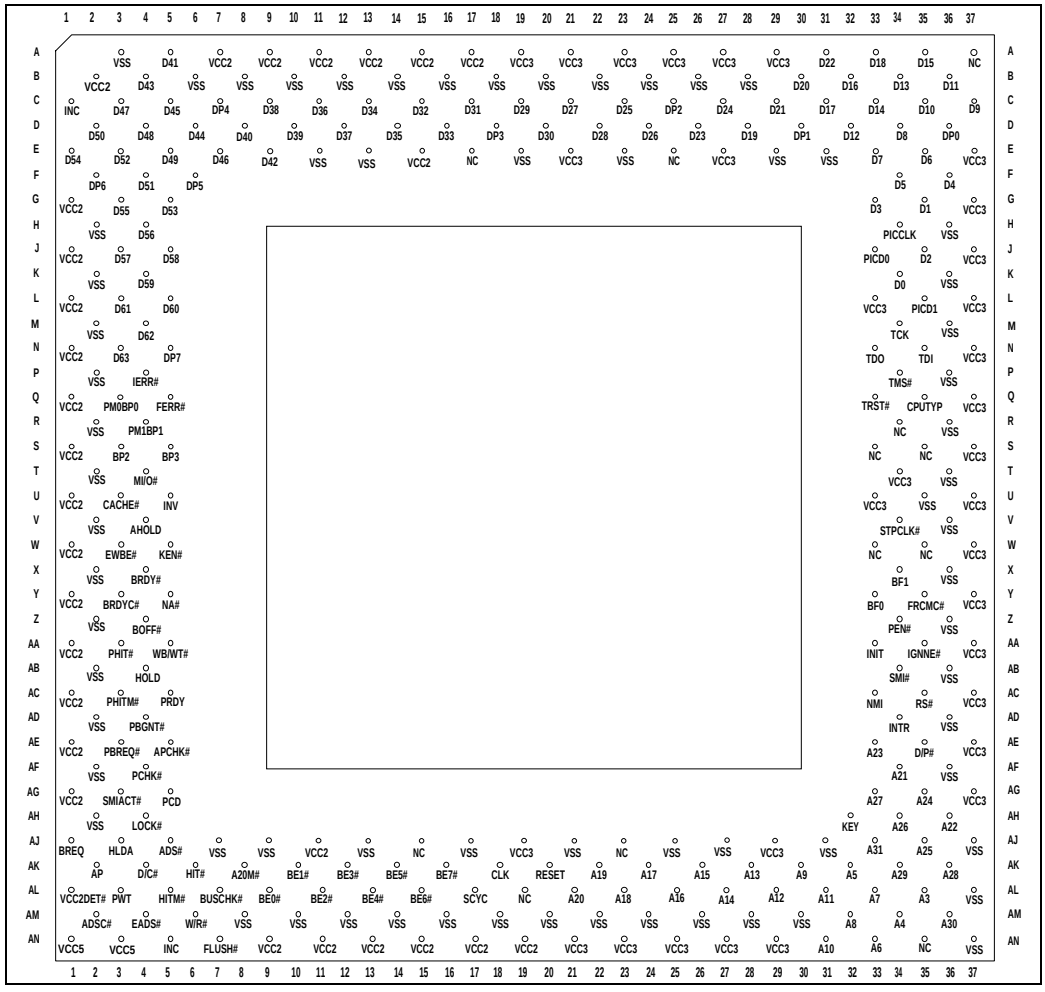


Figure 17-1. Socket 7 Pinout - Top Side View

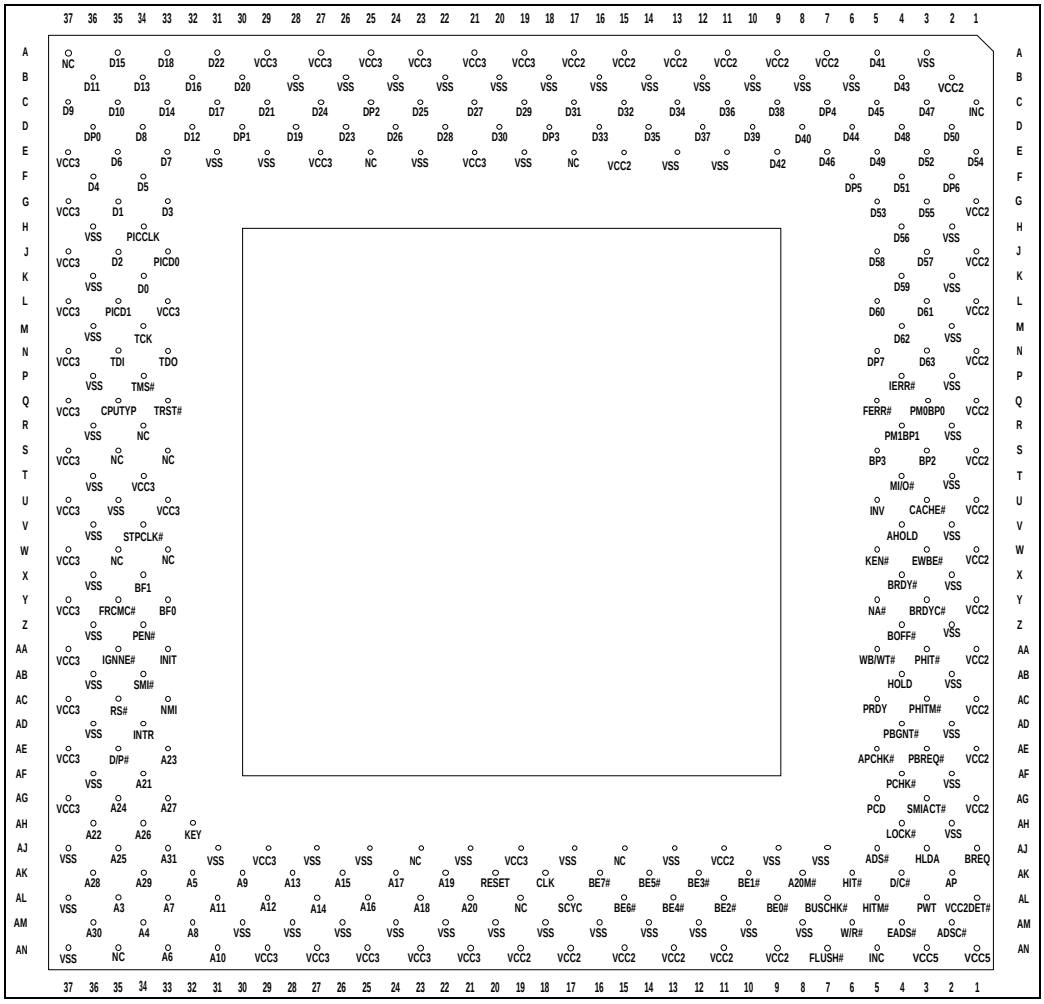


Figure 17-2. Socket 7 Pinout - Pin Side View

17.3.2. Socket 7 Pin Cross Reference Table

In Table 17-1, Socket 7 pins which differ from the Socket 5 pinout or definition are shaded.

Table 17-1. Pin Cross Reference by Pin Name

Address									
A3	AL35	A9	AK30	A15	AK26	A21	AF34	A27	AG33
A4	AM34	A10	AN31	A16	AL25	A22	AH36	A28	AK36
A5	AK32	A11	AL31	A17	AK24	A23	AE33	A29	AK34
A6	AN33	A12	AL29	A18	AL23	A24	AG35	A30	AM36
A7	AL33	A13	AK28	A19	AK22	A25	AJ35	A31	AJ33
A8	AM32	A14	AL27	A20	AL21	A26	AH34		
Data									
D0	K34	D13	B34	D26	D24	D39	D10	D52	E03
D1	G35	D14	C33	D27	C21	D40	D08	D53	G05
D2	J35	D15	A35	D28	D22	D41	A05	D54	E01
D3	G33	D16	B32	D29	C19	D42	E09	D55	G03
D4	F36	D17	C31	D30	D20	D43	B04	D56	H04
D5	F34	D18	A33	D31	C17	D44	D06	D57	J03
D6	E35	D19	D28	D32	C15	D45	C05	D58	J05
D7	E33	D20	B30	D33	D16	D46	E07	D59	K04
D8	D34	D21	C29	D34	C13	D47	C03	D60	L05
D9	C37	D22	A31	D35	D14	D48	D04	D61	L03
D10	C35	D23	D26	D36	C11	D49	E05	D62	M04
D11	B36	D24	C27	D37	D12	D50	D02	D63	N03
D12	D32	D25	C23	D38	C09	D51	F04		

Table 17-1. Pin Cross Reference by Pin Name (Contd.)

Control							
A20M#	AK08	BREQ	AJ01	HIT#	AK06	PRDY	AC05
ADS#	AJ05	BUSCHK#	AL07	HITM#	AL05	PWT	AL03
ADSC#	AM02	CACHE#	U03	HLDA	AJ03	R/S#	AC35
AHOLD	V04	CPUTYP	Q35	HOLD	AB04	RESET	AK20
AP	AK02	D/C#	AK04	IERR#	P04	SCYC	AL17
APCHK#	AE05	D/P#	AE35	IGNNE#	AA35	SMI#	AB34
BE0#	AL09	DP0	D36	INIT	AA33	SMIACT#	AG03
BE1#	AK10	DP1	D30	INTR/LINT0	AD34	TCK	M34
BE2#	AL11	DP2	C25	INV	U05	TDI	N35
BE3#	AK12	DP3	D18	KEN#	W05	TDO	N33
BE4#	AL13	DP4	C07	LOCK#	AH04	TMS	P34
BE5#	AK14	DP5	F06	M/IO#	T04	TRST#	Q33
BE6#	AL15	DP6	F02	NA#	Y05	VCC2DET#	AL01
BE7#	AK16	DP7	N05	NMI/LINT1	AC33	W/R#	AM06
BOFF#	Z04	EADS#	AM04	PCD	AG05	WB/WT#	AA05
BP2	S03	EWBE#	W03	PCHK#	AF04		
BP3	S05	FERR#	Q05	PEN#	Z34		
BRDY#	X04	FLUSH#	AN07	PM0/BP0	Q03		
BRDYC#	Y03	FRCMC#	Y35	PM1/BP1	R04		

APIC		Clock Control		Dual Processor Private Interface	
PICCLK	H34	CLK	AK18	PBGNT#	AD04
PICD0	J33	[BF0]	Y33	PBREQ#	AE03
[DPEN#]		[BF1]	X34	PHIT#	AA03
PICD1	L35	STPCLK#	V34	PHITM#	AC03
[APICEN]					

Table 17-1. Pin Cross Reference by Pin Name (Contd.)

$V_{CC}2$						
A07	A15	G01	Q01	Y01	AG01	AN13
A09	A17	J01	S01	AA01	AJ11	AN15
A11	B02	L01	U01	AC01	AN09	AN17
A13	E15	N01	W01	AE01	AN11	AN19

$V_{CC}3$							
A19	A27	E37	L37	T34	Y37	AG37	AN23
A21	A29	G37	N37	U33	AA37	AJ19	AN25
A23	E21	J37	Q37	U37	AC37	AJ29	AN27
A25	E27	L33	S37	W37	AE37	AN21	AN29

V_{SS}								
A03	B20	E23	M36	V02	AD02	AJ17	AM10	AM26
B06	B22	E29	P02	V36	AD36	AJ21	AM12	AM28
B08	B24	E31	P36	X02	AF02	AJ25	AM14	AM30
B10	B26	H02	R02	X36	AF36	AJ27	AM16	AN37
B12	B28	H36	R36	Z02	AH02	AJ31	AM18	
B14	E11	K02	T02	Z36	AJ07	AJ37	AM20	
B16	E13	K36	T36	AB02	AJ09	AL37	AM22	
B18	E19	M02	U35	AB36	AJ13	AM08	AM24	

NC			INC	$V_{CC}5$	KEY
A37	S33	AJ15	C01	AN01	AH32
E17	S35	AJ23	AN05	AN03	
E25	W33	AL19			
R34	W35	AN35			

17.3.3. Socket 7 Quick Pin Reference

With the exception of the pins in Table 17-2, Socket 7 has the same pin definition as Socket 5.

Table 17-2. Socket 7 Quick Pin Reference

Symbol	Type	Name and Function
CLK, PICCLK	(I)	The **Clock** and **Programmable Interrupt Controller** *Clock* inputs to Socket 7 are **not** 5V tolerant. These inputs must be driven by an appropriate 3.3V clock driver.
KEY	NA	The **KEY** pin is strictly a mechanical keying device. The corresponding pin on the processors is an Internal No Connect and has no electrical purpose.
VCC2DET#	(O)	**V_{CC}2 Detect** is defined to identify processors that require the system to supply a lower voltage on the V_{CC}2 power inputs as compared to the V_{CC}3 inputs. This pin is defined only on the Pentium® processor with MMX™ technology and can be used in flexible motherboard implementations to correctly set the voltage regulator to supply the appropriate voltage to the core (V_{CC}2) power pins. This pin is internally strapped to V_{SS} on the Pentium processor with MMX technology. This pin is an INC on the Pentium processor (75/90/100/120/133/150/166/200).
V_{CC}2	(I)	Socket 7 has 28 power supply pins defined for the core voltage on processors with separate power inputs. For processors with a single power supply requirement, these pins can be considered the same as V_{CC}3 pins and should be driven with the same power source.
V_{CC}3	(I)	Socket 7 has 32 power supply pins defined for the I/O voltage on processors with separate power inputs. For processors with a single supply requirement, these pins are used in conjunction with the V_{CC}2 pins to power the device.

17.4. HARDWARE SYSTEM DESIGN CONSIDERATIONS

17.4.1. Bus Fraction Selection

The Pentium OverDrive processors do not require the bus frequency ratio (BF0 and BF1 pins) to be changed when upgrading a system. The future Pentium OverDrive processor with MMX technology will be internally bonded to the proper bus/core multiplier. See the specific Pentium processor specification requirements for pullup and pulldown resister values on the BF pins.

17.4.2. Power Supply Considerations for Split and Unified Power Planes

17.4.2.1. POWER SUPPLY CONSIDERATIONS FOR SOCKET 7

Socket 7 has defined 28 $V_{CC}2$ power pins and 32 $V_{CC}3$ power pins as well as 53 V_{SS} pins. Ground connections should be made to all the V_{SS} pins in a Socket 7 motherboard design. Note that the Pentium OverDrive processor and the future Pentium OverDrive processor with MMX technology are both capable of accepting 3.3V levels. The connection of the $V_{CC}2$ pins and $V_{CC}3$ pins may be implemented several distinct ways. Motherboards may be designed which implement a non-flexible Socket 7 design supporting only single voltage processors or a flexible Socket 7 design supporting single and dual voltage processors.

17.4.2.2. NON-FLEXIBLE SOCKET 7 POWER IMPLEMENTATIONS

A Socket 7 design with a single external power supply capable of suppling only one voltage to the socket will not support the Pentium processor with MMX technology. In such designs, the power supply must be connected to ALL the $V_{CC}2$ pins as well as ALL of the $V_{CC}3$ pins. The voltage to ALL of the $V_{CC}2$ and $V_{CC}3$ pins must remain within the operating range for the processor. The Pentium processor and the Pentium OverDrive processor are both single voltage processors and have an internally unified power plane. The future Pentium OverDrive processor with MMX technology is also a single voltage processor but it has two internal power planes. When placed in a non-flexible Socket 7 motherboard design, the future Pentium OverDrive processor with MMX technology will receive the same voltage at both the $V_{CC}2$ pins and the $V_{CC}3$ pins. It is imperative that all of the $V_{CC}2$ and $V_{CC}3$ pins are connected to the power plane.

17.4.2.3. FLEXIBLE SOCKET 7 POWER IMPLEMENTATIONS

A flexible Socket 7 motherboard is able to support single and dual voltage processors. Flexible designs fall into two groups determined by their response to unified and split plane processors. There are:

1. Unified plane processors receive power externally to ALL the $V_{CC}2$ pins and ALL the $V_{CC}3$ pins. This can be from a single source with the two power planes externally stitched together at boot up time through VCC2DET# circuitry or from dual sources with current sharing circuitry. Split plane processors receive power from independent sources for $V_{CC}2$ and $V_{CC}3$.
2. Unified plane processors receive power externally to either the $V_{CC}2$ or $V_{CC}3$ pins. Split plane processors receive power from independent sources for $V_{CC}2$ and $V_{CC}3$.

The second of these implementations implies that a unified processor causes one of the system's voltage regulators to shut down upon power up. Since the output set points of the two voltage regulators will not be exactly the same, connecting the two outputs with the processor's internally unified power plane will cause one of them to shut down. This will leave the

processor to be powered by either ALL of the $V_{CC}2$ pins or ALL of the $V_{CC}3$ pins. This is acceptable as long as the current flowing through the processor package does not exceed 8 amps, including the power required by the processor. Also the voltage regulator that remains in operation must be able to provide enough current for the entire system. When a split plane processor is installed, both voltage regulators must continue to function at the proper voltage levels. Since the future Pentium OverDrive processor with MMX technology does not assert VCC2DET#, it is not acceptable to use the VCC2DET# signal to determine whether to power only $V_{CC}2$ or $V_{CC}3$. Note that although the future Pentium OverDrive processor with MMX technology is a single voltage processor, the voltage supplied to the $V_{CC}2$ pins is not required to be exactly the same as the voltage supplied to the $V_{CC}3$ pins as long as they are within the specified votage range. Refer to Chapter 7 of this document for the Pentium processor with MMX technology voltage specifications.

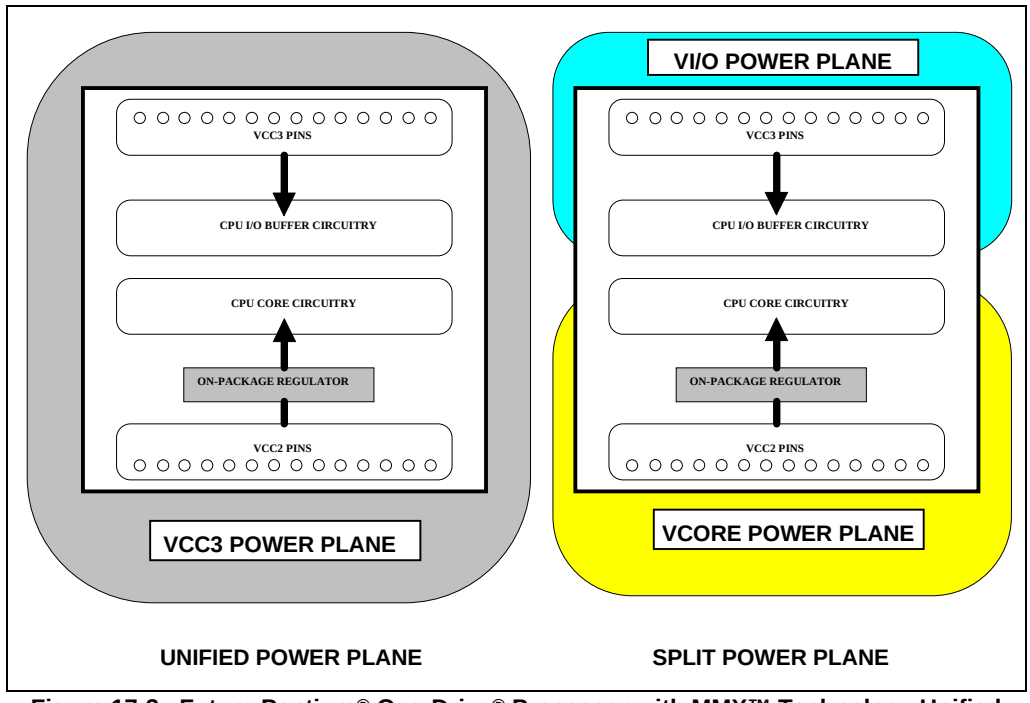


Figure 17-3. Future Pentium® OverDrive® Processor with MMX™ Technology Unified and Split Power Plane Designs

17.4.3. 3.3 Volt Clocks

Socket 7 requires 3.3V clock buffers. The clock driver should be powered by the 3.3 volt processor power plane. Routing of the clocks should be done on the trace layer adjacent to the

V_{SS} layer. If the clock must be routed on the signal layer adjacent to the V_{CC} plane, avoid routing above the 5V area or crossing the split between the 5 volt and 3.3 volt power planes.

17.5. SOFTWARE SYSTEM DESIGN CONSIDERATIONS

17.5.1. CPU Identification

Socket 7 is the universal socket for the "flexible motherboard" and may be populated with a variety of processors. The BIOS should use the CPUID instruction to obtain the CPU signature and features. The various CPUs may have different caches, test registers, and core architecture. Use the signature and feature registers to identify the device rather than making assumptions about the architecture of the installed device.

The CPU identification can be obtained from the EDX register after RESET or through use of the CPUID instruction. The signature for the various processors is noted below:

Table 17-3. CPU Signatures

Pentium® processors (75/90/100/120/133/150/166/200) Pentium processor with MMX™ technology	052xH or 057xH 054xH	Note: Dual processor in second socket will have a Type of "2". For example, a Pentium processor (75/90/100/120/133/150/166/200) in the second socket will be 252xH socket
Pentium OverDrive® processor Future Pentium OverDrive processor with MMX technology	152xH * 154xH	

* Note that the CPUID should be 152xH; however, an errata exists such that it is actually 052xH. See errata notes in the Pentium Processor Specification Update.

For complete details on how to differentiate between processors by using the CPUID instruction, refer to "AP-485, Intel Processor Identification and the CPUID Instruction" (Intel literature Order #241618).

17.5.2. Code Prefetch Queue and Branch Target Buffers

Code should not be written to rely on the specific code prefetch queue or branch target buffer mechanism of a particular processor. With each new generation and family of processors, these mechanisms are subject to change.

Software timing loops that rely on the code execution speed of the processor should never be used. Future Pentium OverDrive processors will execute a given code loop faster than the original processor, causing the execution time of the loop to change.

17.5.3. Model Specific Registers and Test Registers

Model Specific Registers (MSRs) are subject to change between the future OverDrive processor and the original processor it replaces. In addition, MSRs are subject to change with new processor generations and even between processor steppings.

Test registers can be a valuable debug tool. However, production code should not utilize them as they are subject to change with new processor generations.

17.5.4. Intel Architecture MMX™ Technology

Intel's MMX technology is an extension to the Intel architecture which provides for additional performance on multimedia and communications applications. Intel processors that include this technology are still 100% compatible with all "scalar" Intel processors. This means that all existing software that runs on existing Intel processors will continue to run on Intel processors that incorporate MMX technology without modification.

The future Pentium OverDrive processor with MMX technology includes the MMX instruction set. Software can determine that the system has been upgraded to a Intel Architecture processor that supports MMX technology via the CPUID instruction. By loading EAX=1 and then executing the CPUID instruction, the EDX register will reflect the processor feature flags. Bit 23 in EDX will be set to a 1 if the processor supports the MMX instruction set:

```
...
mov     EAX,  1          ;Request feature flags
        CPUID            ;0Fh,0A2h CPUID instruction
test    EDX, 00800000h   ;Is MMX bit (bit#23 of EDX) in feature flag set?
jnz     MMXFound
...
```

If the support for MMX technology is detected, the software can then enable MMX optimized code, drivers, DLLs, etc. to take advantage of the MMX instruction set. Because the future Pentium OverDrive processor with MMX technology will be used to upgrade existing systems, detection of MMX technology by software should be "run-time" versus "install-time".

For the complete definition of the MMX instruction set, see the Architecture and Programming manual within the *Intel Architecture Software Developer's Manual*.

17.6. ELECTRICAL SPECIFICATIONS

This section describes the Socket 7 electrical differences from Socket 5, the Socket 7 DC specifications, and the I/O buffer models of the processors for Socket 7. Socket 7 AC specifications are the same as the Pentium processor.

17.6.1. Socket 7 Electrical Differences from Socket 5

When designing a Socket 7 system based upon an existing Socket 5 system, there are a number of electrical differences that require attention.

17.6.1.1. 3.3 VOLT SUPPLY CURRENT INCREASE

Socket 7 requires that the 3.3 volt supply support 5 amps for the future Pentium OverDrive processor with MMX technology on unified plane designs. The Pentium processor with MMX technology requires a different split plane power supply. Please refer to the Pentium processor with MMX technology electrical specifications for information.

17.6.1.2. CLK AND PICCLK ARE NOT 5V TOLERANT

The clock inputs on Socket 5 are 5 volt tolerant, meaning that the clock V_{IH} may drive as high as 5.55 volts. Processors that support 5 volt clock inputs will also accept 3.3V clock inputs. Socket 7 specifications require 3.3V clock inputs. See section 17.6.4, for more information on the 3.3 volt clock specification.

17.6.1.3. INPUTS AND OUTPUTS

All inputs and outputs of the Socket 7 are 3.3 volt JEDEC standard levels. Both inputs and outputs are also TTL compatible although the inputs cannot tolerate voltage swings above the 3.3 volt V_{IH} maximum. The processor outputs will interface to standard system components with TTL compatible inputs without extra logic. The processors will drive the output according to the TTL specification (but not beyond 3.3 volts). System support components can consist of 3.3 volt devices or open-collector devices. In an open-collector configuration, the external resistor may be biased to $V_{CC}3$.

17.6.1.4. PROCESSOR BUFFER MODEL

The goal of the future Pentium OverDrive processor with MMX technology is to have Pentium processor compatible buffers. However, minor changes have been made in the clock input. The standard “Input Buffer Model” for the future Pentium OverDrive processor with MMX technology should be used for CLK rather than the “Input Buffer Model for Special Group”. Refer to Chapter 8 for additional information regarding buffer types.

17.6.1.5. DUAL POWER SOURCE CAPABILITY

Socket 7 has the capability to support split power plane processors such as the Pentium processor with MMX technology. Refer to the processor specifications for further information.

17.6.2. Absolute Maximum Ratings

Please refer to processor specifications for the absolute maximum ratings for an individual processor. Also refer to the socket vendor specifications for the maximum ratings for the socket implementation.

17.6.3. DC Specifications

Table 17-4. Socket 7 DC Specifications

Operating Conditions for the Pentium® OverDrive® Processor and the Future Pentium OverDrive Processor with MMX™ Technology: $V_{CC}2$ = 3.135V - 3.6V, $V_{CC}3$ = 3.135 - 3.6V, T_A = 0 to 45°C					
Symbol	**Parameter**	**Min**	**Max**	**Unit**	**Notes**
V_{IL}	Input Low Voltage	-0.3	0.8	V	TTL Level
V_{IH}	Input High Voltage	2.0	$V_{CC}3$+0.3	V	TTL Level
V_{OL}	Output Low Voltage		0.4	V	TTL Level (1)
V_{OH}	Output High Voltage	2.4		V	TTL Level (2)
Unified Power Plane					
$I_{CC}3$ + $I_{CC}2$	Power Supply Current		5000	mA	(3)(4)(5)
Split Power Plane					
$I_{CC}3$	I/O Power Supply Current		400	mA	(5)(6)
$I_{CC}2$	Core Power Supply Current		4600	mA	(5)(6)
5 Volt Power Plane					
$I_{CC}5$	Fan/Heatsink Current		200	mA	

NOTES:

1. Parameter measured at -4 mA.
2. Parameter measured at 3 mA.
3. This value should be used for power supply design. It estimated for a worst case instruction mix and V_{CC} + 5%. Power supply transient response and decoupling capacitors must be sufficient to handle the instantaneous current changes occurring during transitions from Stop Clock to full Active modes.
4. This is the value for the unified power plane and split power plane designs for Pentium® processors and the current Pentium OverDrive® processors.
5. These values are different from the Socket 5 specification.
6. These I_{CC} values relate to the split plane implementation of a future Pentium OverDrive processor with MMX™ technology-based system where both $V_{CC}2$ and $V_{CC}3$ are 3.3 volts. The Pentium processor with MMX technology requires a different split plane power supply. Please see the Electrical Specifications chapter of this document for further details..

Table 17-5. Input and Output Characteristics

Symbol	Parameter	Min	Max	Unit	Notes
C_{IN}	Input Capacitance		15	pF	(1)
C_O	Output Capacitance		20	pF	(1)
$C_{I/O}$	I/O Capacitance		25	pF	(1)
C_{CLK}	CLK Input Capacitance		15	pF	(1)
C_{TIN}	Test Input Capacitance		15	pF	(1)
C_{TOUT}	Test Output Capacitance		20	pF	(1)
C_{TCK}	Test Clock Capacitance		15	pF	(1)
I_{LI}	Input Leakage Current		± 15	µA	$0 < V_{IN} < V_{IL}$ $V_{IH} > V_{IN} > V_{CC}$ (2)
I_{LO}	Output Leakage Current		± 15	µA	$0 < V_{IN} < V_{IL}$ $V_{IH} > V_{IN} > V_{CC}$ (2)
I_{IL}	Input Leakage Current		-400 200 ±200	µA	V_{IN} = 0.4V (3) V_{IN} = 2.4V (4) $0 < V_{IN} < V_{CC}$3 (5)

NOTES:
1. Guaranteed by design.
2. This parameter is for input without pullup or pulldown.
3. This parameter is for input with pullup.
4. This parameter is for input with pulldown.
5. This parameter is for CLK and PICCLK.

Table 17-6. Power Dissipation for Thermal Design

Parameter	Max (1)	Unit	Notes
Active Power Dissipation	17	Watts	(2) (3)

NOTES:
1. Systems must be designed to thermally dissipate the maximum active power dissipation. It is determined using worst case instruction mix with nominal V_{CC} and also takes into account the thermal time constants of the package.
2. Maximum power dissipation of the future Pentium® OverDrive® processor with MMX™ technology.
3. This is different than the Socket 5 specifications.

17.6.4. 3.3 Volt Clock Signal Specifications

The maximum overshoot, maximum undershoot, overshoot threshold duration, undershoot threshold duration, and maximum ringback specifications for CLK/PICCLK are described below:

MAXIMUM OVERSHOOT AND MAXIMUM UNDERSHOOT SPECIFICATION: The maximum overshoot of the CLK/PICCLK signals should not exceed $V_{CC}3$,nominal + 0.9V. The maximum undershoot of the CLK/PICCLK signals must not drop below -0.9V.

OVERSHOOT THRESHOLD DURATION SPECIFICATION: The overshoot threshold duration is defined as the sum of all time during which the CLK/PICCLK signal is above $V_{CC}3$,nominal + 0.5V within a single clock period. The overshoot threshold duration must not exceed 20% of the period.

UNDERSHOOT THRESHOLD DURATION SPECIFICATION: The undershoot threshold duration is defined as the sum of all time during which the CLK/PICCLK signal is below -0.5V within a single clock period. The undershoot threshold duration must not exceed 20% of the period.

MAXIMUM RINGBACK SPECIFICATION: The maximum ringback of CLK/PICCLK associated with their high states (overshoot) must not drop below $V_{CC}3$ - 0.8V as shown in Figure 17-5. Similarly, the maximum ringback of CLK/PICCLK associated with their low states (undershoot) must not exceed 0.8V as shown in Figure 17-7.

Refer to Table 17-7 and Table 17-8 for a summary of the clock overshoot and undershoot specifications for the Pentium processor with MMX technology.

17.6.4.1. CLOCK SIGNAL MEASUREMENT METHODOLOGY

The waveform of the clock signals should be measured at the bottom side of the processor pins using an oscilloscope with a 3 dB bandwidth of at least 20 MHz (100 MS/s digital sampling rate). There should be a short isolation ground lead attached to a processor pin on the bottom side of the board. An 1 MOhm probe with loading of less than 1 pF (e.g., Tektronics 6243 or

Tektronics 6245) is recommended. The measurement should be taken at the CLK (AK18) and PICCLK (H34) pins and their nearest V_{SS} pins (AM18 and H36, respectively).

MAXIMUM OVERSHOOT, MAXIMUM UNDERSHOOT AND MAXIMUM RINGBACK SPECIFICATIONS: The display should show continuous sampling (e.g., infinite persistence) of the waveform at 500 mV/div and 5 nS/div (for CLK) or 20 nS/div (for PICCLK) for a recommended duration of approximately five seconds. Adjust the vertical position to measure the maximum overshoot and associated ringback with the largest possible granularity. Similarly, readjust the vertical position to measure the maximum undershoot and associated ringback. There is no allowance for crossing the maximum overshoot, maximum undershoot or maximum ringback specifications.

OVERSHOOT THRESHOLD DURATION SPECIFICATION: A snapshot of the clock signal should be taken at 500 mV/div and 500 pS/div (for CLK) or 2 nS/div (for PICCLK). Adjust the vertical position and horizontal offset position to view the threshold duration. The overshoot threshold duration is defined as the sum of all time during which the clock signal is above $V_{CC}3$,nominal + 0.5V within a single clock period. The overshoot threshold duration must not exceed 20% of the period.

UNDERSHOOT THRESHOLD DURATION SPECIFICATION: A snapshot of the clock signal should be taken at 500 mV/div and 500 pS/div (for CLK) or 2 nS/div (for PICCLK). Adjust the vertical position and horizontal offset position to view the threshold duration. The undershoot threshold duration is defined as the sum of all time during which the clock signal is below -0.5V within a single clock period. The undershoot threshold duration must not exceed 20% of the period.

These overshoot and undershoot specifications are illustrated graphically in Figure 17-4 to Figure 17-7.

Table 17-7. Overshoot Specification Summary

Specification Name	Value	Units	Notes
Threshold Level	$V_{CC}3$,nominal + 0.5	V	(1) (2)
Maximum Overshoot Level	$V_{CC}3$,nominal + 0.9	V	(1) (2)
Maximum Threshold Duration	20% of clock period above threshold voltage	nS	(2)
Maximum Ringback	$V_{CC}3$,nominal - 0.8	V	(1) (2)

NOTES:

1. $V_{CC}3$, nominal refers to the voltage measured at the bottom side of the $V_{CC}3$ pins. See Section 7.1.2.1.1 for details.
2. See Figure 17-4 and Figure 17-5.

Table 17-8. Undershoot Specification Summary

Specification Name	Value	Units	Notes
Threshold Level	-0.5	V	(1)
Minimum Undershoot Level	-0.9	V	(1)
Maximum Threshold Duration	20% of clock period below threshold voltage	nS	(1)
Maximum Ringback	0.8	V	(1)

NOTE:

1. See Figure 17-6 and Figure 17-7.

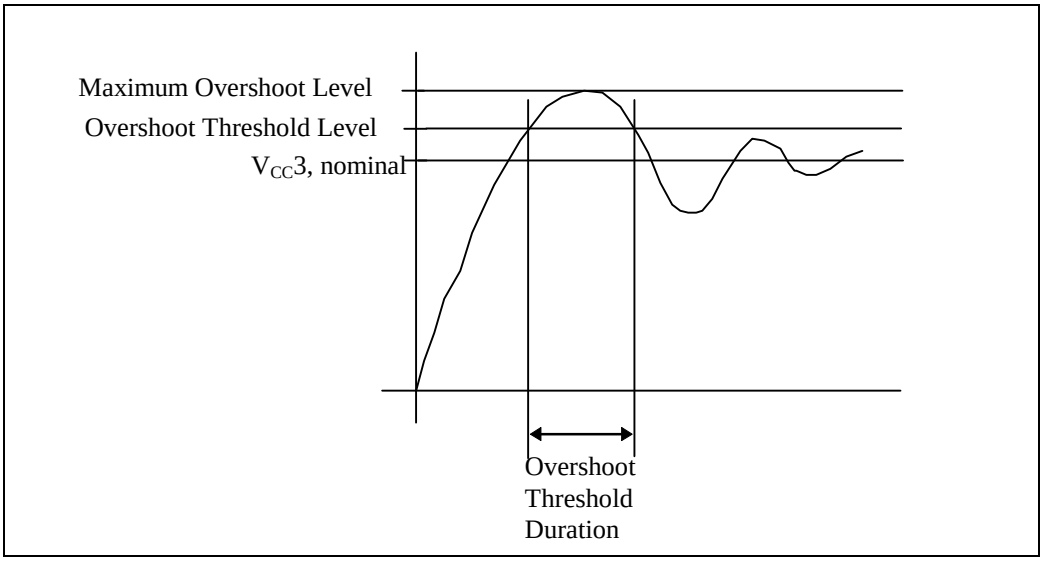


Figure 17-4. Maximum Overshoot Level, Overshoot Threshold Level, and Overshoot Threshold Duration

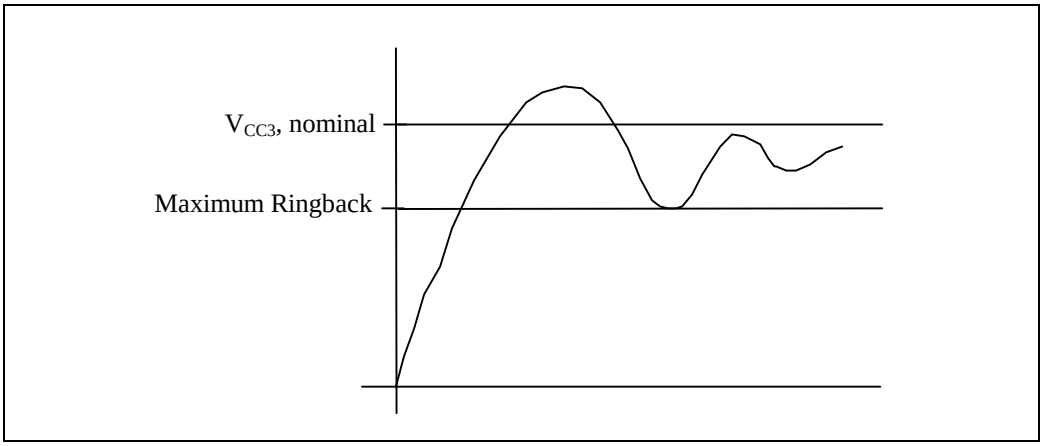


Figure 17-5. Maximum Ringback Associated with the Signal High State

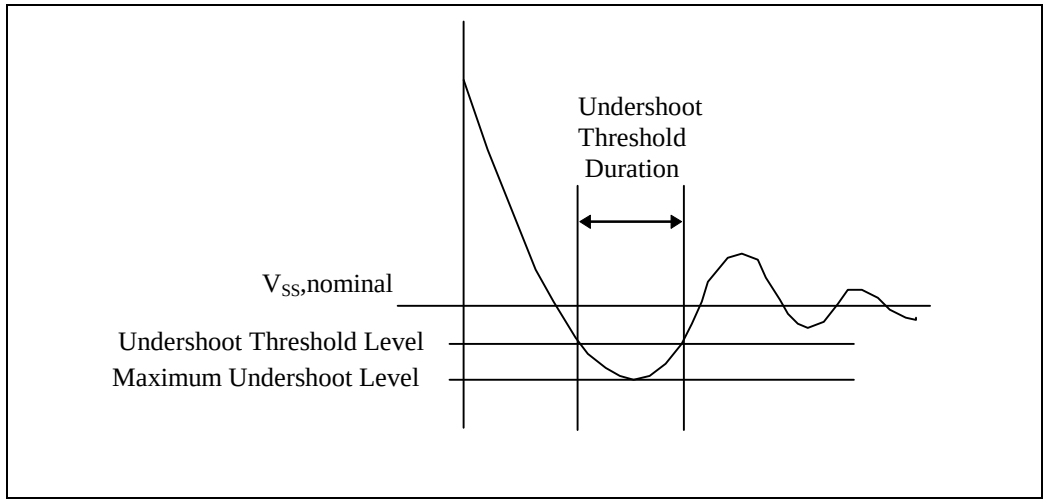


Figure 17-6. Maximum Undershoot Level, Undershoot Threshold Level, and Undershoot Threshold Duration

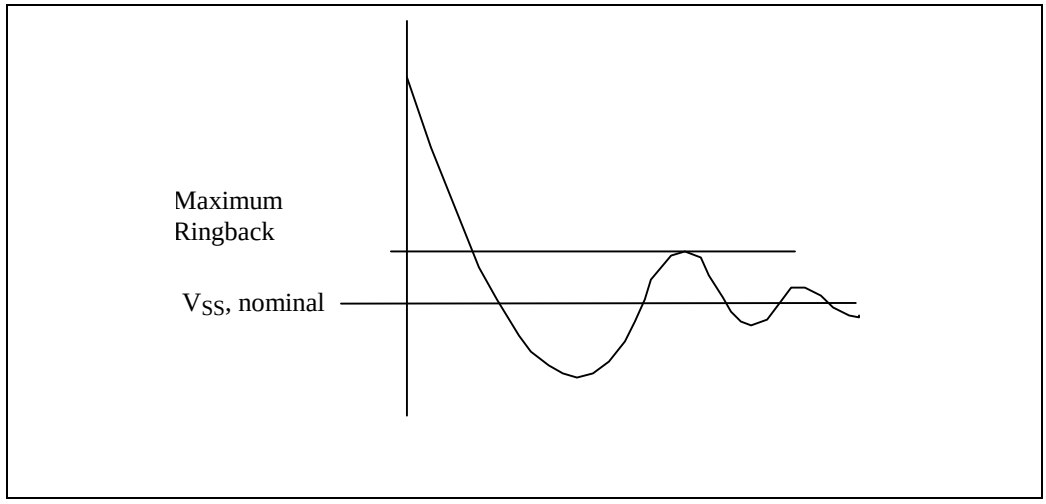


Figure 17-7. Maximum Ringback Associated with the Signal Low State

17.7. MECHANICAL SPECIFICATIONS

17.7.1. Socket 7 Mechanical Specifications

Although socket specifications differ among vendors, some parameters are required for all qualified sockets. New to Socket 7 is the addition of standardized heatsink clips. The motherboard layout should encompass keepout zones around these clips. Figure 17-8 indicates the socket clips and keep out zones. Refer to vendor datasheets for mechanical dimensions of individual sockets.

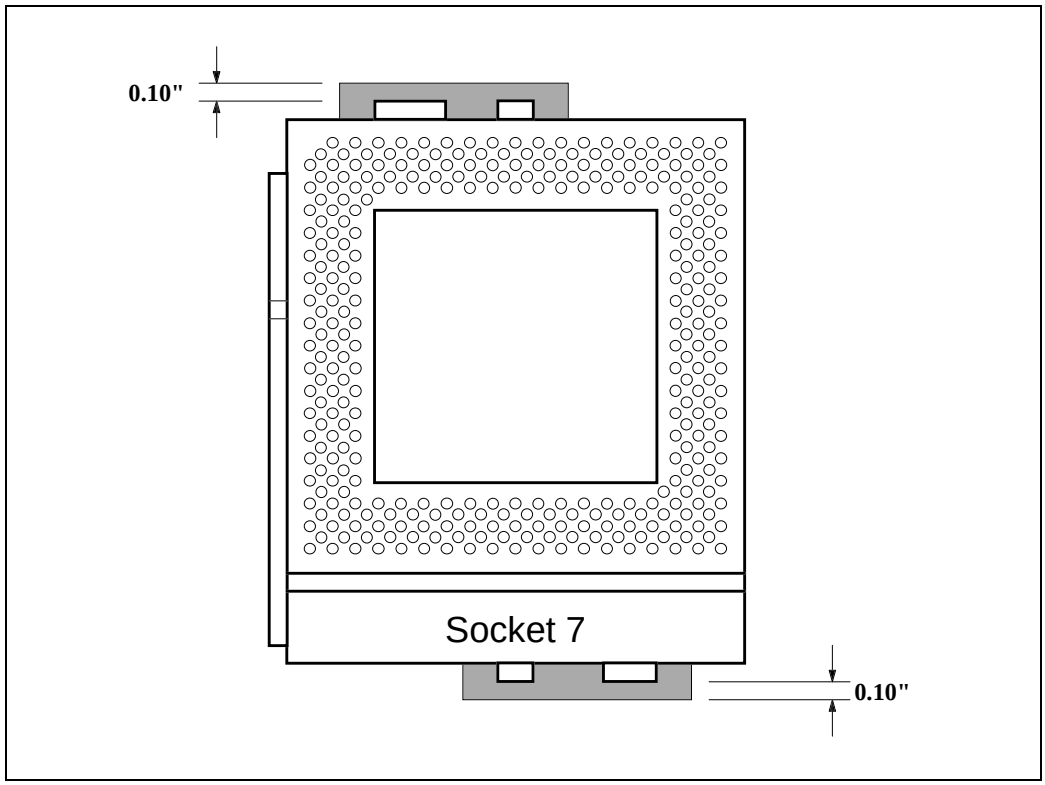


Figure 17-8. Socket 7 Heatsink Clip Tabs and Keepout Zones

17.7.2. Pentium® OverDrive® Processor Mechanical Specifications

The future Pentium OverDrive processor with MMX technology will be packaged in a 320-pin staggered pin grid array (SPGA). The pins will be arranged in a 37 x 37 matrix and the

package dimensions will be 1.95" x 1.95" (4.95cm x 4.95cm). The future Pentium OverDrive processor with MMX technology is shipped with an attached fan/heatsink.

Table 17-9. Future Pentium® OverDrive® Processor with MMX™ Technology Package Information Summary

	Package Type	Total Pins	Pin Array	Package Size
Future Pentium® OverDrive® processor with MMX™ technology	SPGA	320	37 x 37	1.95" x 1.95"

Table 17-10 and Figure 17-9 show the Socket 7 dimensions and the space requirements for the future Pentium OverDrive processor with MMX technology. The future Pentium OverDrive processor with MMX technology will have a fan/heatsink attached. For proper cooling, the fan/heatsink requires an additional space clearance.

Table 17-10. Socket 7 Requirements for SPGA Package Dimensions

Symbol	Millimeters		Inches		Notes
	Min	Max	Min	Max	
A		44.45		1.75	Includes Airspace (1)
A1	30.48	34.29	1.2	1.35	(1)
A2	2.62	3.73	0.103	0.147	
A3		30.48		1.200	(1)
A4	10.16		0.400		Air Space
A5		0.89		0.035	Lid Thickness
D	49.28	50.29	1.940	1.980	
D1	45.47	45.97	1.790	1.810	
L	3.05	3.30	0.120	0.130	
N	321		321		SPGA Pins (2)
S1	1.52	2.67	0.060	0.105	

NOTES:
1. These values are different from the Socket 5 specification.
2. The number of pins is increased from that of Socket 5.

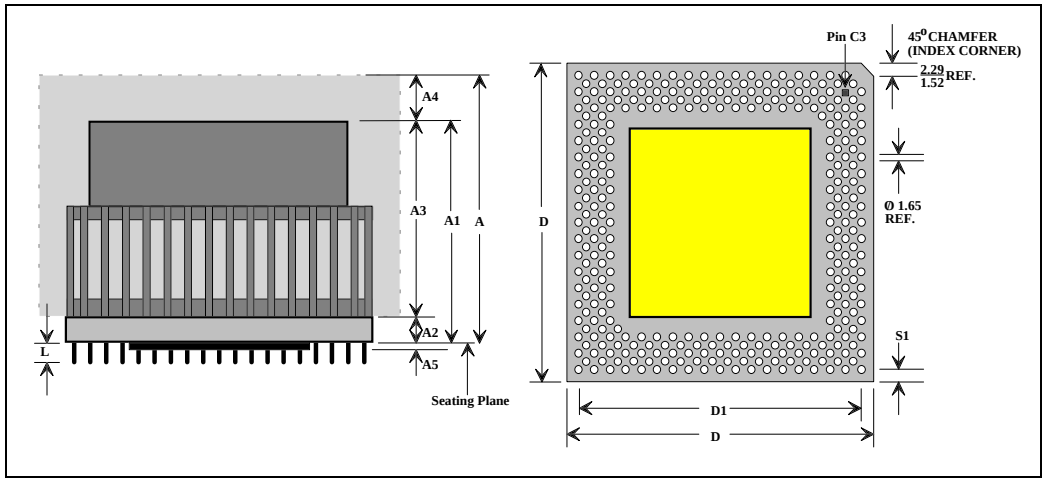


Figure 17-9. Future Pentium® OverDrive® Processor with MMX™ Technology Dimensions and Space Requirements

17.8. THERMAL SPECIFICATIONS

17.8.1. Thermal Information

The future Pentium OverDrive processors with MMX technology will be cooled with an integrated fan/heatsink cooling solution. The future Pentium OverDrive processors with MMX technology with an integrated fan/heatsink are specified for proper operation when T_A (air temperature entering the fan/heatsink) is a maximum of 45°C. When the $T_A(max) \leq 45°C$ specification is met, the fan/heatsink will keep T_C (case temperature) within the specified range, provided airflow through the fan/heatsink is unimpeded.

17.8.2. Thermal and Physical Space Requirements

The future Pentium OverDrive processors with MMX technology for Socket 7 will utilize an integrated fan/heatsink cooling solution. Intel's fan/heatsink cooling solution requires that the air temperature entering the fan/heatsink (T_A) does not exceed 45°C under worst case conditions.

A required height of 0.4" airspace is required above the fan/heatsink unit to ensure that the airflow through the fan/heatsink is not blocked. Blocking the airflow to the fan/heatsink reduces the cooling efficiency and decreases the fan lifetime. Figure 17-10 illustrates an acceptable airspace clearance above the fan/heatsink.

The fan/heatsink will reside within the boundaries of the surface of the chip (1.95” x 1.95”). There are also free airspace clearance requirements around the ceramic package to ensure that the airflow exiting the fan/heatsink is not blocked. Figure 17-10 details the minimum space needed around the chip package to ensure proper airflow through the fan/heatsink.

As shown in Figure 17-10, it is acceptable to allow any device (i.e. add-in cards, surface mount device, chassis etc.) to enter within the free space distance of 0.2” from the chip package if it is not taller than the level of the heatsink base. In other words, if a component is taller than height ‘B’, it cannot be closer to the chip package than distance ‘A’. This applies to all four sides of the chip package, although the ZIF socket lever cam area will generally automatically meet this specification since the width is larger than distance ‘A’ (0.2”).

To avoid localized heating at the processor, it is critical that adequate airflow be provided at the socket. A clear air path and adequate venting must be provided to prevent hot spots from occurring.

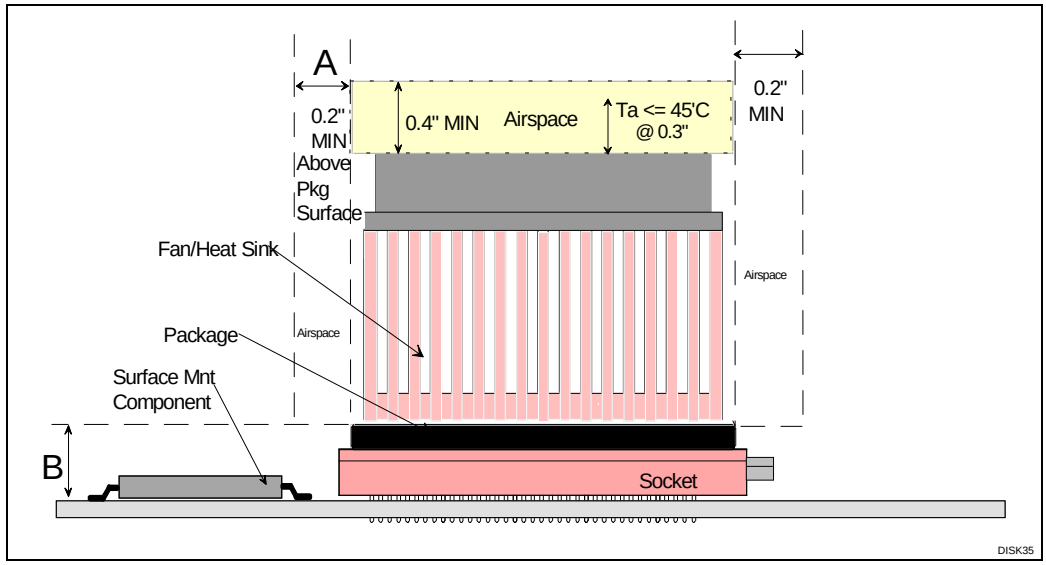


Figure 17-10. Thermal and Physical Space Requirements

17.8.2.1. PHYSICAL REQUIREMENTS

- The future Pentium OverDrive processor with MMX technology requires 1.75” vertical clearance above the surface (opposite pin side) of Socket 7 when installed.
- The future Pentium OverDrive processor with MMX technology requires 0.2” clearance around all four sides of the package.
- The future Pentium OverDrive processor with MMX technology requires space greater than specified above for end user installation.

17.8.2.2. THERMAL REQUIREMENTS

- The future Pentium OverDrive processor with MMX technology specifies a maximum air temperature entering the fan/heatsink of 45°C. T_A is measured where the air enters the fan/heatsink unit approximately 0.3” above the top of the fan/heatsink.

17.8.2.3. FAN/HEATSINK SUPPLY REQUIREMENTS

- The future Pentium OverDrive processor with MMX technology requires 5 volt power connections through the $V_{CC}5$ package pins.

18

Pentium® Processors for Mobile Systems

CHAPTER 18
PENTIUM® PROCESSORS FOR MOBILE SYSTEMS

18.1. INTRODUCTION

Intel manufactures a reduced power version of the latest Pentium processor, the Pentium with voltage reduction technology as well as the Mobile Pentium processor with MMX technology, targeting the mobile market. The Pentium processor for mobile systems is offered in the Tape Carrier Package (TCP) and the Staggered Pin Grid Array (SPGA) package. It has all the advanced features of the 3.3V Pentium processor.

TCP has several features which allow high-performance notebooks to be designed with the Pentium processor, including the following:

- TCP dimensions are ideal for small form-factor designs.
- TCP has superior thermal resistance characteristics.
- 2.45V or 2.9V core and 3.3V I/O buffer V_{CC} inputs reduce power consumption significantly, while maintaining 3.3V compatibility externally.
- The SL Enhanced feature set, which was initially implemented in the Intel386 CPU.

The architecture and internal features of the Mobile version of the Pentium processor are identical to the desktop processor specifications provided in this document, except several features not used in mobile applications which have been eliminated to streamline it for mobile applications.

18.2. DOCUMENT REFERENCE LIST

For information regarding Pentium processors for mobile systems, you may obtain documents listed in Table 18-1 by calling our toll-free literature distribution center at 1-800-548-4725.

Table 18-1. Document Reference List

Document Title	Document Type	Document Number
Mobile Pentium® Processor with MMX™ Technology	Data Sheet	243292
Pentium Processor with Voltage Reduction Technology Max. Operating Frequency: 75 Mhz, 100 MHz, 120 MHz, 133 MHz, 150 MHz	Data Sheet	242557
Pentium Processor at iCOMP® Index 815\100 Mhz Pentium Processor at iCOM Index 735\90 Mhz Pentium Processor at iCOMP® Index 610\75 Mhz with Voltage Reduction Technology	Data Sheet	242973
Pentium Processor at iCOMP Index 610\75	Data Sheet	242323
Pentium Processor (610\75) Design Considerations for Mobile Systems	Application Note	242417
Pentium Processor (610\75) Power Supply Considerations for Mobile Systems	Application Note	242415
Pentium Processor with Voltage Reduction Technology: Power Supply Design Considerations for Mobile Systems	Application Note	242558
Packaging Databook	Databook	240800

A

Errata and S-Specs for the Pentium® Processor Family

APPENDIX A
Errata and S-Specs for the Pentium® Processor Family

GENERAL INFORMATION

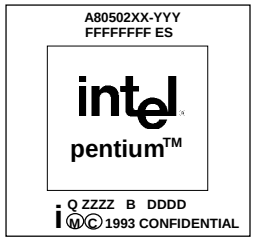
Nomenclature

S-Specs are exceptions to the published specifications, and apply only to the units assembled under that s-spec.

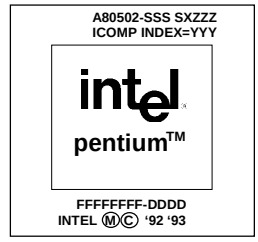
Errata are design defects or errors. Errata may cause the Pentium processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Top Markings

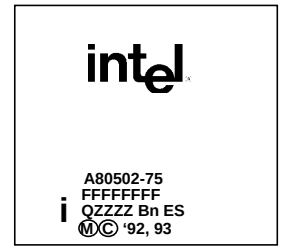
B-Step Engineering Samples:



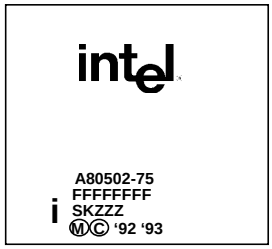
B-Step Production Units:



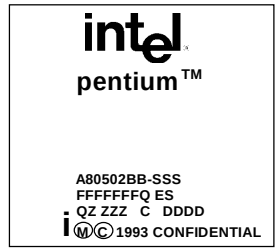
B-Step TCP Engineering Samples:



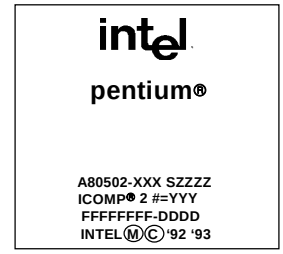
B-Step TCP Production Units:



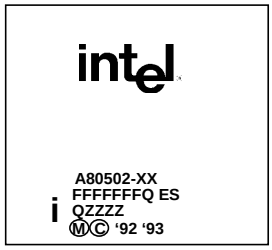
C-Step Engineering Samples:



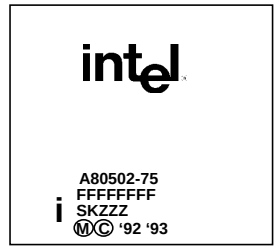
C and cB1-Step Production Units:



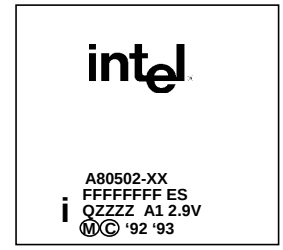
C-Step Engineering Sample TCP Units:



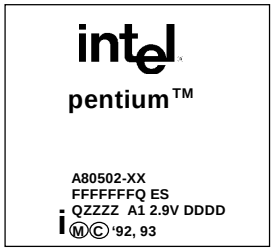
C-Step Production TCP Units:



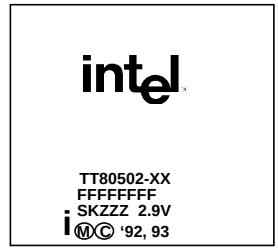
mA1-Step Engineering Samples TCP Units:



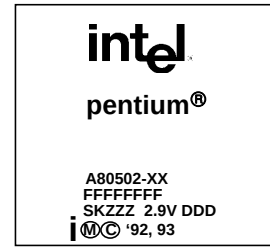
mA1-Step Engineering Sample SPGA Units:



mA1-Step Production TCP Units:



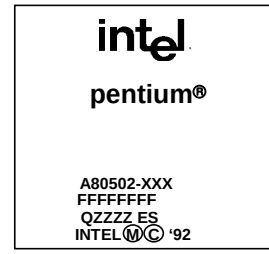
mA1-Step Production SPGA Units:



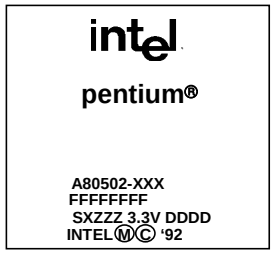
mcB1-Step Engineering Sample TCP Units:

mcB1-Step Production TCP Units:

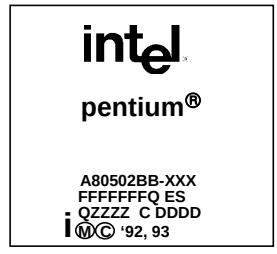
mcB1-Step Engineering Sample SPGA Units:



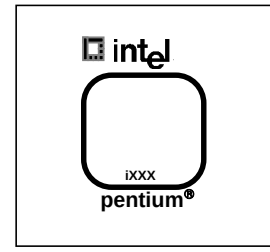
mcB1-Step Production SPGA Units:



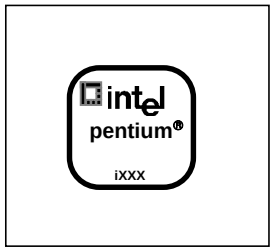
cC0-Step Engineering Sample Units:



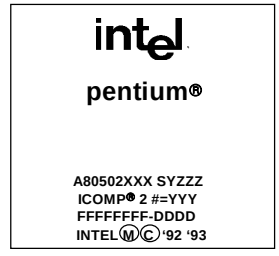
cC0-Step Engineering Sample PPGA Units:



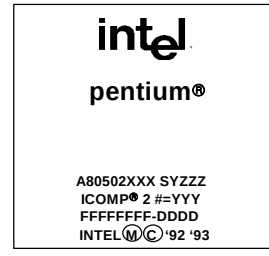
cC0-Step 200-MHz Engineering Sample PPGA Units:



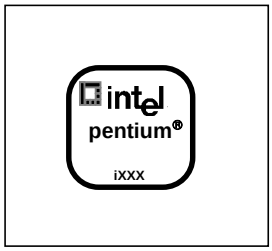
133 MHz cC0-Step Production Units:



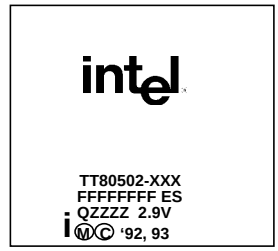
150 and 166-MHz cC0-Step Production Units:



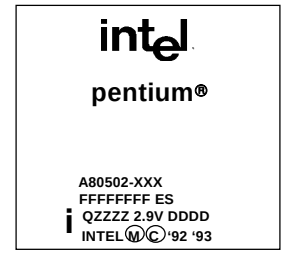
cC0-Step Production PPGA Units:



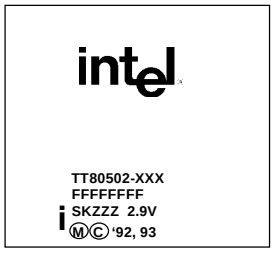
mA4-Step Engineering Sample TCP Units:



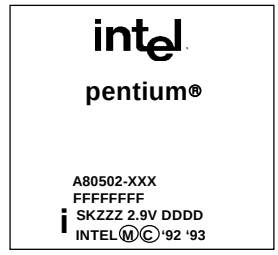
mA4-Step Engineering Sample SPGA Units:



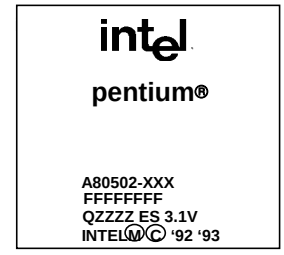
mA4-Step Production TCP Units:



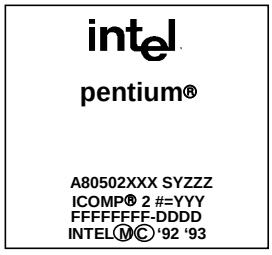
mA4-Step Production SPGA Units:



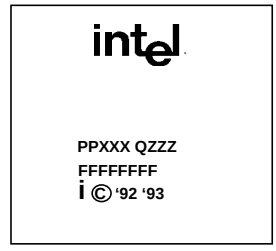
mcC0-Step Engineering Sample SPGA Units



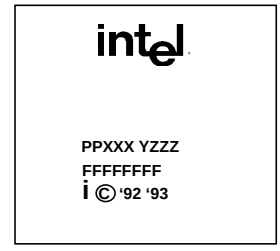
mcC0-Production SPGA Steppings:



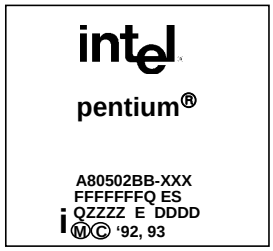
mcC0-Step Engineering Sample TCP



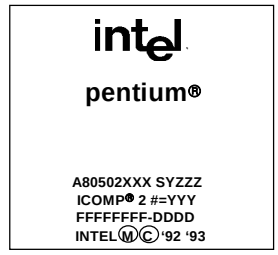
mcC0-Step Production TCP Units:



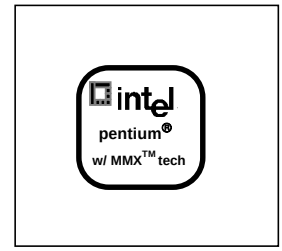
E0-Step Engineering Sample Units:



E0-Step Production Units:

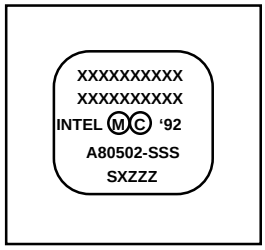


xA3-Step Pentium® processor w/ MMX™ technology Engineering Sample PPGA Units:

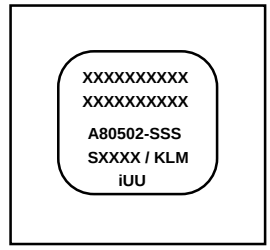


Bottom Markings

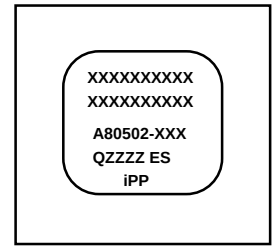
C-Step Production Units:
(before 7/95)



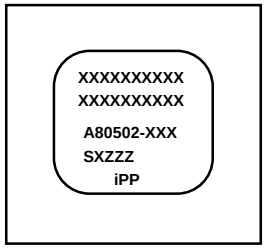
C and cB1-Step Production Units:
(after 7/95)



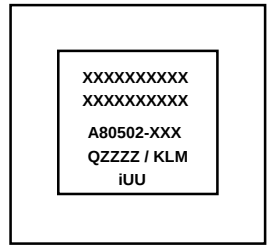
mcB1-Step Engineering Sample SPGA Units:



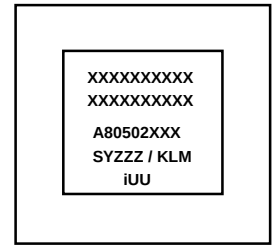
mcB1-Step Production SPGA Units:



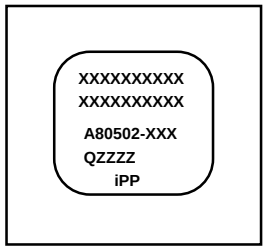
cC0-Step Engineering Sample Units:



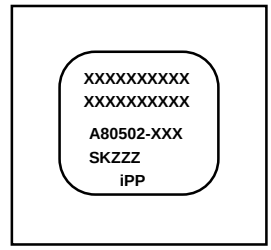
cC0-Step Production Units:



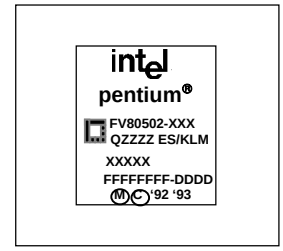
mA4-Step Engineering Sample SPGA Units:



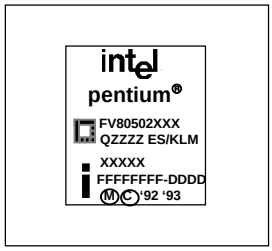
mA4-Step Production SPGA Units:



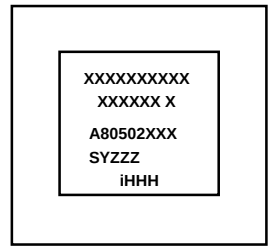
cC0-Step Engineering Sample PPGA Units:



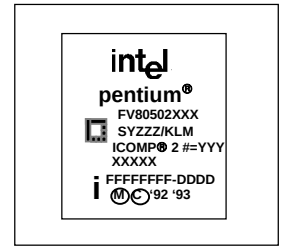
cC0-Step 200 MHz Engineering Sample PPGA Units:



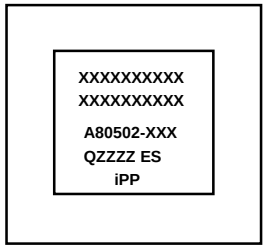
mcC0-Step Production SPGA Units:



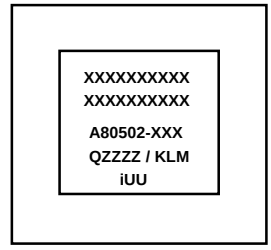
cC0-Step Production PPGA Units:



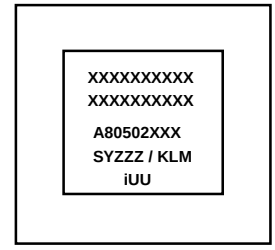
mcC0-Step Engineering Sample SPGA Units



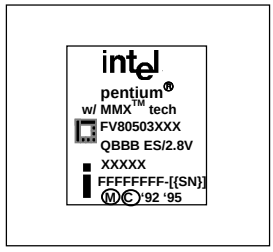
E0-Step Engineering Sample Units



E0-Step Production Units:

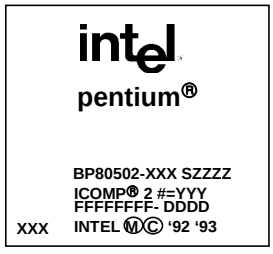


xA3-Step Pentium® processor w/ MMX™ technology Engineering Sample PPGA Units:

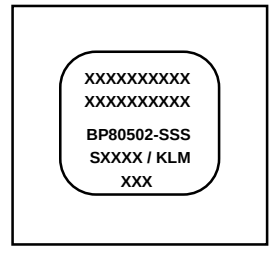


Boxed Pentium® Processors with Attached Fan Heatsink

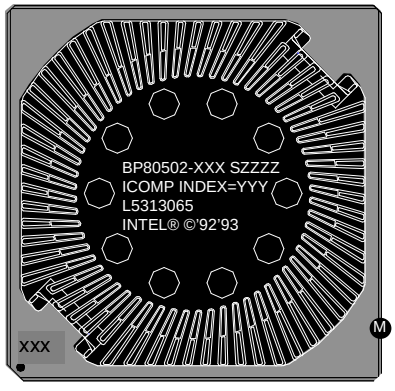
Top Side Marking:



Bottom Side Marking:



Fan Heatsink Base Marking:



NOTES:

- XX or XXX = Core Speed (MHz).
- BB = Bus speed (Mhz)
- SXZZZ/SYZZZ/SZZZZ = Product S-Spec number
- FFFFFFFF = FPO # (Test Lot Traceability #).
- For packages with heat spreaders, the inner line box defines the spreader edge.
- Ink Mark = All logo information on the heat spreader.
- Laser Mark = The two lines of information above and below the heat spreader. All bottomside information is laser mark.
- ES = Engineering Sample.
- QZZZZ = Sample Specification number.
- DDDD = Serialization code.
- YYY = : iCOMP® Index 2.0 OR iCOMP Index. Intel is making an enhancement to the current plastic PGA (PPGA) and ceramic PGA (CPGA) desktop and mobile Pentium® processors with the addition of the iCOMP Index 2.0 rating as part of the processor package mark. For PPGA Pentium processors, the iCOMP Index 2.0 will be marked on the bottom side (pin side) of the package and for CPGA it will be marked on the top side of the package. The part marking will be: iCOMP 2 # = XXX (67 for 75-MHz, 81 for 90-MHz, 90 for 100-MHz, 100 for 120-MHz, 111 for 133-MHz, 114 for 150-MHz, 127 for 166-MHz, and 142 for 200-MHz). Older parts may be marked with the iCOMP Index (610 for 75-MHz and 735 for 90-MHz, 815 for 100-MHz, 1000 for 120-MHz, 1110 for 133-MHz, 1176 for 150-MHz and 1308 for 166-MHz parts).
- TT = TCP Package, A = SPGA Package
- The bottom markings on the C and cB1-step production units will replace the existing bottom marking on C-step parts effective 7/95.
- UU = 75 or 133 for 75- or 133-MHz Pentium processors, PP for all other speeds and MPP for mobile Pentium processors
- K = V for VRE voltage range and S for standard voltage range
- L = M for min valid MD timings and S for min valid standard timings
- M = U is not tested for DP, is tested for UP and MP and S is tested for DP, UP and MP

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0	5	2	1	B1	75/50	Q0540	ES
2	5	2	1	B1	75/50	Q0541	ES
0	5	2	1	B1	90/60	Q0542	STD
0	5	2	1	B1	90/60	Q0613	VR
2	5	2	1	B1	90/60	Q0543	DP
0	5	2	1	B1	100/66	Q0563	STD
0	5	2	1	B1	100/66	Q0587	VR
0	5	2	1	B1	100/66	Q0614	VR
0	5	2	1	B1	75/50	Q0601	TCP Mobile
0	5	2	1	B1	90/60	SX879	STD
0	5	2	1	B1	90/60	SX885	MD
0	5	2	1	B1	90/60	SX909	VR
2	5	2	1	B1	90/60	SX874	DP, STD
0	5	2	1	B1	100/66	SX886	MD
0	5	2	1	B1	100/66	SX910	VR, MD
0	5	2	2	B3	90/60	Q0628	STD
0 or 2	5	2	2	B3	90/60	Q0611	STD
0 or 2	5	2	2	B3	90/60	Q0612	VR
0	5	2	2	B3	100/66	Q0677	VRE/MD
0	5	2	2	B3	75/50	Q0606	TCP Mobile
0	5	2	2	B3	75/50	SX951	TCP Mobile
0	5	2	2	B3	90/60	SX923	STD
0	5	2	2	B3	90/60	SX922	VR
0	5	2	2	B3	90/60	SX921	MD
2	5	2	2	B3	90/60	SX942	DP, STD
2	5	2	2	B3	90/60	SX943	DP, VR
2	5	2	2	B3	90/60	SX944	DP, MD
0	5	2	2	B3	90/60	SZ951[5]	STD

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information (Contd.)

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0	5	2	2	B3	100/66	SX960	VRE/MD
0 or 2	5	2	4	B5	75/50	Q0704	TCP Mobile
0 or 2	5	2	4	B5	75/50	Q0666	STD
0 or 2	5	2	4	B5	90/60	Q0653	STD
0 or 2	5	2	4	B5	90/60	Q0654	VR
0 or 2	5	2	4	B5	90/60	Q0655	MD
0 or 2	5	2	4	B5	100/66	Q0656	MD
0 or 2	5	2	4	B5	100/66	Q0657	VR, MD
0 or 2	5	2	4	B5	100/66	Q0658	VRE/MD
0	5	2	4	B5	120/60	Q0707	VRE/MD[1]
0	5	2	4	B5	120/60	Q0708	STD[1]
0	5	2	4	B5	75/50	SX975	TCP Mobile
0 or 2	5	2	4	B5	75/50	SX961	STD
0 or 2	5	2	4	B5	75/50	SZ977[5]	STD
0 or 2	5	2	4	B5	90/60	SX957	STD
0 or 2	5	2	4	B5	90/60	SX958	VR
0 or 2	5	2	4	B5	90/60	SX959	MD
0 or 2	5	2	4	B5	90/60	SZ978[5]	STD
0 or 2	5	2	4	B5	100/66	SX962	VRE/MD
0	5	2	5	C2	75/50	Q0725	TCP Mobile
0 or 2	5	2	5	C2	75/50	Q0700	STD
0 or 2	5	2	5	C2	75/50	Q0749	MD
0 or 2	5	2	5	C2	90/60	Q0699	STD
0 or 2	5	2	5	C2	100/50 or 66	Q0698	VRE/MD
0 or 2	5	2	5	C2	100/50 or 66	Q0697	STD
0	5	2	5	C2	120/60	Q0711	VRE/MD
0	5	2	5	C2	120/60	Q0732	VRE/MD
0	5	2	5	C2	133/66	Q0733	MD

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information (Contd.)

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0	5	2	5	C2	133/66	Q0751	MD
0	5	2	5	C2	133/66	Q0775	VRE/MD
0	5	2	5	C2	75/50	SK079	TCP Mobile
0 or 2	5	2	5	C2	75/50	SX969	STD
0 or 2	5	2	5	C2	75/50	SX998	MD
0 or 2	5	2	5	C2	75/50	SZ994[5]	STD
0 or 2	5	2	5	C2	75/50	SU070[6]	STD
0 or 2	5	2	5	C2	90/60	SX968	STD
0 or 2	5	2	5	C2	90/60	SZ995[5]	STD
0 or 2	5	2	5	C2	90/60	SU031[6]	STD
0 or 2	5	2	5	C2	100/50 or 66	SX970	VRE/MD
0 or 2	5	2	5	C2	100/50 or 66	SX963	STD
0 or 2	5	2	5	C2	100/50 or 66	SZ996[5]	STD
0 or 2	5	2	5	C2	100/50 or 66	SU032[6]	STD
0	5	2	5	C2	120/60	SK086	VRE/MD
0	5	2	5	C2	120/60	SX994	VRE/MD
0	5	2	5	C2	120/60	SU033[6]	VRE/MD
0	5	2	5	C2	133/66	SK098	MD
0	5	2	5	mA1[4]	75/50	Q0686	VRT[2], TCP
0	5	2	5	mA1[4]	75/50	Q0689	VRT[2], SPGA
0	5	2	5	mA1[4]	90/60	Q0694	VRT[2], TCP
0	5	2	5	mA1[4]	90/60	Q0695	VRT[2], SPGA
0	5	2	5	mA1[4]	75/50	SK089	VRT[2], TCP
0	5	2	5	mA1[4]	75/50	SK091	VRT[2], SPGA
0	5	2	5	mA1[4]	90/60	SK090	VRT[2], TCP
0	5	2	5	mA1[4]	90/60	SK092	VRT[2], SPGA
0 or 2	5	2	B	cB1[4]	120/60	Q0776	STD/no Kit[3]

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information (Contd.)

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0 or 2	5	2	B	cB1[4]	133/66	Q0772	STD/no Kit[3]
0 or 2	5	2	B	cB1[4]	133/66	Q0773	STD
0 or 2	5	2	B	cB1[4]	133/66	Q0774	VRE/MD, no Kit[3]
0 or 2	5	2	B	cB1[4]	120/60	SK110	STD/no Kit[3]
0 or 2	5	2	B	cB1[4]	133/66	SK106	STD/no Kit[3]
0 or 2	5	2	B	cB1[4]	133/66	S106J[7]	STD/no Kit[3]
0 or 2	5	2	B	cB1[4]	133/66	SK107	STD
0 or 2	5	2	B	cB1[4]	133/66	SU038[6]	STD/no Kit[3]
0	5	2	B	mcB1[4]	100/66	Q0884	VRT[2], TCP
0	5	2	B	mcB1[4]	120/60	Q0779	VRT[2], TCP
0	5	2	B	mcB1[4]	120/60	Q0808	3.3V, SPGA
0	5	2	B	mcB1[4]	100/66	SY029	VRT[2], TCP
0	5	2	B	mcB1[4]	120/60	SK113	VRT[2], TCP
0	5	2	B	mcB1[4]	120/60	SK118[7]	VRT[2], TCP
0	5	2	B	mcB1[4]	120/60	SX999	3.3V, SPGA
0 or 2	5	2	C	cC0	133/66	Q0843	STD/No Kit[3]
0 or 2	5	2	C	cC0	133/66	Q0844	STD
0 or 2	5	2	C	cC0	150/60	Q0835	STD
0 or 2	5	2	C	cC0	150/60	Q0878	STD, PPGA[9]
0 or 2	5	2	C	cC0	166/66	Q0836	VRE/No Kit[3]
0 or 2	5	2	C	cC0	166/66	Q0841	VRE
0 or 2	5	2	C	cC0	166/66	Q0886	VRE, PPGA[9]
0 or 2	5	2	C	cC0	166/66	Q0890	VRE, PPGA[9]
0	5	2	C	cC0	166/66	Q0949[8]	VRE, PPGA[9]
0 or 2	5	2	C	cC0	200/66	Q0951F[10]	VRE, PPGA[9]
0	5	2	C	cC0	200/66	Q0951[8]	VRE, PPGA[9]
0 or 2	5	2	C	cC0	120/60	SY062	STD
0 or 2	5	2	C	cC0	133/66	SY022	STD/ No Kit[3]

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information (Contd.)

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0 or 2	5	2	C	cC0	133/66	SY023	STD
0 or 2	5	2	C	cC0	133/66	SU073[6]	STD/ No Kit[3]
0 or 2	5	2	C	cC0	150/60	SY015	STD
0 or 2	5	2	C	cC0	150/60	SU071[6]	STD
0 or 2	5	2	C	cC0	166/66	SY016	VRE/ No Kit[3]
0 or 2	5	2	C	cC0	166/66	SY017	VRE
0 or 2	5	2	C	cC0	166/66	SU072[6]	VRE/ No Kit[3]
0	5	2	C	cC0	166/66	SY037[8]	VRE, PPGA[9]
0 or 2	5	2	C	cC0	200/66	SY044	VRE, PPGA[9]
0	5	2	C	cC0	200/66	SY045[8]	VRE, PPGA[9]
0	5	7	0	mA4[4]	75/50	Q0848	VRT[2], TCP
0	5	7	0	mA4[4]	75/50	Q0851	VRT[2], SPGA
0	5	7	0	mA4[4]	90/60	Q0849	VRT[2], TCP
0	5	7	0	mA4[4]	90/60	Q0852	VRT[2], SPGA
0	5	7	0	mA4[4]	100/66	Q0850	VRT[2], TCP
0	5	7	0	mA4[4]	100/66	Q0853	VRT[2], SPGA
0	5	7	0	mA4[4]	75/50	SK119	VRT[2], TCP
0	5	7	0	mA4[4]	75/50	SK122	VRT[2], SPGA
0	5	7	0	mA4[4]	90/60	SK120	VRT[2], TCP
0	5	7	0	mA4[4]	90/60	SK123	VRT[2], SPGA
0	5	7	0	mA4[4]	100/66	SK121	VRT[2], TCP
0	5	7	0	mA4[4]	100/66	SK124	VRT[2], SPGA
0	5	2	C	mcC0[4]	100/66	Q0887	TCP/VRT[2]
0	5	2	C	mcC0[4]	120/60	Q0879	TCP/VRT[2]
0	5	2	C	mcC0[4]	120/60	Q0880	SPGA 3.1V
0	5	2	C	mcC0[4]	133/66	Q0881	TCP/VRT[2]
0	5	2	C	mcC0[4]	133/66	Q0882	SPGA 3.1V
0	5	2	C	mcC0[4]	150/60	Q024	TCP/VRT[2]

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information (Contd.)

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0	5	2	C	mcC0[4]	150/60	Q0906	TCP 3.1V
0	5	2	C	mcC0[4]	150/60	Q040	SPGA/VRT[2]
0	5	2	C	mcC0[4]	75/50	SY056	TCP/VRT[2]
0	5	2	C	mcC0[4]	100/66	SY020	TCP/VRT[2]
0	5	2	C	mcC0[4]	100/66	SY046	SPGA 3.1V
0	5	2	C	mcC0[4]	120/60	SY021	TCP/VRT[2]
0	5	2	C	mcC0[4]	120/60	SY027	SPGA 3.1V
0	5	2	C	mcC0[4]	120/60	SY030	SPGA 3.3V
0	5	2	C	mcC0[4]	133/66	SY019	TCP/VRT[2]
0	5	2	C	mcC0[4]	133/66	SY028	SPGA 3.1V
0	5	2	C	mcC0[4]	150/60	SY061	TCP/VRT[2]
0	5	2	C	mcC0[4]	150/60	SY043	TCP 3.1V
0	5	2	C	mcC0[4]	150/60	SY058	SPGA/VRT[2]
0	5	2	6	E0	75/50	Q0846	TCP Mobile
0 or 2	5	2	6	E0	75/50	Q0837	STD
0 or 2	5	2	6	E0	90/60	Q0783	STD
0 or 2	5	2	6	E0	100/50 or 66	Q0784	STD
0 or 2	5	2	6	E0	120/60	Q0785	VRE
0	5	2	6	E0	75/50	SY009	TCP Mobile
0 or 2	5	2	6	E0	75/50	SY005	STD
0 or 2	5	2	6	E0	75/50	SU097[5]	STD
0 or 2	5	2	6	E0	75/50	SU098[6]	STD
0 or 2	5	2	6	E0	90/60	SY006	STD
0 or 2	5	2	6	E0	100/50 or 66	SY007	STD
0 or 2	5	2	6	E0	100/50 or 66	SU110[5]	STD
0 or 2	5	2	6	E0	100/50 or 66	SU099[6]	STD
0 or 2	5	2	6	E0	120/60	SY033	STD
0 or 2	5	2	6	E0	120/60	SU100[6]	STD

Basic 75/90/100/120/133/150/166/200-MHz Pentium® Processor and Pentium Processor with MMX™ Technology Identification Information (Contd.)

CPUID				Manufacturing Stepping	Speed (MHz) Core / Bus	S-Spec	Comments
Type	Family	Model	Stepping				
0 or 2	5	4	4	xA3	150/60	Q020	ES, PPGA[12]
0 or 2	5	4	4	xA3	166/66	Q019	ES, PPGA[12]
0 or 2	5	4	4	xA3	200/66	Q018	ES, PPGA[12]
0 or 2	5	4	4	xA3	166/66	SY059	PPGA[12]
0 or 2	5	4	4	xA3	166/66	SL239	SPGA[12]
0 or 2	5	4	4	xA3	200/66	SY060	PPGA[12]
0	5	4	4	mxA3	150/60	Q061	ES, PPGA[11]
0	5	4	4	mxA3	150/60	Q016	ES, TCP[11]
0	5	4	4	mxA3	166/66	Q017	ES, TCP[11]
0	5	4	4	mxA3	166/66	Q062	ES, PPGA[11]
0	5	4	4	mxA3	150/60	SL22G	TCP[11]
0	5	4	4	mxA3	150/60	SL246	PPGA[11]
0	5	4	4	mxA3	166/66	SL22F	TCP[11]
0	5	4	4	mxA3	166/66	SL23Z	PPGA[11]

NOTES:

- For a definition of STD, VR, VRE, MD, VRE/MD, refer to S-Spec 10 in this document. ES refers to Engineering Samples. DP indicates that this part can only be used as a dual processor. CPU Type of "2" or "0 or 2" indicates this part supports dual processing.
- The Type corresponds to bits [13:12] of the EDX register after RESET, bits [13:12] of the EAX register after the CPUID instruction is executed. This is shown as 2 different values based on the operation of the device as the primary processor or the dual processor upgrade.
- The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed.
- The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed.
- The Stepping corresponds to bits [3:0] of the EDX register after RESET, bits [3:0] of the EAX register after the CPUID instruction is executed.

1. T_{CASE} = 60°C.
2. VRT Intel's Voltage Reduction Technology: The V_{CC} for I/O is 3.3V, but the core V_{CC}, accounting for about 90% of power usage, is reduced to 2.9V, to reduce power consumption and heating.
3. No Kit means that part meets the specifications but is not tested to support 82498/82493 and 82497/82492 cache timings
4. STEPPING The cB1 stepping is logically equivalent to the C2-step, but on a different manufacturing process. The mcB1 step is logically equivalent to the cB1 step (except it does not support DP, APIC or FRC). The mcB1, mA1, mA4 and mcC0-steps also use Intel's VRT (Voltage Reduction Technology, see note 2 above) and are available in the

TCP and/or SPGA package, primarily to support mobile applications. The mxA3 is logically equivalent to the xA3 stepping (except it does not support DP or APIC). All mobile steppings are distinguished by an additional “m” prefix, for “mobile”. All steppings of the Pentium® processor with MMX™ technology are distinguished by an additional “x” prefix.

5. This is a boxed Pentium processor without the attached fan heatsink.
6. This is a boxed Pentium processor with an attached fan heatsink.
7. These parts do not support boundary scan. S106J was previously marked (and is the same as) SK106J.
8. DP, FRC and APIC features are not supported on these parts.
9. These parts are packaged in the Plastic Pin Grid Array (PPGA) package. For additional specifications of this package, see specification clarifications 27 and 28.
10. Some Q0951F units are marked on the bottom side with spec number Q0951 and with an additional line immediately underneath spelling out “Full Feature” to properly identify the unit.
11. This is a mobile Pentium processor with MMX technology with a core operating voltage of 2.285V - 2.665V.
12. This is a desktop Pentium processor with MMX technology with a core operating voltage of 2.7V-2.9V.

SUMMARY TABLE OF CHANGES

The following table indicates the Specification Changes, S-Specs, Errata, Specification Clarifications or Documentation Changes, which apply to the listed Pentium processor (75/90/100/120/133/150/166/200) and Pentium processor with MMX technology steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

CODES USED IN SUMMARY TABLE

X:	Erratum or S-Spec that applies to this stepping.
Fix:	This erratum or S-Spec is intended to be fixed in a future stepping of the component.
Fixed:	This erratum or S-Spec has been previously fixed.
NoFix	There are no plans to fix this erratum.
(No mark) or (Blank Box):	This erratum or S-Spec is fixed in listed stepping or does not apply to listed stepping.
DP:	Dual processing related errata.
AP:	APIC related errata.
TCP:	Applies to the listed stepping of a mobile Pentium processor in a TCP package only.

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	S-SPECS
1					X				X					Fixed	t_{6a}, t_{6b}, max valid delay A31-A3, BE7#-BE0#, ADS#, LOCK#
2					X				X					Fixed	Minimum required voltage separation between $V_{CC}3$ and $V_{CC}2$
3						X	X							Fixed	V_{IH} for TRST#
4							X							Fixed	V_{IL} for BF0 and BF1 is reduced
5										X				Fixed	Boundary scan timing changes
6										X				Fixed	SPGA $V_{CC}2$ supply voltage change
7					X				X					Fixed	AC specifications for the Pentium® processor with Voltage Reduction Technology
8							X			X				Fixed	Reduced V_{IL} for TCK
9	X	X	X	X		X								Fixed	Mixing steppings in dual processing mode
10	X	X	X	X		X		X						Fixed	MD/VR/VRE specifications
11				X										Fixed	120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0732, Q0733, Q0751, Q0775, SK086, SX994, SK098, SU033) do not support dual processing
12				X										Fixed	120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) do not support FRC
13				X										Fixed	120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) V_{CC} to CLK startup specification
14				X										Fixed	120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) current leakage on PICD1 pin
15													X	Fix	Mobile stop clock power
16												X	X	Fix	I_{IH}, input leakage current
17												X	X	Fix	Max valid delay A3-A31
18													X	Fix	Max valid delay ADS#
19													X	Fix	Max valid delay HITM#
20													X	Fix	Max valid delay data bus D0-D63
21												X		Fix	Desktop stop clock power

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	ERRATA
1	X	X	X											Fixed	Branch trace messages during lock cycles
2	X	X	X											Fixed	Breakpoint or single-step may be missed for one instruction following STI
3	X	X	X											Fixed	I/O restart does not function during single stepping or data breakpoint exceptions
4	X	X	X											Fixed	NMI or INIT in SMM with I/O restart during single-stepping
5	X	X	X											Fixed	SMI# and FLUSH# during shutdown
6	X	X	X											Fixed	No shutdown after IERR#
7	X	X	X											Fixed	FLUSH# with a breakpoint pending causes false DR6 values
8	X													Fixed	Processor core may not serialize on bus idle
9	X	X	X	X	X	X								Fixed	SMIACT# premature assertion during replacement writeback cycle
10	Superseded by a specification change														STPCLK# deassertion not recognized for 5 CLKs after BRDY# returned
11	X	X	X											Fixed	Future Pentium OverDrive® processor FERR# Contention in Two-Socket Systems
12	X													Fixed	Code cache lines are not invalidated if snooped during AutoHALT or stop grant states
13	X													Fixed	STPCLK# assertion during execution of the HALT instruction hangs system
14	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	NMI or INIT during HALT within SMM may cause large amount of bus activity
15	X	X	X	X	X	X	X	X	X	X	X			Fixed	RUNBIST restrictions when run through boundary scan circuitry
16	X	X	X	X		X		X			X			Fixed	FRC mode miscompare due to uninitialized internal register
17	Superseded by a specification change														STPCLK# restrictions during EWBE#
18	X	X	X											Fixed	Multiple allocations into branch target buffer
19	X	X	X											Fixed	100-MHz REP MOVS speed path
20	X	X	X											Fixed	Overflow undetected on some numbers on FIST
21	X	X	X											Fixed	Six operands result in unexpected FIST operation

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	ERRATA
22	X													Fixed	Snoop with table-walk violation may not invalidate snooped line
23	X	X												Fixed	Slight precision loss for floating-point divides on specific operand pairs
24	X	X	X											Fixed	FLUSH#, INIT or machine check dropped due to floating-point exception
25	X	X	X	X	X	X	X		X					Fixed	Floating-point operations may clear alignment check bit
26	X	X	X	X	X	X	X		X					Fixed	CMPXCHG8B across page boundary may cause invalid opcode exception
27	X	X	X									X	X	NoFix	Single-step debug exception breaks out of HALT
28	X	X	X	X	X	X	X	X	X	X	X			Fixed	Branch trace message corruption
29	X	X	X	X		X								Fixed	FRC lock-step failure during APIC write
30	X	X	X	X	X	X	X		X					Fixed	BE4#-BE0# sampled incorrectly at Min Vih
31	X	X	X	X		X								Fixed	Incorrect PCHK# output during boundary scan if in DP mode
32	X	X	X	X	X	X	X		X					Fixed	EIP altered after specific FP operations followed by MOV Sreg, Reg
33	X	X	X	X	X	X	X	X	X	X	X			Fixed	WRMSR into illegal MSR does not generate GP Fault
34	X	X	X											Fixed	Inconsistent data cache state from concurrent snoop and memory write
35	X	X	X											Fixed	BE3#-BE0# not driven during boundary scan if RESET high
36	X	X	X	X	X	X	X	X	X	X	X			Fixed	Incorrect FIP after RESET
37	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Second assertion of FLUSH# not ignored
38	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Segment limit violation by FPU operand may corrupt FPU state
39	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	FP exception inside SMM with pending NMI hangs system
40	X	X	X	X	X	X	X							Fixed	Current in Stop Clock state exceeds specification
41	X	X	X	X	X	X	X		X		X			Fixed	STPCLK# buffer samples incorrectly during boundary scan testing
42	X	X	X	X	X	X	X							Fixed	Incorrect decode of certain 0F instructions
43	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Data breakpoint deviations

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	ERRATA
44	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Event monitor counting discrepancies
45	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	VERR type instructions causing page fault task switch with T bit set may corrupt CS:EIP
46	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	BUSCHK# interrupt has wrong priority
47	X	X	X	X	X				X		X			Fixed	BF and CPUTYP buffers sample incorrectly during boundary scan testing
48	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Matched but disabled data breakpoint can be lost by STPCLK# assertion.
49	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	STPCLK# ignored in SMM when INIT or NMI pending
50	X	X	X	X	X	X	X	X	X	X	X			Fixed	STPCLK# pullup not engaged at RESET
51	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	A fault causing a page fault can cause an instruction to execute twice
52	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Machine check exception pending, then HLT, can cause skipped or incorrect instruction, or CPU hang
53	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	FBSTP stores BCD operand incorrectly If address wrap & FPU error both occur
54	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	V86 interrupt routine at illegal privilege level can cause spurious pushes to stack
55	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Corrupted HLT flag can cause skipped or incorrect instruction, or CPU hang
56	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Benign exceptions can erroneously cause double fault
57	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Double fault counter may not increment correctly
58					X		X		X	X				Fixed	Some input pins may float high when core V_{CC} powers up after I/O V_{CC} (mobile CPU)
59	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Short form of mov EAX/ AX/ AL may not pair
60	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Turning off paging may result in prefetch to random location
61	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	STPCLK# or FLUSH# after STI
62	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	REP string instruction not interruptable by STPCLK#
63	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Single step may not be reported on first instruction after FLUSH#

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	ERRATA
64	X	X	X	X		X		X			X	X	X	NoFix	Double fault may generate illegal bus cycle
65	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	TRST# not asynchronous
66	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	STPCLK# on RSM to HLT causes non-standard behavior
67	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Code cache dump may cause wrong IERR#
68	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Asserting TRST# pin or issuing JTAG instructions does not exit TAP Hi-Z state
69	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	ADS# may be delayed after HLDA deassertion
70	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Stack underflow in IRET gives #GP, not #SS
71	X	X	X	X	X	X	X	X	X	X	X	X	X	NoFix	Performance monitoring pins PM[1:0] may count the events incorrectly
72												X	X	Fix	BIST is disabled
73												X	X	NoFix	Branch trace messages may cause system hang
74												X	X	Fix	Enabling RDPMC in CR4 and also using SMM may cause shutdown
75												X	X	Fix	Event monitor counting discrepancies (fix)
76												X	X	NoFix	Event monitor counting discrepancies (Nofix)
77												X	X	Fix	INVD may leave valid entries in the cache due to snoop interaction
78												X	X	NoFix	TLB update is blocked after a specific sequence of events with a misaligned descriptor
1DP	X	X	X											Fixed	Problem with external snooping while two cycles are pending on the bus
2DP	X	X	X											Fixed	STPCLK# assertion and the stop grant bus cycle
3DP	X	X	X											Fixed	External snooping with AHOLD asserted may cause processor to hang
4DP	X	X	X											Fixed	Address parity check not supported in dual processing mode
5DP	X	X												Fixed	Inconsistent cache state may result from interprocessor pipelined READ into a WRITE

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	ERRATA
6DP	X	X	X											Fixed	Processors hang during Zero WS, pipelined bus cycles
7DP	X	X	X											Fixed	Bus lock-up problem in a specific dual processing mode sequence
8DP	X	X	X	X		X								Fixed	Incorrect assertion of PHITM# without PHIT#
9DP	X	X	X	X		X								Fixed	Double issuance of read cycles
10DP	X	X	X	X		X								Fixed	Line invalidation may occur on read or prefetch cycles
11DP	X	X	X	X		X		X			X			Fixed	EADS# or floating ADS# may cause extra invalidates
12DP	X	X	X	X		X								Fixed	HOLD and BOFF# during APIC cycle may cause dual processor arbitration problem
13DP	X	X	X	X		X								Fixed	System hang after hold during local APIC 2nd INTA cycle
14DP	X	X	X	X		X		X			X			Fixed	External snoop can be incorrectly invalidated
15DP	X	X	X	X		X		X			X	X		NoFix	STPCLK# re-assertion recognition constraint with DP
16DP	X	X	X	X		X		X			X	X		NoFix	Second assertion of FLUSH# during flush acknowledge cycle may cause hang
1AP	X	X	X											Fixed	Remote read message shows valid status after a checksum error
2AP	X	X	X											Fixed	Chance of clearing an unread error in the error register
3AP	X	X	X											Fixed	Writes to error register clears register
4AP	X	X	X											Fixed	Three interrupts of the same priority causes lost local interrupt
5AP	X	X	X											Fixed	APIC bus synchronization lost due to checksum error on a remote read message
6AP	X	X	X											Fixed	HOLD during a READ from local APIC register may cause incorrect PCHK#
7AP	X	X	X											Fixed	HOLD during an outstanding interprocessor pipelined APIC cycle hangs processor
8AP	X	X	X											Fixed	PICCLK reflection may cause an APIC checksum error
9AP	X	X	X	X		X		X			X			Fixed	Spurious interrupt in APIC through local mode

NO.	B1	B3	B5	C2	mA1	cB1	mcB1	cC0	mA4	mcC0	E0	xA3	mxA3	Plans	ERRATA
10AP	X	X	X											Fixed	Potential for lost interrupts while using APIC in through Local mode
11AP	X	X	X	X		X								Fixed	Back to back assertions of HOLD or BOFF# may cause lost APIC write cycle
12AP	X	X	X	X		X		X						Fixed	System hangs when BOFF# is asserted during second internal INTA cycle
13AP	X	X	X	X		X		X			X			Fixed	APIC pipeline cycle during cache linefill causes restarted cycle to lose its attribute
14AP	X	X	X	X		X		X			X	X		NoFix	INIT and SMI via the APIC three-wire bus may be lost
15AP								X			X			Fixed	IERR# in FRC lock-step mode during APIC write
16AP	X	X	X	X	X	X	X	X			X			Fixed	Inadvertent BRDY# during external INTA cycle with BOFF#
17AP	X	X	X	X		X		X			X			Fixed	APIC read cycle may not complete upon assertion of BOFF# and HOLD
18AP	X	X	X	X		X		X			X	X		NoFix	PICCLK must toggle for at least twenty cycles before RESET
19AP												X		Fix	APIC ID can not be changed
1TCP	X													Fixed	CPU may not reset correctly due to floating FRCMC# pin
2TCP	X			X	X		X		X	X	X			Fixed	BRDY# does not have buffer selection capability

S-SPECS

1. t_{6a}, t_{6c}, Max Valid Delay A31-A3, BE7#-BE0#, ADS#, LOCK#

Symbol	Parameter	Datasheet Max (nS)	S-Spec Max (nS)	Notes
t_{6a}	ADS#, BE0-7#	7.0	7.25	50 MHz, 60 MHz bus
t_{6c}	A31-A3, LOCK#	7.0	7.25	50 MHz, 60 MHz bus

2. Minimum Required Voltage Separation Between V_{CC3} and V_{CC2}

In order to ensure proper operation a minimum of 120mV separation must be maintained between $V_{CC}3$ (min) and $V_{CC}2$ (max) at all times while the processor is powered up. In order to provide flexibility in the power supply design, the voltage tolerance of +/- 165mV will be supported on both supplies, however, the voltage difference between the two supplies must remain greater than 120mV. Therefore, the capacitive decoupling scheme that handles current transients must be chosen to support this requirement. It is recommended that the OEM provide sufficient decoupling capacitance for the desired voltage tolerance distribution between $V_{CC}3$ and $V_{CC}2$, and perform actual system measurements to validate the design.

For further information, contact your local Intel Sales office.

$V_{CC}3$ (Min)	$V_{CC}2$ (Max)	$V_{CC}3$ (Min) - $V_{CC}2$ (Max) Standard	$V_{CC}3$ (Min) - $V_{CC}2$ (Max) S-Spec
3.3 - 0.165V	2.9 + 0.165V	70mV	120mV*

* Test condition is at $V_{CC}2$ = 2.9V + 0.165V = 3.065V: $V_{CC}3$ = 3.3V - 0.115V = 3.185V

3. V_{IH} For TRST#

Symbol	Pin	Standard Min	S-Spec Min	Unit
V_{IH}	TRST#	2.0	2.2	Volts

4. V_{IL} For BF0 and BF1 is Reduced

Symbol	Pin	Standard Max	S-Spec Max	Unit
V_{IL}	BF0, BF1	0.8	0.6	Volts

5. Boundary Scan Timing Changes

The boundary scan valid delay minimum time for t_{53} and t_{55} has been reduced for the mcC0 stepping as indicated below. This applies to both SPGA and TCP packages.

Symbol	Parameter	Standard Min Time (60/66MHz)	S-Spec Min Time (60/66MHz)
t_{53}	TDO Valid Delay	3.0 nS	2.8 nS
t_{55}	All Non-Test Outputs Valid Delay	3.0 nS	2.5 nS

6. SPGA $V_{CC}2$ Supply Voltage Change

The core supply voltage ($V_{CC}2$) required is changed from 2.9V to 3.1V. This applies to SPGA 100/120/133 MHz units only. I/O voltage supply ($V_{CC}3$) remains at 3.3V+/-165mV.

Symbol	Parameter	Standard Supply Voltage	S-Spec Supply Voltage
$V_{CC}2$	Core voltage supply	2.9V+/-165mV	3.1V+/-165mV

7. AC Specifications for the Pentium® Processor with Voltage Reduction Technology

The TCP and SPGA Pentium Processor with Voltage Reduction Technology AC specifications for 60- and 66-MHz bus operation have been published in the *Pentium® Processor with Voltage Reduction Technology* datasheets (Order Numbers 242973 and 242557). The mA1 and mA4 steppings differ from the published specifications as noted below.

SPGA AC Specifications for 60-MHz Bus

Symbol	Parameter	S-Specs 60MHz Bus (nS)		Mobile Standard 60MHz Bus (nS)	
		Min	Max	Min	Max
t_{6e}	A3-A31 Valid Delay	1.1	7.7	1.1	6.3
t_{12}	D0–D63, DP0–7 Write Data Valid Delay	1.3	7.8	1.3	7.5
t_{35}	D0–D63, DP0–7 Read Data Hold Time	2.0		1.5	

TCP AC Specifications for 60-MHz Bus

Symbol	Parameter	S-Specs 60MHz Bus (nS)		Mobile Standard 60MHz Bus (nS)	
		Min	Max	Min	Max
t_{6e}	A3-A31 Valid Delay	1.1	7.0	1.1	6.3
t_{35}	D0–D63, DP0–7 Read Data Hold Time	2.0		1.5	

SPGA AC Specifications for 66-MHz Bus

Symbol	Parameter	S-Specs 66MHz Bus (nS)		Mobile Standard 66MHz Bus (nS)	
		Min	Max	Min	Max
t_{6a}	BE0-7# Valid Delay	1.0	7.25	1.0	7.0
t_{6c}	LOCK# Valid Delay	1.1	7.25	1.1	7.0
t_{6d}	ADS# Valid Delay	1.0	7.0	1.0	6.0
t_{6e}	A3-A31 Valid Delay	1.1	7.5	1.1	6.3
t_{6f}	M/IO# Valid Delay	1.0	7.0	1.0	5.9
t_{9c}	HLDA Valid Delay	1.0	7.2	1.0	6.8
t_{10a}	HIT# Valid Delay	1.0	8.0	1.0	6.8
t_{12}	D0–D63, DP0–7 Write Data Valid Delay	1.3	7.8	1.3	7.5
t_{16b}	EADS# Setup Time	5.5		5.0	
t_{24b}	PEN# Setup Time	5.0		4.8	
t_{34}	D0–D63, DP0–7 Read Data Setup Time	3.0		2.8	
t_{35}	D0–D63, DP0–7 Read Data Hold Time	2.0		1.5	

TCP AC Specifications for 66-MHz Bus

Symbol	Parameter	S-Specs 66MHz Bus (nS)		Mobile Standard 66MHz Bus (nS)	
		Min	Max	Min	Max
t_{6c}	LOCK# Valid Delay	1.1	7.25	1.1	7.0
t_{6d}	ADS# Valid Delay	1.0	7.0	1.0	6.0
t_{6e}	A3-A31 Valid Delay	1.1	7.0	1.1	6.3
t_{6f}	M/IO# Valid Delay	1.0	6.8	1.0	5.9
t_{10a}	HIT# Valid Delay	1.0	8.0	1.0	6.8
t_{16b}	EADS# Setup Time	5.5		5.0	
t_{24b}	PEN# Setup Time	5.0		4.8	
t_{34}	D0–D63, DP0–7 Read Data Setup Time	3.0		2.8	
t_{35}	D0–D63, DP0–7 Read Data Hold Time	2.0		1.5	

8. Reduced V_{IL} For TCK

Symbol	Pin	Standard Min	S-Spec Min	Unit
V_{IL}	TCK	0.8	0.6	Volts

9. Mixing Steppings in Dual Processing Mode

Some OEMs may choose to ship their systems with one processor, and then perform a field upgrade and add a second processor dual processing system. In some cases, the two processors may not be of the exact same stepping. If there is a need to mix steppings of the Pentium processor in a dual processing system, the following guidelines must be met:

1. The processors must be set to run at the same frequencies, and the same bus/core fractions. For example: If the primary processor is running at 60 and 90 MHz, the dual processor must also run at 60 and 90 MHz.
2. The CPUTYP pin of the dual processor socket must be tied to V_{CC}.
3. Use the following table for restrictions, or workarounds required to mix the steppings. Each of the notes is the errata that may cause the system to fail. There may be other applicable errata, please see the errata descriptions for a full listing and the complete details of the workaround.

Mixing Stepping Matrix				
	B1 as Dual (CM Package)	**B3 as Dual (CM Package)**	**B5 as Dual**	**C2 as Dual**
B1 as Primary	5DP	5DP	5DP	5DP
B3 as Primary	5DP	5DP	5DP	5DP
B5 as Primary	5DP	5DP	1DP	1DP
C2 as Primary	5DP	5DP	1DP	No pipeline restrictions

NOTES:

5DP: Workaround requires pipelining disabled.

1DP: Workaround requires either pipelining disabled, or AHOLD pin held active one clock longer than BOFF# deassertion.

10. MD/VR/VRE Specifications

There are some changes to the standard V_{CC} and timing specifications to support the highest performance operation of the Pentium processor.

STD: The V_{CC} specification for the C2 and subsequent steppings of the Pentium processor is V_{CC} = 3.135V to 3.6V. The voltage range for B-step parts remains at 3.135V–3.465V. Note that all E0-step production parts are standard voltage.

VR: This is a reduced voltage specification that has the range of 3.300V–3.465V.

VRE/MD: These parts have a reduced and shifted voltage specification, and reductions in the minimum output valid delays on the list of pins in the table below. The VRE voltage range for the C2 and subsequent steppings of the Pentium processor is V_{CC} = 3.40-3.60V. The VRE voltage range for B-step parts remains at 3.45-3.60V.

MD: This is a reduction in the minimum valid timings on a subset of output pins. Due to faster operation of the core, and faster operation of the transistors at the higher voltages these minimum valid timings need to be met. These parts have the standard V_{CC} specification.

	Previous	**Current**
Operating V_{CC} Range (VRE)	3.45 to 3.60V	3.40 to 3.60V

There are no allowances for crossing the high and low limits of the voltage specification. Part operation beyond these ranges cannot be guaranteed. For more information on measurement techniques, see Chapter 7 of the *Pentium® Processor Family Developer's Manual* and the application note *Implementation Guidelines for 3.3V Pentium® Processors with VR/VRE Specifications.*

Symbol	Signal	Min Valid STD (50/60/66) Specifications	Min Valid, MD (50/60/66) Specifications	Units
t_{6c}	A3-16	1.1	0.5	nS
t_{6c}	A17-31	1.1	0.6	nS
t_{6a}	W/R#	1.0	0.8	nS
t_{6a}	M/IO#	1.0	0.8	nS
t_{6a}	D/C#	1.0	0.8	nS
t_{6c}	LOCK#	1.1	0.9	nS
t_{10b}	HITM#	1.1	0.7	nS
t_{6a}	BE0-7#	1.0	0.9	nS

11. 120-MHz and 133-MHz Parts (Q0707, Q0708, Q0711, Q0732, Q0733, Q0751, Q0775, SK086, SX994, SU033, SK098) Do Not Support Dual Processing

The 120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0732, Q0733, Q0751, Q0775, SK086, SX994, SU033, SK098) do not support dual processing as defined in the *Pentium® Processor Family Developer's Manual.* Dual processing support will be added in a future stepping.

12. 120-MHz and 133-MHz Parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) Do Not Support FRC

The 120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) do not support FRC as defined in *the Pentium® Processor Family Developer's Manual*

13. 120-MHz and 133-MHz Parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) V_{CC} to CLK Startup Specification

The specification for the maximum time from V_{CC} reaching nominal value to the time the CLK must toggle is 30 ms for 120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098). If this specification is not met it may impact the long term reliability of the component.

V_{CC} to CLK Startup Time: 30 ms maximum.

14. 120-MHz and 133-MHz Parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) Current Leakage on PICD1 Pin

The leakage current specification as described in the *Pentium® Processor Family Developer's Manual* is 200µA. The leakage specification for 120-MHz and 133-MHz parts (Q0707, Q0708, Q0711, Q0733, Q0751, Q0775, SK086, SK098) differs as follows:

Symbol	Parameter	Min	Max	Unit	Notes
I_{IH}	Input Leakage Current		250	µA	Vin = 0.4V (1)

NOTE: (1) This parameter is for input with pull up resistor.

15. Mobile Stop Clock Power

Parameter	Standard Power	S-SPEC Power
Mobile Maximum Stop Clock Power Dissipation	50 mW	150 mW
Mobile Typical Clock Power Dissipation (1)	20 mW	85 mW

NOTES:

1. Typical stop clock power dissipation is not tested.

16. I_{IH} Input Leakage Current

Symbol	Parameter	Standard Leakage Current	S-SPEC Leakage Current
I_{IH} (1), (2)	Input Leakage Current	200 µA	400 µA

NOTES:

1. This parameter is for input with pulldown.
2. Vin = 2.4 V.

17. Max Valid Delay A3-A31

Symbol	Parameter	Bus Frequecny	Standard Max Valid Delay	S-SPEC Max Valid Delay
Desktop t_{6e}	A3-A16	60, 66 MHz	6.3 nS	6.6 nS
Desktop t_{6e}	A17-A31	60, 66 MHz	6.3 nS	6.6 nS
Mobile t_{6e}	A3-A31	60 MHz	6.3 nS	7.0 nS
Mobile t_{6e}	A3-A31	66 MHz	6.3 nS	6.6 nS

18. Max Valid Delay ADS#

Symbol	Parameter	Bus Frequecny	Standard Max Valid Delay	S-SPEC Max Valid Delay
Mobile t_{6d}	ADS#	60 MHz	7.0 nS	7.0 nS (1)
Mobile t_{6d}	ADS#	66 MHz	6.0 nS	6.4 nS

1. This is for reference only, not an S-Spec.

19. Max Valid Delay HITM#

Symbol	Parameter	Bus Frequecny	Standard Max Valid Delay	S-SPEC Max Valid Delay
Mobile t_{10b}	HITM#	60 MHz	6.0 nS	6.7 nS
Mobile t_{10b}	HITM#	66 MHz	6.0 nS	6.4 nS

20. Max Valid Delay Data Bus D0-D63

Symbol	Parameter	Bus Frequecny	Standard Max Valid Delay	S-SPEC Max Valid Delay
Mobile t_{12}	D0-D63	60 MHz	7.5 nS	8.3 nS
Mobile t_{12}	D0-D63	66 MHz	7.5 nS	8.0 nS

21. Desktop Stop Clock Power

Parameter	Standard Power	S-SPEC Power
Desktop Maximum Stop Clock Power Dissipation	265 mW	375 mW
Desktop Typical Clock Power Dissipation	30 mW	160 mW

ERRATA

1. Branch Trace Message During Lock Cycles

PROBLEM: During instruction execution tracing only two Branch Trace messages can be buffered. If there is a possibility of a third message being delivered from the instruction being executed, the machine will stall to avoid overwriting either of the messages that are buffered. If this instruction is a "locked read-modify-write" operation, the processor will hang up due to internal service contention for the bus controller logic.

IMPLICATION: This problem only effects operation of the component while performing instruction tracing on the CPU. It has not been seen using Intel development tools.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

2. Breakpoint or Single-Step May Be Missed for One Instruction Following STI

PROBLEM: If the following conditions are met, the processor may shut off the interrupt window for one instruction following STI:

1. The address of the instruction preceding the STI instruction is a hit in the BTB.
2. The target address of the BTB hit does not correspond to the instruction following the STI instruction.

This will prevent breakpoints, single-step or other external interrupts from being recognized during this time.

IMPLICATION: The processor may not recognize NMI, SMI# INIT, FLUSH#, BUSCHK#, R/S#, code/data breakpoint and single-step for one instruction after executing STI. This is not a problem unless breakpoints or single-stepping is used and then the only effect is that the breakpoint would be missed.

WORKAROUND: Do not set a breakpoint on the next sequential instruction after STI, or disable branch prediction to prevent this problem.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

3. I/O Restart Does Not Function During Single Stepping or Data Breakpoint Exceptions

PROBLEM: If an SMI# interrupt is generated while a data breakpoint exception is pending or during single-stepping, an I/O restart attempt will not be successful.

IMPLICATION: If this problem occurs, it will not be possible to restart the I/O instruction.

WORKAROUND: Do not allow single-stepping or data breakpoint exceptions when attempting to restart an I/O instruction.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

4. NMI or INIT in SMM with I/O Restart During Single Stepping

PROBLEM: An NMI# or INIT may be falsely accepted while in an SMM handler if *all* of the following conditions are met:

1. I/O restart is enabled with the ITR bit in TR12.
2. An SS or DBP exception is pending at the time that the SMI# is recognized.
3. NMI# or INIT is asserted before another fault occurs or before an INTR occurs (and the IF flag is set).

IMPLICATION: If the above conditions are met, the processor may falsely go into an interrupt service routine or begin the INIT sequence.

WORKAROUND: Do not enable I/O trap restart while single-stepping.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

5. SMI# and FLUSH# During Shutdown

PROBLEM: If the processor transitions from the shutdown state to System Management mode via an SMI# interrupt, and FLUSH# is asserted after the SMI# interrupt is recognized, the processor returns to the shutdown state rather than to the SMM handler.

IMPLICATION: After the FLUSH# is asserted, the processor erroneously returns to the shutdown state when it should return to the SMM handler.

WORKAROUND: Do not assert SMI# while the processor is in the shutdown state.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

6. No Shutdown After IERR#

PROBLEM: If an internal parity error is reported to the IERR# pin and a mispredicted branch or a trap with higher priority than shutdown occurs, then the processor may not shutdown.

IMPLICATION: During the reporting of an internal parity error, the IERR# pin may go active without a processor shutdown.

WORKAROUND: When the IERR# pin is asserted, force the system to shutdown.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

7. FLUSH# with a Breakpoint Pending Causes False DR6 Values

PROBLEM: If I/O restart is enabled during single-stepping or while a breakpoint is pending, and a FLUSH# is asserted, the wrong value will be stored in DR6.

IMPLICATION: The debug status register (DR6) may contain false information.

WORKAROUND: Do not assert FLUSH# when I/O restart is enabled with single-stepping or a breakpoint is pending.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

8. Processor Core May Not Serialize on Bus Idle

PROBLEM: Under rare circumstances, a BOFF# asserted to the processor, while it is internally waiting (looping) for the completion of all pending bus cycles, may cause the processor to proceed with the event before all pending bus activity is completed. However, bus cycle ordering will not change.

The following is a list of events identified that may possibly effect system operation:

- SMI# pending
- OUT instructions
- Serializing instructions
- Stop Grant special cycle
- AutoHALT special cycle
- INIT asserted while there is a memory mapped APIC register access

SMI# pending If BOFF# is used to back off a bus cycle while an SMI# is pending, the processor may assert SMIACT# before re-starting the aborted bus cycles.

Serializing instruction If BOFF# is used to back off a bus cycle due to a serializing instruction, the processor may start executing the next instruction before restarting or completing the previous bus cycle. The processor, however, will not reorder any bus cycles for the new instruction in front of bus cycles for the previous instruction.

Invalidation during cache line fill
If BOFF# is used to back off a cache line fill and BOFF# occurs after the data has been returned to the processor but before the end of the line fill, an invalidation request during this time may result in the cache invalidation to occur before the line fill has completed, This may cause the cache line to remain in a valid state after the invalidation has completed. Note that if the invalidation request comes in via WBINVD or FLUSH#, the line fill would have to be backed off at least twice (or once for INVD) in order for the cache line to remain in a valid state after the invalidation has completed.

OUT instruction If BOFF# is used to back off a bus cycle due to an OUT instruction, the processor may start executing the next instruction before the bus cycle due to OUT has completed. (NOTE: the OUT instruction is similar to the serializing instructions except that it does not stop the prefetch of the subsequent instruction.) The processor, however, will not reorder any bus cycles for the new instruction in front of the OUT bus cycle.

Stop Grant special cycle
If BOFF# is used to back off a Stop Grant special cycle, the processor may hang.

AutoHALT special cycle
If BOFF# is used to back off a AutoHALT special cycle, the processor may hang.

INIT asserted while there is a memory mapped APIC register access
If BOFF# is used to back off a memory mapped APIC register access while an INIT is pending, the processor may hang. The access could be either a read or a write, and is an access to the local APIC register space.

IMPLICATION: This problem has only been observed in internal test vehicles. The six events have different possible implications.

SMI# pending The processor may enter SMM before restarting the aborted bus cycle. The SMIACT# assertion may cause the restarted bus cycle to run to SMRAM space.

Serializing Instruction Since the cycles are not reordered, a system should not encounter any problems unless it depends on the serializing instruction causing an external event prior to execution of the next instruction.

Invalidation during cache line fill

In a rare instance, a cache line may remain in the valid (E or S) state after the cache invalidation has completed.

OUT instruction Since the cycles are not reordered, a system should not encounter any problems unless it depends on the OUT instruction causing an external event prior to execution of the next instruction. For example, an OUT instruction may be used to assert the A20M# signal prior to the next instruction. In this case, observed code has followed the OUT with an I/O read (IN) to ensure the signal is properly asserted. A second case, could be using an OUT instruction to configure/initialize and interrupt controller and follow it with STI to enable interrupts. Once again no failure would be observed. The controller would respond with the spurious interrupt vector.

Stop Grant special cycle

If BOFF# is used to back off a Stop Grant special cycle, the processor may hang. Stop Grant and Stop Clock states for low power consumption cannot be used.

AutoHALT special cycle

If BOFF# is used to back off a AutoHALT special cycle, the processor may hang. This means that the lower power AutoHALT state is not usable. This does not affect the normal HALT state, entered with the HLT instruction though.

INIT asserted while there is a memory mapped APIC register access

If BOFF# is used to back off a memory mapped APIC register access (read or write) while an INIT is pending, the processor may hang. The INIT would only be used during a switch from Protected to Real mode, which is normally associated with a system reboot. In the processor lockup occurs a reboot may have to be performed via system powerdown.

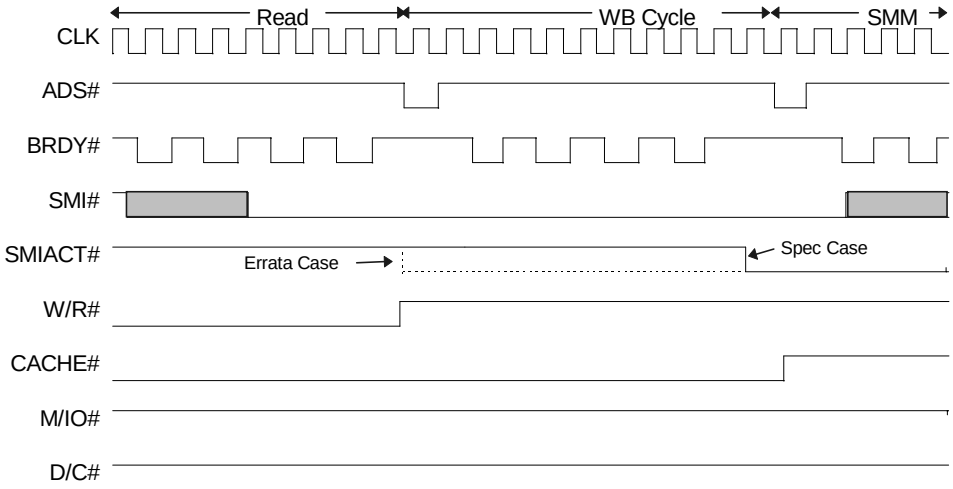
WORKAROUND: Restrict the use of BOFF# for the described events. In addition, the SMI# pending event can be eliminated by locating SMRAM so that it does not shadow standard memory and does not require SMIACT# for memory decode. The OUT or Serializing instruction events are also eliminated if the next instruction does not depend on the result of the event before executing the instruction. The Stop Grant special cycle event is also eliminated by not asserting STPCLK#. The AutoHALT special cycle event is also eliminated by disabling AUTOHALT (set TR12.AHD bit to '1').

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

9. SMIACT# Premature Assertion During Replacement Writeback Cycle

PROBLEM: If a data read cycle triggers a replacement writeback cycle and the SMI# signal is asserted before the last BRDY# of the read cycle, the processor may assert the SMIACT# signal prematurely. It occurs on all steppings when the processor is in 2/3 bus mode. **If the processor is in 1/2 bus mode it will not exhibit this erratum for C2 and subsequent steppings.**

Before the processor asserts SMIACT# in response to an SMI# request, it should complete all pending write cycles (including emptying the write buffers). However, if the appropriate conditions occur, the SMIACT# signal may get asserted during the replacement writeback cycle, at anytime during the writeback cycle. See below diagram.



IMPLICATION: If the SMIACT# signal is used by the system logic to decode SMRAM (e.g., SMRAM is located in an area that is overlaid on top of normal memory space, e.g. system, video etc.), then the replacement writeback cycle with SMIACT# asserted could occur to SMM space. Systems that locate SMRAM in its own distinct memory space (non-overlaid) should not experience data corruption once the SMRAM has been initialized and relocated.

Some board designs and/or chip sets may contain logic which locks when an SMIACT# is asserted during the writeback cycle. This is logic dependent and not all systems will fail when this condition occurs, although data corruption could still result.

WORKAROUND: Use one of following:

Non Overlaid SMRAM

In systems that relocate the SMBASE so that it does not overlay normal memory space.

1. Do not use SMIACT# as a decode signal once SMRAM has been relocated. When entering SMM for the first time to relocate the SMBASE to the non-overlaid region the system must be in one of three modes: the processor L1 cache must be in WT mode, the default SMBASE location (30000H) should be marked non-cacheable, or the L1 cache should be turned off. Once the SMBASE has been relocated to the non-overlaid region and SMIACT# is no longer used to decode SMRAM, the processor's L1 cache may be used in WB mode or the memory area at 30000H may be configured as a cacheable region.

Overlaid SMRAM

In systems that locate SMRAM over normal cacheable or non-cacheable memory space there are several software and hardware workarounds.

1. Operate the processor with the L1 cache in WT mode only. This will eliminate all processor WB cycles and thus prevent the error condition from occurring. The performance impact to the system for this could be between 5-15% depending on L2 cache size, speed, and operating mode (WB/WT).
2. Assert FLUSH# one clock before SMI# or assert FLUSH# and SMI# simultaneously and synchronously with the processor clock. The processor always assigns FLUSH# a higher priority than

SMI# and thus it will delay the assertion of SMIACT#. This is already a necessary requirement to maintain cache coherency when SMRAM overlays a cacheable normal memory area. Depending on the chip set, external hardware may be required to synchronize SMI# and FLUSH#. The performance impact of this solution depends heavily on frequency of SMIs. No performance impact should be visible if SMM is accessed infrequently, such as during periods of bus inactivity. Frequent SMM access (i.e. 18/sec) will result in less than 1% processor performance decrease.

Note that other hardware workarounds may be possible. Any other hardware workaround needs to take into account the possible case when BOFF# or AHOLD is asserted thereby restarting the writeback cycle after SMIACT# may have been asserted.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

10. STPCLK# Deassertion Not Recognized for 5 CLKs After BRDY# Returned

This erratum has been superseded by a specification change.

11. Future Pentium® OverDrive® Processor FERR# Contention in Two-Socket Systems

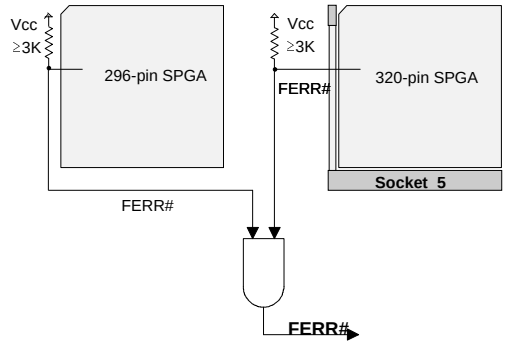
PROBLEM: When the future Pentium OverDrive processor is plugged into Socket 5 of a two socket 75-, 90-, or 100-MHz Pentium processor system, the OEM processor shuts down following RESET to allow the OverDrive processor to drive the bus. In this case, the FERR# output of the 75-, 90-, and 100-MHz Pentium processor continues to be driven HIGH (inactive).

IMPLICATION: If the system uses the FERR# output of the OEM processor, and has the signal connected together between the two sockets (296-pin SPGA OEM socket and 320-pin Socket 5), contention on this signal is certain since the future Pentium OverDrive processor, when placed into Socket 5, will also drive this output. This signal contention can cause component and even board reliability issues.

WORKAROUND: There are two possible workarounds for this erratum.

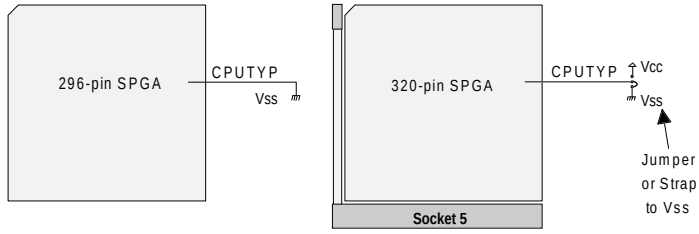
1. **For upgradability in two socket systems:** External logic may be used to connect the FERR# outputs from the two sockets (296-pin SPGA and 320-pin Socket 5) in a way that will resolve the signal contention.

   An external AND gate may be used to combine the outputs of the FERR# signals from the two sockets. A pullup resistor (≥3KΩ) is required on the FERR# output from both of the sockets in order to allow proper operation of both the dual processor and the future Pentium OverDrive processor. See the following figure:



2. **Upgradability by modifying a two socket system to be a single socket system:** This workaround involves modifying a design that includes two socket sites (296-pin SPGA and 320-pin Socket 5) such that it effectively becomes a single socket design.

   A dual processing two socket system must have CPUTYP tied to V_{SS} on the 296-pin SPGA OEM socket, and tied to V_{CC} on Socket 5 (320-pin SPGA). This workaround includes inserting a jumper on the Socket 5 CPUTYP signal (or strapping Socket 5 CPUTYP directly to V_{SS}) to make this the primary processor site. The system would then effectively become a single-socket design. NOTE: If a jumper is used, it must be set by the OEM prior to system sale (not by the end-user at the time of the future Pentium OverDrive processor purchase and installation). This jumper would set the CPUTYP signal on Socket 5 to V_{SS}. If the same board design is used for dual processing, this jumper (or strapping option) may be set to V_{CC} for those systems, as shown in the following figure:

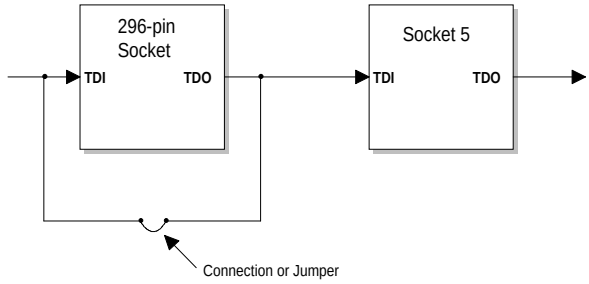


By setting CPUTYP to V_{SS}, and inserting the 75-, 90-, and 100-MHz Pentium processor into Socket 5 prior to system sale, the design can be treated as if it is a single socket system. When upgrading with the future Pentium OverDrive processor, the 75-, 90-, and 100-MHz Pentium processor is removed from Socket 5 and replaced by the OverDrive processor upgrade.

IMPLICATIONS OF WORKAROUND #2:

Other implications of this workaround include Boundary Scan, and any other signals not connected together between Socket 5 and the 296-pin SPGA socket site.

If Socket 5 follows the 296-pin socket in the Boundary Scan chain, the TDI input and TDO output of the 296-pin socket site must be connected together by the OEM prior to system sale in order to skip this socket site and complete the path to socket, as shown in the following figure. This connection is necessary only if Boundary Scan will be used by end-users after system sale.



All of the signals which are not connected together in a dual socket system must be handled by both socket sites if the feature is desired. These signals are APCHK#, BP[3:0], IERR#, PM[1:0], PRDY, and R/S#.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

12. Code Cache Lines Are Not Invalidated if Snooped During AutoHALT or Stop Grant States

PROBLEM: If the code cache is snooped while the processor is in the Stop Grant state or the AutoHALT Powerdown state, and there is a hit to a line in the code cache with the INV pin asserted, the line may not be invalidated. In normal operation, a hit to a line in the code cache results in that line being invalidated.

IMPLICATION: This problem will cause the snooped line to remain valid in the cache. This may cause the processor to execute an invalid instruction that erroneously remained valid in the code cache. NOTE: HIT# is properly asserted. This may occur in DMA transfers, or transfers to hard disks. It was found on a disk drive that used a BIOS that used HALTs extensively in the boot sequence and performed data transfers after the CPU entered the AutoHALT state. Since this occurs in both the AutoHALT state and the Stop Grant states of the SL Enhanced power management features, both of these features should not be used unless one of the listed workarounds is implemented. Not using the power management features could impact the compliance of an Energy Star Compliant System.

WORKAROUND: Use one of the following:

1. Disable the AutoHALT Powerdown feature by setting the TR12.AHD bit (bit 6) to '1' and do not assert STPCLK#.
2. Flush the caches before or upon entry into the AutoHALT or Stop Grant states. The flush will be serviced immediately if in the AutoHALT state, while the flush will be serviced after the Stop Grant state is exited.
3. Wake up the processor via an NMI or an R/S# prior to snooping the code cache three clocks before the EADS# of the snoop. If the system generates either NMI or R/S# pins based on AHOLD the processor will come out of the out of the low power state to service the Snoop. (This workaround assumes that the minimum 2 clock space between AHOLD and EADS# is not being used.)
4. Use the HIT# indication from the processor to flush the cache if the processor is in the AutoHALT Powerdown or Stop Grant state. The 75-, 90-, and 100-MHz Pentium processors asserts the HIT# pin when a snoop has caused a hit in the code cache. With this Workaround, it is necessary for the system to externally track the status of the cache line in the processor. (i.e., if the processor is in AutoHALT Powerdown or Stop Grant state and the INV pin was active during the snoop, then if the HIT# is returned active, assert FLUSH#.)

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

13. STPCLK# Assertion During Execution of the HALT Instruction Hangs System

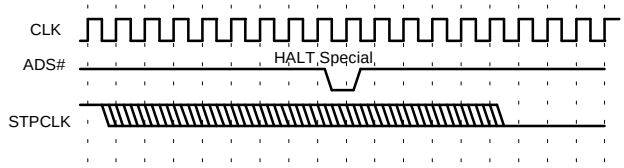
PROBLEM: If STPCLK# is asserted during the execution of the HALT instruction the processor will enter the Stop Grant mode without driving a Stop Grant bus cycle on to the bus. There is a 14 clock window around the ADS# for the HALT special cycle that this erratum occurs. (see figure) This erratum happens even when the AutoHALT power down feature is disabled. There are 3 scenarios for the symptom of this problem, depending on the way the system gets out of HALT.

1. When using INTR, the Interrupt Acknowledge Cycle appears on the bus, but then no other cycles. (The device has entered the Stop Grant state without issuing a Stop Grant special cycle.)
2. When using NMI or INIT, no bus cycles appear on the bus. (The device has hung up, the core has started a bus cycle but the clocks to the core have been stopped)
3. When using SMI#, the SMIACT# is driven asserted, and the SMM state dump completes, and no other cycles appear on the bus.

In addition when there is an event that brings the CPU out of HALT temporarily like FLUSH# or R/S#, if STPCLK# is asserted as the processor re-enters the HALT state, the erratum will occur.

At this time the processor has entered the Stop Grant mode but the part should have generated a Stop Grant special cycle prior to entry. If STPCLK# is deasserted for at least 1 clock, prior to the interrupt assertion the processor will resume correct operation.

The following figure depicts the minimum window around the HALT special cycle ADS#:



IMPLICATION: If the processor enters the Stop Grant state without issuing a Stop Grant special cycle, the state tracking machines of a chip set will be corrupted. A chip set will typically wait for the Stop Grant cycle before deasserting the STPCLK# pin. This will cause a system to hang.

WORKAROUND: Use one of the following:

- Do not use STPCLK# and disable the STPCLK# feature.
- If there is a way to deassert STPCLK# based on a timer tick, or the decode of the HALT special cycle prior to the interrupt then the system will not hang.
- If STPCLK# is being generated via software control such as an I/O instruction, then correct STPCLK# assertion can be guaranteed by using a code loop or string of no-ops that are equal to the latency of the STPCLK# assertion. As long as this code does not contain the HALT instruction, there is no possibility of this erratum occurring.

For example:

```
      MOV   CX, STPCLK_Delay  ;set the delay to the falling edge of STPCLK#.
      OUT   Stop_Clock_port, AX     ;trigger STPCLK#
L1:   NOP                           ;The loop delay should be at least equal to
                                    ;the hardware delay in asserting the STPCLK#
                                    ;signal.
      LOOP  L1
      NOP                                 ;Ten NOP instructions must follow the
         .                                ;assertion of STPCLK#.
         .
         .
      NOP
```

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

14. NMI or INIT During HALT Within SMM May Cause Large Amount of Bus Activity

PROBLEM: If a HALT or REP (repeat string instruction) instruction is executed while the processor is in System Management mode (SMM), and an NMI or INIT is asserted prior to interrupt initialization, the processor may continuously re-execute the HALT, and generate the HALT special cycle, or it will perform iterations of the REP instruction that was executed. Normally the processor would ignore NMI and INIT while in SMM. However, NMI and INIT will be enabled inside of SMM if interrupts have been enabled and then an INTR signal is recieved. Also, exceptions, when taken, enable NMI and INIT inside of SMM, but this behavior is not part of the Intel Architecture.

IMPLICATION: The processor may continuously run the same cycle on the bus until a non-masked interrupt is detected. There are no other problems associated with the erratum, the component resumes correct operation at this time. This impacts the "low power operation" that might have been expected with the use of a HALT while in SMM.

WORKAROUND: Use one of the following:

1. Do not use HALT while in SMM.
2. If the system must use HALT in SMM, the system is required to initialize interrupt vector tables prior to use of any interrupts, doing so will ensure the error will not occur. The system must ensure that NMI and INIT are not asserted while the processor is HALTed in System Management mode, prior to interrupt vector initialization.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

15. RUNBIST Restrictions When Run through Boundary Scan Circuitry

PROBLEM: When the built in self test (Runbist) is run via the Boundary Scan circuitry a failing result is shown on the device. This failing result appears even after initializing the RESET cell as described in Chapter 11 of the *Pentium® Processor Family Developer's Manual.*

IMPLICATION: If one of the workarounds listed is not implemented then the system cannot depend of the result of this test as part of a Boundary Scan generated manufacturing test or power on test.

WORKAROUND: Use one of the following workarounds. Both of these workarounds rely on the initialization of the RESET scan cell as stated in the Specification Clarifications section of this document.

1. Although not IEEE 1149.1 compatible, it has been found that if BRDY# is asserted low for every ADS# the processor generates, the Runbist test completes correctly. If the system can return these BRDY#s to the CPU then the BIST functionality can be utilized on the processor through Boundary Scan.
2. If RESET is held HIGH during the execution of the RUNBIST Boundary Scan instruction and the subsequent 2^{19} core clocks.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

16. FRC Mode Miscompare Due to Uninitialized Internal Register

PROBLEM: There is a mismatch and a resulting IERR# assertion when running in FRC mode due to an uninitialized internal register in the paging unit. The failure mechanism is due to uninitialized data being driven on the upper 32-bits of the data bus while updating a page table entry on the lower 32-bits (upon enabling paging). The data bits that mismatch are not valid during that bus cycle (byte enables are inactive), so the IERR# output is due to a spurious comparison.

IMPLICATION: The FRC mode of the processor requires use of a workaround to initialize the paging unit if addresses in the upper 32 bits are accessed.

WORKAROUND: Initialize this internal register through software by performing a dummy page table lookup on the upper 32 bits. (In a code segment with linear address bit 22 set to '1', turn paging on and then turn it off again immediately).

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

17. STPCLK# Restrictions During EWBE#

This erratum has been superseded by a specification change.

18. Multiple Allocations Into Branch Target Buffer

PROBLEM: A specific sequence of code may cause the Pentium processor to erroneously allocate the same branch into multiple ways of the Branch Target Buffer. These multiple entries may contain conflicting branch predictions. If this occurs and the branch address is accessed for a branch prediction, an incorrect entry may be written into the instruction cache, resulting in the possible execution of invalid or erroneous opcodes and probable activation of the IERR# signal. The incorrect write is dependent on process and circuit sensitivities which vary from one unit to the next.

The occurrence is extremely rare and is software dependent. A specific sequence of code is required to create the condition. In addition, Branch Target Buffer taken/not taken predictions associated with this code must proceed in a specific pattern.

Sensitive code can be summarized as follows: Any conditional jump (possibly paired with a previous instruction), sequentially followed by a move to a segment register, with any jump or instruction pair containing any jump at the target address (LBL_A below) of the first jump. An example follows:

```
                JCXZ LBL_A
                MOV ES, CX
                     .
                     .
                     .
LBL_A:          CALL LBL_B
```

For this erratum to occur, there must also be a specific pattern of taken/not-taken in the conditional jump (JCXZ), as well as a specific pattern of hit/miss in the segment descriptor cache for the segment register load.

IMPLICATION: When this erratum occurs, the processor may execute invalid or erroneous instructions and may assert IERR#. Depending on software and system configuration, the user may see an application error message or a system reset.

WORKAROUND: Several workarounds are available:

1. Disable the Branch Target Buffer by setting the NBP bit (0) to '1' in TR12. This results in approximately 7 percent degradation in performance on the BAPCo benchmark suite, a measure of typical system performance.
2. Use a software patch to avoid the sensitive sequence of instructions. The specific code sequence has only been observed in Windows* 3.10 and 3.11.
3. Ensure that the descriptor tables reside in non-cacheable areas of memory.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

19. 100-MHz REP MOVS Speed Path

PROBLEM: A speed path exists in the Pentium processor that may cause failures at the rated operating frequency of the part. Under certain rare and specific conditions, the speed path can cause the REP MOVS instruction to be misprocessed.

For this speed path to be exercised, the following conditions must be met:

1. The processor must be executing a REP MOVS instruction.
2. The source and destination operands must reside within the same cache line.
3. There must be a snoop coincident with the REP MOVS.

Because this is a speed path, its occurrence is dependent on temperature, voltage, and process variation (differs from one unit to the next). Failures have only been observed when operating near the upper limit of the temperature range and near the lower limit of the voltage range, and, then, only in a fraction of parts.

When this erratum occurs, the result is that an extra data item is copied during the REP MOVS. For example, in following code sequence, the Dword in memory location 108h may be erroneously copied into memory location 118h. When the error occurs, ESI will contain the value 10Ch rather than 108h and EDI will contain 11Ch rather than 118h.

```
MOV ECX, 2
MOV ESI, 100h
MOV EDI, 110h
REP MOVSD
```

The problem has been only observed in 100-MHz multi- or dual-processor machines with multi-threaded software; there have been no observed failures in uniprocessor systems. Multi- and dual-processing environments have higher processor utilization and more intense snoop activity than uniprocessor systems.

IMPLICATION: When this erratum occurs, the software may malfunction. This erratum has only been observed when running several instances of the WinBEZ* application on Windows NT* 3.1.

The failure may manifest itself in one of four ways:

1. A process window is dropped.
2. The screen locks with a red, vertical stripe.
3. The system hangs completely.
4. An application error message occurs.

WORKAROUND: Intel has implemented a tighter test screen to preclude future instances of this speed path. Operating the L1 cache in writethrough mode reduces the frequency of occurrence and provides additional margin in the system design. For multiprocessor systems with dedicated caches, Intel's TPC benchmark testing indicates that operating the L1 cache in writethrough mode results in less than a 5 percent performance impact. For shared-cache dual-processor systems, the performance impact is significantly higher, and L1 cache writethrough operation is not recommended.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

20. Overflow Undetected on Some Numbers on FIST

PROBLEM: On certain large positive input floating-point numbers, and only in two processor rounding modes, the instructions FIST[P] m16int and FIST[P] m32int fail to process integer overflow. As a consequence, the expected exception response for this situation is not correctly provided. Instead, a zero is returned to memory as the result.

The problem occurs only when all of the following conditions are met:

1. The FIST[P] instruction is either a 16- or 32-bit operation; 64-bit operations are unaffected.
2. Either the 'nearest' or 'up' rounding modes are being used. Both 'chop' and 'down' rounding modes are unaffected by this erratum.
3. The sign bit of the floating-point operand is positive.
4. The operand is one of a very limited number of operands. The table below lists the operands that are affected. For an operand to be affected, it must have an exponent equal to the value listed and a significand with the most significant bits equal to '1'. Additionally in the 'up' rounding mode, at least one of the remaining lower bits of the significand must be '1'. The Exponent and the two Significand columns describe the affected operands exactly, while the values in the column titled 'Approximate Affected Range' are only for clarity.

FIST[P] Operation	Rounding Mode	Unbiased Exponent	Upper Bits of Significand	Lower Bits of Significand	Approximate Affected Range
16 bit	Up	15	16 MSB's = '1'	At least one '1'	65,535.50 ± 0.50
	Nearest	15	17 MSB's = '1'	don't care	65,535.75 ± 0.25
32 bit	Up	31	32 MSB's = '1'	At least one '1'	4,294,967,295.50 ± 0.50
	Nearest	31	33 MSB's = '1'	don't care	4,294,967,295.75 ± 0.25
64 bit	Problem does not occur				

Actual VS. Expected Response

Actual Response

When the flaw is encountered, the processor provides the following response:

- A zero is returned as a result. This result gets stored to memory.
- The IE (Invalid Operation) bit in the FPU status word is not set to flag the overflow.
- The C1 (roundup) and PE bits in the FPU status word may be incorrectly updated.
- No event handler is ever invoked.

Expected Response

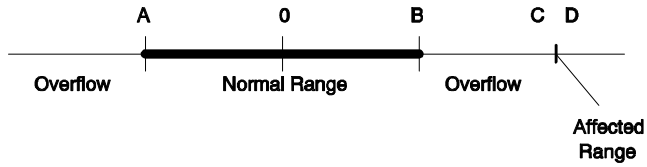
The expected processor response when the invalid operation exception is masked is:

- Return a value 10000....0000 to memory. Set the IE bit in the FPU status word.
- The IE (Invalid Operation) bit in the FPU status word is not set to flag the overflow.

The expected processor response when the invalid operation exception is unmasked is:

- Do not return a result to memory. Keep the original operand intact on the stack.
- Set the IE bit in the FPU status word.
- Appropriately vector to the user numeric exception handler.

IMPLICATION: An unexpected result is returned when an overflow condition occurs without being processed or flagged. Integer overflow occurs rarely in applications. If overflow does occur, the likelihood of the operand being in the range of affected numbers is even more rare. The range of numbers affected by this erratum is outside that which can be converted to an integer value. The figure below and corresponding table detail the normal range of numbers (between A and B) and the range affected by this erratum (between C and D):



16-bit Operation	A	B	C	D
Round Nearest	[-32,768.5]	(+32,767.5)	[+65,535.5]	(+65,536.0)
Round Up	(-32,769.0)	[+32,767.0]	(+65,535.0)	(+65,536.0)
32-bit Operation	**A**	**B**	**C**	**D**
Round Nearest	[-2,147,483,648.5]	(+2,147,483,647.5)	[+4,294,967,295.5]	(+4,294,967,296.0)
Round Up	(-2,147,483,649.0)	[+2,147,483,647.0]	(+4,294,967,295.0)	(+4,294,967,296.0)

NOTE:

[xxx.x] indicates the endpoint is included in the interval; (xxx.x) indicates the endpoint is *not* included in the interval.

Furthermore, given that the problem cannot occur in the 'chop' rounding mode, and given that the 'chop' rounding mode is the standard rounding mode in ANSI-C and ANSI-FORTRAN 77, it is unlikely that this condition will occur in most applications.

This erratum is not believed to affect application programs in general. Applications will need to handle exceptional behavior and take the appropriate actions to recover from exceptions. In order to do this applications will need to do either range checking prior to conversion or implement explicit exception handling. If an application relies on explicit exception handling the possibility of an error exists. It is, however, believed that applications written in languages that support exception handling will most likely do range checking, thereby allowing the application to be compiled with a no check option for performance reasons when the application has been debugged.

WORKAROUND: Any of three software workarounds will completely avoid occurrence of this erratum:

1. Range checking performed prior to execution of the FIST[P] instruction will avoid the overflow condition from occurring, and is likely to already be implemented.
2. Use the 'chop' or 'down' rounding modes. This erratum will never occur in these modes. By default, both ANSI-C and FORTRAN will use the 'chop' mode.
3. Use the FRNDINT (Round floating-point value to integer) instruction immediately preceding FIST[P]. FRNDINT will always round the value correctly.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

21. Six Operands Result in Unexpected FIST Operation

PROBLEM: For six possible operands and only in two processor rounding modes (up, down), the FIST or FISTP (floating-point to integer conversion and store) instructions return unexpected results to memory. Additionally, incorrect status information is returned under certain conditions in all 4 rounding modes.

The flaw occurs only on certain operands on the instructions FIST[P] m16int, FIST[P] m32int, and FIST[P] m64int. These operands are ±0.0625, ±0.125, and ±0.1875.

The following table details the conditions for the flaw and the results returned. For use of any of the six above-listed operands, refer to the left-hand side of the table in the column for a given combination of sign and rounding mode. The corresponding right-hand side of the table shows the results which will occur for the given conditions. These results include the C1 (Condition Code 1) and PE (Precision Exception) bits and, in two instances, storing of unexpected results.

Operand (any one of)	Rounding Mode	Result Expected / Actual	Status Bits	
			PE Expected / Actual	C1 Expected / Actual
+0.0625	nearest	SAME	1 / Unchanged	SAME
+0.1250	chop	SAME	1 / Unchanged	SAME
+0.1875	down	SAME	1 / Unchanged	SAME
	up	0001 / 0000	1 / Unchanged	1 / 0
–0.0625	nearest	SAME	1 / Unchanged	SAME
–0.1250	chop	SAME	1 / Unchanged	SAME
–0.1875	down	- 0001 / 0000	1 / Unchanged	1 / 0
	up	SAME	1 / Unchanged	SAME

IMPLICATION: An incor rect result (0 instead of +1 or -1) is returned to memory for the six previously listed operands. Incorrect results are returned to memory only when in the 'up' rounding mode with a positive sign or in the 'down' rounding mode with a negative sign. The majority of applications will be unaffected by this erratum since the standard rounding mode in ANSI-C and ANSI-FORTRAN 77 is 'chop', while BASIC and the ISO PASCAL intrinsic ROUND function use 'nearest' rounding mode. In addition, 'nearest' is also the default rounding mode in the processor.

Incorrect status information is also insignificant to most applications. Given that the PE bit in the numeric status register is 'sticky', and that most floating-point instructions will set this bit, PE will most likely have already been set to '1' before execution of the FIST[P] instruction and therefore the PE bit will not be seen to be incorrect. In addition, the unexpected values for the C1 bit are unlikely to be significant for most applications since the only usage of this information is in exception handling.

WORKAROUND: Either of two software workarounds will completely avoid this erratum.

1. Use the FRNDINT (Round floating-point value to integer) instruction immediately preceding FIST[P]. FRNDINT will always round the value correctly.
2. Use of 'nearest' or 'chop' modes will always avoid any incorrect result being stored (although the PE bit may have incorrect values).

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

22. Snoop With Table-Walk Violation May Not Invalidate Snooped Line

PROBLEM: If an internal snoop (as a result of ADS#) or external snoop (EADS#) with invalidation (INV) occurs coincident with a page table walk violation, the snoop may fail to invalidate the entry in the instruction cache. A page table walk violation occurs when the processor speculatively prefetches across a page boundary and this page is not accessible or not present. This violation results in a page fault if this code is executed. A page fault does not occur if the code is not executed.

For this erratum to occur, all the following conditions must be met:

1. A snoop with invalidation is run in the code cache. The snoop may be internal or external.

2. The Pentium processor is performing a page table walk to service an instruction TLB miss. The page table walk results in a violation (this may or may not lead to a page or other fault due to a speculative fetch).
3. The page table walk results in a violation (this may or may not lead to a page or other fault due to a speculative fetch).
4. The EADS# of the external snoop or ADS# of the table update occur within the window of failure.

   The window is defined by:

   a. 1-3 clocks after BRDY# is returned for the page directory or table read.
   b. 2-n clocks after BRDY# is returned for the page directory or table read if the set address of a buffered write matches that of the instruction cache lookup. “n” is determined by the time to complete two new data write bus cycles from the data cache.

IMPLICATION: This erratum has not been observed on any system. It was found only through investigation of component schematics, and Intel has only duplicated it on a proprietary test system by forcing failure conditions using the internal test registers. Its low frequency of occurrence is due to the way most systems operate; DMA devices snoop code 4 bytes at a time so that each line will get snooped and invalidated multiple times.

If this erratum occurs and a line is not invalidated in the instruction cache, then the instruction cache may have a coherency problem. As a result the processor may execute incorrect instructions leading to a GPF or an application error. This erratum affects only self Modifying code and bus masters/DMA devices. Due to necessary conditions, this erratum is expected to have an extremely low frequency of occurrence.

WORKAROUND: There are two workarounds. Because of the rarity of occurrence of this erratum, many OEMs may choose not to implement either workaround.

1. Rewrite the device driver for the DMA devices such that after DMA is complete, the instruction cache is invalidated using the TR5.cntl=11 (flush) and CD=0 (code cache) bits.
2. Disable the L1 cache.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

23. Slight Precision Loss for Floating-point Divides on Specific Operand Pairs

PROBLEM: For certain input datum the divide, remaindering, tangent and arctangent floating-point instructions produce results with reduced precision.

The odds of encountering the erratum are highly dependent upon the input data. Statistical characterization yields a probability that about one in nine billion randomly fed operand pairs on the divide instruction will produce results with reduced precision. The statistical fraction of the total input number space prone to failure is 1.14×10^{-10}. The degree of inaccuracy of the result delivered depends upon the input data and upon the instruction involved. On the divide, tangent, and arctangent instructions, the worst-case inaccuracy occurs in the 13th significant binary digit (4th decimal digit). On the remainder instruction, the entire result could be imprecise.

This flaw can occur in all three precision modes (single, double, extended), and is independent of rounding mode. This flaw requires the binary divisor to have any of the following bit patterns (1.0001, 1.0100, 1.0111, 1.1010 or 1.1101) as the most significant bits, followed by at least six binary ones. This condition is necessary but not sufficient for the flaw to occur. The instructions that are affected by this erratum are: FDIV, FDIVP, FDIVR, FDIVRP, FIDIV, FIDIVR, FPREM, FPREM1, FPTAN, FPATAN.

During execution, these instructions use a hardware divider that employs the SRT (Sweeney-Robertson-Tocher) algorithm which relies upon a quotient prediction PLA. Five PLA entries were omitted. As a result

of the omission, a divisor/remainder pair that hits one of these missing entries during the lookup phase of the SRT division algorithm will incorrectly predict a intermediate quotient digit value. Subsequently, the iterative algorithm will return a quotient result with reduced precision.

The flaw will **not** occur when a floating-point divide is used to calculate the reciprocal of the input operand in single precision mode, nor will it occur on integer operand pairs that have a value of less than 100,000.

IMPLICATION: For certain input datum, there will be a loss in precision of the result that is produced. The loss in precision can occur between the 13th and 64th significant binary digit in extended precision. On the remainder instruction the entire result could be imprecise.

The occurrence of the anomaly depends upon all of the following:

1. The rate of use of the specific FP instructions in the Pentium processor.
2. The data fed to them. The way in which the results are propagated for further computation by the application.
3. The way in which the results are propagated for further computation by the application.
4. The way in which the final result of the application is interpreted.

Because of the low probability of the occurrence with respect to the input data (one in nine billion random operand pairs), this anomaly is of low significance to users unless they exercise the CPU with a very large number of divides and/or remainder instructions per day, or unless the data fed to the divisor is abnormally high in sensitive bit patterns.

WORKAROUND: Due to the extreme rarity of this flaw, a workaround is not necessary for almost all users. However, Intel is replacing components for end users and OEM's upon request. In addition, a software patch is available for compiler developers. If you are a compiler developer, contact your local Intel representative to obtain this, or download from the World Wide Web Intel support server (www.intel.com).

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

A white paper, *Statistical Analysis of Floating-point Flaw in the Pentium™ Processor,* (Order Number 242481), is available that includes a complete analysis performed by Intel.

24. FLUSH#, INIT or Machine Check Dropped Due to Floating-point Exception

PROBLEM: HARDWARE FLUSH and INIT requests and Machine Check exceptions may be dropped if they occur coincidentally with a floating-point exception.

The following conditions are necessary for this erratum to occur:

1. Two floating-point instructions are paired and immediately followed by an integer instruction.
2. The floating-point instruction in the u-pipe causes a floating-point exception.
3. The FLUSH, INIT, or Machine Check occurs internally on the same clock as the integer instruction.

IMPLICATION: The processor caches will not be flushed, or the INIT request may be dropped.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

25. Floating-point Operations May Clear Alignment Check Bit

PROBLEM: The Alignment Check bit (bit 18 in the EFLAGS register) may be inadvertently cleared.

This erratum occurs if a fault occurs during execution of the FNSAVE instruction. After servicing the fault and resuming and completing the FNSAVE, the AC bit will be '0'. Expected operation is that the contents of AC are unchanged.

IMPLICATION: The only known use being made of the AC bit, at this time, is to aid code developers in aligning data structures for performance optimizations. As a result, there are no hardware or system application implications known to Intel at this time. Operating systems and applications will not be affected by this erratum.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

26. CMPXCHG8B Across Page Boundary May Cause Invalid Opcode Exception

PROBLEM: Use of the new Pentium processor specific CMPXCHG8B instruction may result in an invalid opcode exception.

If the CMPXCHG8B opcode crosses a page boundary such that the first two bytes are located in a page which is present in physical memory and the remaining bytes are located in a non-present page, unexpected program execution results. In this case, the processor generates an Invalid Opcode exception and passes control to exception handler number 6. Normal execution would be for a Page Fault to occur when the non-present page access is attempted, followed by reading in of the requested page from a storage device and completion of the CMPXCHG8B instruction.

IMPLICATION: This erratum only affects existing software which is Pentium processor aware and uses the CMPXCHG8B instruction. Any occurrence would generate an invalid opcode exception and pass control to exception handler 6.

WORKAROUND: Any software which uses Pentium processor specific instructions should ensure that the CMPXCHG8B opcode does not cross a page boundary.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

27. Single Step Debug Exception Breaks Out of HALT

PROBLEM: When Single Stepping is enabled (i.e. the TF flag is set) and the HLT instruction is executed the processor does not stay in the HALT state as it should. Instead, it exits the HALT state and immediately begins servicing the Single Step exception.

IMPLICATION: The behavior described above is identical to Intel486 CPU behavior.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

28. Branch Trace Message Corruption

PROBLEM: When performing execution tracing (in normal or fast mode), the linear address of the instruction causing the taken branch is sent to the bus as part of a branch trace message. In a tight loop of code, the reported linear address of the instruction causing the taken branch may be corrupted in some branch trace messages. If the first branch trace message completes on the bus before the second one is posted, the problem will be avoided. Note that this erratum applies to normal mode for processor steppings prior to C2 and to fast mode on all processor steppings.

IMPLICATIONS: This erratum only affects execution tracing, a specialized feature allowing external hardware to track the flow of instructions as they execute in the processor. Regular operation of the processor is not affected.

WORKAROUND: Use normal trace mode for processor steppings C2 and later since these steppings are not affected by this erratum in normal mode.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

29. FRC Lock Step Failure During APIC Write

PROBLEM: During FRC, the APIC write cycle is not driven on the external bus. When the internal APIC write data is compared with data on the external bus, the checker processor sees a comparison error and asserts IERR#.

IMPLICATIONS: This erratum only affects operation during FRC mode and will cause inadvertent IERR#'s to occur.

WORKAROUND: Ignore IERR# when doing APIC write cycles.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

30. BE4#-BE0# Sampled Incorrectly at Min Vih

PROBLEM: Due to strong internal pull down resistors on BE4#-BE0#, these pins are pulled low upon RESET high. A minimum input high voltage may be read into these pins incorrectly.

IMPLICATION: An input high voltage could be sampled as a "0", for instance, during APIC identification or boundary scan. This violates IEEE spec 1149.1 which states that regardless of the state of RESET in the boundary scan mode, the value driven should be that in the boundary scan cell.

WORKAROUND: To ensure a logical "1" is read, drive a minimum Input High Voltage (Vih) of 3.0V @6mA into these pins.

Otherwise, if testing boundary scan in a system environment which does not meet the above criteria, these pins can be left untested by marking these pins as "INTERNAL" in the BSDL file.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

31. Incorrect PCHK# Output During Boundary Scan if in DP Mode

PROBLEM: In a DP system, weak output drivers are enabled at PCHK# during the falling edge of RESET when APICEN and DPEN# are enabled on the primary processor, (and CPUTYP tied to Vss). For the dual processor the weak driver at PCHK# is enabled whenever APICEN is enabled, (and CPUTYP tied to V_{CC}).

The weak driver at PCHK# will drive a “1” with a long rise time, exceeding the specification for PCHK# max valid delay.

IMPLICATION: A weak “1” driven by PCHK# could be sampled as “0” during boundary scan testing. The maximum frequency of TCK may have to be decreased. This violates IEEE spec 1149.1 which states that regardless of the state of APICEN, DPEN#, CPUTYP or RESET in the boundary scan mode, the value driven should be that in the boundary scan cell.

WORKAROUND: To ensure a “1” is driven during boundary scan test, from the falling edge of TCK wait at least 55 nS before sampling PCHK#. The board’s capacitve load is estimated at 40pf. Please refer to *Pentium® Processor Family Developer’s Manual* Section 5.1.48 which specifies Ro=360 Ohms for the PCHK# weak driver.

Otherwise, if testing boundary scan in a system environment which does not meet the above criteria, these pins can be left untested by marking these pins as "INTERNAL" in the BSDL file.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

32. EIP Altered After Specific FP Operations Followed by MOV Sreg, Reg

PROBLEM: A specific sequence of code may cause corruption of the EIP. This specific code sequence must contain **all** of the following characteristics:

1. Three consecutive FP instructions
2. The third FP instruction must be immediately followed by a MOV Sreg, Reg instruction. (Sreg means one of the 6 segment registers. Note that **all other** ways of changing an Sreg, such as “POP Sreg”, “MOV Sreg, Mem”, LES, LDS, LSS etc., and far JMPs, CALLs & RETs etc., will **not** cause this problem.)
3. The descriptor selected by the MOV Sreg, Reg must not be available in the on-chip cache of the most recently used descriptors.
4. In addition, there are specific restrictions on each of the 3 consecutive FP instructions:
5. The first instruction in the sequence must be an FP instruction which can be paired with FXCH (FADD, FSUB, FMUL, FDIV, FLD, FCOM, FUCOM, FCHS, FTST & FABS.)
6. The second FP instruction must be FXCH, since it’s the only FP instruction that can go down the v-pipe (a pair of FP instructions must run together in u and v to allow this problem to occur).
7. The last (third) FP instruction must cause the CPU to test for an unsafe condition. Instructions that compare 2 numbers, and adjust FP flags as the result (FCOM(P), FICOM(P), FUCOM(P) & FTEST) are the usual sources of this test, but only with certain arguments (e.g. NANs and infinities). (In rare situations, FDIV (with a denominator of zero) and FSQRT (negative argument) will cause an unsafe condition test.)

```
Example:  FADD            ; is able to go into the u pipe & pair with FXCH in the v pipe
          FXCH            ; must be in the v pipe for this problem to occur
          FCOMP           ; can cause test for an unsafe condition
          MOV Sreg, Reg   ; selected descriptor must be a “miss’ in the segment descriptor cache
```

IMPLICATION: If this erratum occurs, an erroneous value is written into the EIP (instruction pointer), causing unpredictable results. This will most frequently result in an invalid opcode exception.

Intel knows of no existing application or OS code which involves this sequence. First, FXCH is rarely used just before a compare type instruction. Usually a calculation would be done just before compare which would set up one of the numbers to be compared at the FP stack top. Further, to make use of the compare, the FP flags must first be stored in a location where they can be tested, or loaded into EFLAGS for testing,

so the next instruction is typically FSTSW, not an Sreg load. Also, Sreg loads are typically not done by application code in a 32-bit environment.

WORKAROUND: Move any instruction between the compare and the Sreg load, such as FSTSW. Even NOP will work.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

33. WRMSR Into Illegal MSR Does Not Generate GP Fault

PROBLEM: The WRMSR and RDMSR instructions allow writing and reading of special MSRs (Model Specific Registers) based on the index number placed in ECX. The architecture was specified to reject access to illegal MSRs by generating the fault GP(0) if WRMSR or RDMSR is executed with an illegal index. However, negative indices, all of which are illegal, do not trigger GP(0).

IMPLICATION: If RDMSR is used with negative indices, undefined values will be read into EAX. If WRMSR is used with negative indices, undefined processor behavior may result.

WORKAROUND: Do not use illegal indices with WRMSR and RDMSR.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

34. Inconsistent Data Cache State From Concurrent Snoop and Memory Write

PROBLEM: This is a generalization of the Dual Processing specific erratum 5DP. Although this erratum has not been verified in a real Pentium processor-based system without using the DP mode, detailed analysis with the simulation model indicates that it is possible for the erratum to occur as described here. The possible occurrence requires the following conditions:

1. The processor begins a WRITE cycle to a writeback (WB) line in its L1 cache that is in the (S) (shared) state.
2. A non-invalidating snoop using AHOLD/EADS# is generated by another bus master by reading the *same* cache line *before* the completion of the WRITE.
3. There is at least one more cache in the system that holds a copy of the cache line.

When the snoop occurs during this window, the snoop is mishandled and the cache line will transition to the exclusive (E) state instead of the shared (S) state. An additional write to the same cache line by the processor will cause a cache state transition from (E) to modified (M) and will not generate a bus cycle. Since a bus cycle is not generated, other caching agents in the system that hold the cache line in the shared (S) state will not be updated and will contain stale data.

This erratum occurs because the sequence of cycles completed inside the processor is different from the sequence of cycles started on the bus, which is a memory write by the processor followed by a snoop on the same address. Inside the processor, the snoop occurs, and then the memory write completes. This can only happen if the snoop occurs in a window between the ADS# assertion by the processor for its WRITE cycle, and up to 2 CPU clocks after the system assertion of BRDY# at the end of the WRITE cycle.

This erratum can only occur if AHOLD/EADS# is used for snoops; if HOLD or BOFF# are used to force a snoop before the processor WRITE is completed, it will be restarted after the snoop and handled correctly. In addition, this erratum cannot occur on memory updates where the LOCK# signal is asserted.

IMPLICATION: The processor's L1 cache may become incoherent with an external cache. A memory read cycle by an external bus master could read stale data.

WORKAROUND: Designs which do not snoop under AHOLD are not affected. Uniprocessor systems using a lookaside L2 cache, such as those built with either the 82430NX or 82430FX PCIsets, are not affected because a read by an external bus master will always snoop the L1 as well as L2. Intel knows of no uniprocessor implementation which is subject to this erratum. While this erratum is more likely to occur in a multi-processing environment, Intel is not aware of any designs which have demonstrated a susceptibility to this issue.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

35. BE3#-BE0# Not Driven During Boundary Scan if RESET High

PROBLEM: During boundary scan, BE3#-BE0# are always tristated when RESET is high regardless of the value in the control cell. If RESET is low the proper value will be driven.

IMPLICATION: When RESET is high stong pull downs are enabled, and the value of “0” will always appear at the BE3#-BE0# pins. This violates IEEE spec 1149.1 which states that regardless of RESET in the boundary scan mode, the value driven should be that in the boundary scan cell.

WORKAROUND: If testing boundary scan in a system environment, these pins can be left untested by marking these pins as "INTERNAL" in the BSDL file.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

36. Incorrect FIP After RESET

PROBLEM: After a RESET, the floating point instruction pointer (FIP) should be initialized to 00000000h. The FIP will instead retain the value it contained prior to the RESET. The FIP gets updated whenever the processor decodes a floating point instruction other than an administrative floating point instruction (FCLEX, FLDCW, FSTCW, FSTSW, FSTSWAX, FSTENV, FLDENV, FSAVE, FRSTOR and FWAIT). If an FSAVE or FSTENV is executed after a RESET and before any non-administrative floating point instruction caused the FIP to be updated, the old value contained in the FIP will be saved. If a non-administrative floating point instruction is the first floating point instruction executed after RESET, the old value in the FIP will be overwritten and any successive FSAVE of FSTENV will save the correct value.

The FIP is used by software exception handlers to determine which floating point instruction caused the exception. The only instructions that can cause an exception are non-administrative floating point instructions, so a non-administrative floating point instruction is usually executed before an FSAVE or FSTENV.

IMPLICATION: If an FSAVE or FSTENV is executed after a RESET and before any non-administrative floating point instruction, the incorrect FIP will be saved.

WORKAROUND: If an FSAVE or FSTENV is executed after a RESET and before a non-administrative floating point instruction is executed, perform an FINIT instruction after RESET as recommended in the *Intel Architecture Software Developer’s Manual*, Volume 3, Section 8.2. This will set the FIP to 00000000h. Otherwise, no workaround is required.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

37. Second Assertion of FLUSH# Not Ignored

PROBLEM: If FLUSH# is asserted while the processor is servicing an existing flush request, a second flush operation will follow after the first one completes. Proper operation is for a second assertion of FLUSH# to be ignored between the time the first FLUSH# is asserted and completion of its Flush Acknowledge cycle.

IMPLICATION: A system that asserts FLUSH# during a flush that's already in progress will flush the cache a second time. Flushing the cache again is not necessary and results in a slight performance degradation.

WORKAROUND: For best performance, the system hardware should not assert any subsequent FLUSH# while a flush is already being serviced.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

38. Segment Limit Violation by FPU Operand May Corrupt FPU State

PROBLEM: On the Intel486, 80386 and earlier processors, if the operand of the FSTENV/FLDENV instructions, or the FSAVE/FRSTOR instructions, exceeds a segment limit during execution, the resulting General Protection fault blocks completion of the instruction. (Actually, interrupt #9 is generated in the 80386 and earlier.) This leaves the FPU state itself (with FLDENV, FRSTOR) or its image in memory (with FSTENV, FSAVE) partly updated, thus corrupted, and the instruction generally is non-restartable. It is stated in the *Intel Architecture Software Developer's Manual*, Volume 3, Chapter 17 that the Pentium processor fixes this problem by starting these instructions with a test read of the first and last bytes of the operand. Thus if there is a segment limit violation, it is triggered before the actual data transfer begins, so partial updates cannot occur.

This improvement works as intended in the large majority of segment limit violations. There is however a special case in which the beginning and end of the FPU operand are within the segment, so the endpoints pass the initial test, but part of the operand exceeds the segment limit. Thus part way through the data transfer, the limit is violated, the GP fault occurs, and thus the FPU state is corrupted. Note that this is a subset of the cases which will cause the same problem with Intel486 and earlier CPUs, so any code that executes correctly on those CPUs will run correctly on the Pentium processor.

This erratum will happen when both the segment limit and a 16 or 32 bit addressing wrap around boundary falls within the range of the FPU operand, with the segment limit below the wrap boundary. (To use a 16 bit wrap boundary of course, one must be executing code using 16 bit addressing.) The upper endpoint of the FPU operand wraps to near the bottom of the segment, so it passes the initial test. But part way through the data transfer the CPU tries to access memory above the segment limit but below the wrap boundary, causing the GP fault with the FPU state partly copied. This erratum can also happen if the segment limit is at or above a 16 bit addressing wrap boundary, with both straddled by an FPU operand that is **not aligned on an 8 byte boundary.** Test of the upper endpoint wraps and thus passes. But when the instruction is actually transferring data, the misalignment forces the CPU to calculate extra addresses for special bus cycles. This special address calculation does not support the 16 bit wrap, so the GP fault is triggered when the segment limit is crossed.

Note that the *Intel Architecture Software Developer's Manual*, Volume 3, Chapter 17 warns in general that the Pentium processor may store only part of operands which generate a memory fault by crossing either a segment or page limit. This erratum is just one case of that general problem, and all cases will be avoided by following the recommended programming practice of never straddling segment or page boundaries with operands. Note also that the handling of operands which straddle such boundaries is processor specific, so code which uses such straddling will behave differently when run on different Intel Architecture processors.

IMPLICATION: This erratum can corrupt that state of the FPU and will cause a GP fault. This generally will require that the task using the FPU be restarted, but it will not cause unflagged errors in results. Code written following Intel recommendations, and any code which runs on the Intel486 (or earlier) CPUs, will not cause this erratum. The case where the Pentium processor will experience this erratum is a small subset of the cases in which the Intel486 (and earlier) CPUs will be corrupted.

WORKAROUNDS:

1. Do not use code in which FPU operands wrap around the top of their segments.

2. If one must use FPU operands which wrap at the top of their segments, make sure that they are aligned on an 8 byte boundary, **and** that the segment limit is not below the 16 or 32 bit wrap boundary.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

39. FP Exception Inside SMM with Pending NMI Hangs System

PROBLEM: If a previous FPU instruction has caused an unmasked exception, and an FP instruction is executed inside SMM with an NMI pending, the system will hang unless the system is both DOS compatible (CR0.NE=0), **and** external interrupts are enabled.

IMPLICATION: For standard PC-AT systems, NMI is typically used (if at all) to indicate a parity error, and the response required is a system reset, to preserve data integrity. So this erratum will only occur when the system has already suffered a parity error; the effect of the erratum is only to force reset inside SMM, instead of after the RSM when the NMI would normally be serviced. In a system where NMI is **not** used for an error that requires shutdown, the workaround should be implemented.

A properly designed system should not experience a hang-up. In such a system the SMM BIOS checks for pending interrupts before issuing an FSAVE or FRSTOR. If an interrupt is pending, the BIOS will exit SMM to handle the interrupt. If an interrupt is not present, the BIOS will disable interrupts (for example, it will disable NMI by writing to the chip set) and only then will issue the FP instruction.

WORKAROUND: If FPU instructions are used in SMM, **and** NMI is used for other than an error that requires shutdown, NMI should be blocked from outside the CPU during SMM.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

40. Current in Stop Clock State Exceeds Specification

PROBLEM: During the stop clock mode, a portion of the internal processor circuitry may remain active. This problematic circuitry is part of the boundary scan logic, and is not properly reset unless boundary scan is run. The current drawn will result in a violation of the stop clock power dissipation specification.

IMPLICATION: The Pentium processor will draw more current than specified for the stop clock state. The specification for Stop Clock Power dissipation is 50mW. Under the condition of the erratum, the power consumption can reach 60-80mW, depending on the type of processor and core voltage.

WORKAROUND: Use one of following:

1. Ground TCK or TRST#. Note that this will NOT allow boundary scan to be run.
2. Ground TCK or TRST# through a 1K Ohm resistor. This will allow a normal boundary scan to be performed if desired.

For power sensitive designs, please observe that as both TCK and TRST# have an internal 30K Ohm pull-up resistor, the workarounds will not result in any significant additional current draw.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

41. STPCLK# Buffer Samples Incorrectly During Boundary Scan Testing

PROBLEM: During boundary scan input testing, the boundary scan input path in the STPCLK# buffer is disabled when RESET is high.

IMPLICATION:. The boundary scan cell in the STPCLK# buffer captures a "1" from the STPCLK# pin regardless of the actual data on that pin when RESET is high. This violates the IEEE Specification 1149.1 which states that the value driven should always be that in the boundary scan cell regardless of the state of RESET. However, the buffer functions correctly when the EXTEST instruction is used.

WORKAROUND: If testing boundary scan in a system environment this pin can be left untested by marking this pin as "INTERNAL" in the BSDL file.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

42. Incorrect Decode of Certain 0F Instructions

PROBLEM: With a specific arrangement of instructions in memory and certain asynchronous events, the processor may incorrectly decode certain 0F prefixed instructions.

In order for this erratum to occur there must be a very specific arrangement of instructions in memory. In conjunction, these instructions must be resident in the cache and an asynchronous cache line replacement must occur during execution of these instructions. The conditions for this erratum are as follows:

1. There is a 0F prefix instruction (other than 0F80-0F8F) which can begin in the range of the most significant byte of the first cache line (1F) through the 0E byte of the second cache line, followed by the rest of the 0F prefixed instruction. (See cache lines 1 & 2 in Fig. 1; the 0F byte is at offset 1F in line 1.)
2. The processor must execute a branch to the second half of the first cache line shown below (10 - 1F).
3. There is a replacement or invalidation of cache line 2 shown below. This replacement or invalidation must complete in a narrow window (3 or less CPU clocks) between the decode of the 0F byte from the instruction queue, and the decode of the rest of the instruction.
4. The third consecutive cache line must contain the bit pattern 0F80 - 0F8F offset by 33 bytes from the 0F byte of the first instruction. There must be a spacing of exactly 33 bytes between the first and subsequent 0F bytes.
5. All 3 lines must already be resident in the instruction cache and must be from sequential linear memory addresses.

This erratum can occur only if **all** of the above conditions are met. After the first byte (0F) of the opcode is decoded, but before the rest of the instruction can be decoded, the second cache line gets replaced or invalidated. While the processor is waiting for the line containing the second byte of the 0F opcode to be read in again from memory, the 2 bytes offset by 33 bytes from the 0F byte of the stalled instruction are temporarily presented to the instruction decoder. Normally this data would be completely ignored, however if the bit pattern is in the range of 0F80 - 0F8F, then the decode of the 0F byte of the stalled instruction is discarded. (Note that this happens **only** with the 0F prefixed instructions.) When the cache line fill has returned the missing line, the second byte of the stalled instruction is incorrectly interpreted as the start of the instruction.

3 Consecutive L1 Cache Lines, Holding Consecutive Code

1F	10	0E	00	Byte offset within cache line
			8X 0F	N + 40h = Memory Addr. for code in Cache Line 3
		0F.....		N + 20h = Memory Addr. for code in Cache Line 2
0F				N = Memory Addr. for code in Cache Line 1

IMPLICATION: When this erratum occurs, the processor will execute invalid or erroneous instructions. Depending on software and system configuration, the user will typically see an application error message or system reset.

WORKAROUND: There are currently no workaround identified for existing code.

This erratum may be eliminated in code that is being created or recompiled as follows: The compiler must check for the occurrence of bytes 0F, not 8X, 31 other bytes, 0F and 8X. When such an occurrence is found, a NOP inserted or any other change in spacing will prevent the alignment required for this erratum to occur.

A loader based workaround can also be implemented as follows: At load time, scan the executable for the existence of a 0F instruction (other than 0F8X), check the cache alignment of the 0F instruction and check for the existence of a 0F8X bit pattern 33 bytes beyond the 0F byte of the first instruction. If these conditions are found, the page containing this code sequence can be marked as non-cacheable.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

43. Data Breakpoint Deviations

The following three problems are deviations from the data breakpoint specification when a fault occurs during an FP instruction while the data breakpoint is waiting to be serviced. They all share the same workaround. In the first case the breakpoint **is serviced,** incorrectly, before the actual data access that should trigger it takes place; in the other cases the breakpoint is **not serviced** when it should be.

PROBLEM A: First, the debug registers must be set up so that any of the FP instructions which read from memory (except for FRSTOR, FNRSTOR, FLDENV and FNLDENV) will trigger a data breakpoint upon accessing its memory operand. **Second**, there must be an unmasked FP exception pending from a previous FP instruction when the FP load or store instruction enters the execution stage. This so far would cause, per specification, a branch to the FP exception handler. The data breakpoint would not be triggered until/ unless the memory access is made after return from the exception handler. But if **third**, either of the external interrupts INTR or NMI is asserted after the FP instruction enters the execution stage, but before the branch to the FP exception handler occurs, this erratum is generated. In this situation, the processor should branch to the external interrupt handler, but instead it goes to the data breakpoint handler. This is incorrect because the data access that should trigger the breakpoint has not occurred yet.

PROBLEM B: Interrupts are blocked for the instruction after a MOV or POP to SS (to allow a MOV or POP to ESP to complete a stack switch before any interrupt). If the MOV or POP to SS triggers a data breakpoint, it normally is serviced after the following instruction is executed. However, if the following instruction is a FP instruction **and** there is a pending FP error from a preceding FP instruction (even if the error is masked), the delayed data breakpoint is forgotten.

PROBLEM C: If the sequence of memory accesses during execution of FSAVE or FSTENV (or their counterparts FNSAVE and FNSTENV) touches an enabled data breakpoint location, the data breakpoint exception (interrupt 1) occurs at the end of the FP instruction. If however the sequence of memory accesses cross a segment limit after touching the data breakpoint location, the General Protection (GP) fault will occur. This erratum is that as the processor branches to the GP fault handler, the valid data breakpoint is forgotten.

IMPLICATION: This erratum will only be seen by software or hardware developers using the data breakpoint feature of the debug registers. It can cause data breakpoints to be both lost, and asserted prematurely, as long as the contributing FP and GP errors remain uncorrected.

WORKAROUND: Use one of the following:

1. General solution: For problems A & B to occur, an FP error must be caused by a preceding FP instruction, and in problem C, the FP operand causes a segment limit violation. These errors are all indicated in the normal way, despite this erratum. Eliminate them and this erratum disappears, allowing the data breakpoint debugging to proceed normally. Since debugging is usually done in successive stages, this workaround is usually performed as part of the debugging process.
2. Problem A may also be handled by blocking NMI and INTR during debugging.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

44. Event Monitor Counting Discrepancies

PROBLEM: The Pentium processor contains two registers which can count the occurrence of specific events used to measure and monitor various parameters which contribute to the performance of the processor. There are several conditions where the counters do not operate as specified.

In some cases it is possible for the same instruction to cause the "Breakpoint match" (event 100011, 100100, 100101 or 100110) event counter to be incremented multiple times for the same instruction. Instructions which generate FP exceptions may be stalled and restarted several times causing the counter to be incremented every time the instruction is restarted. In addition, if FLUSH# or STPCLK# is asserted during a matched breakpoint or if a data breakpoint is set on a POP SS instruction, the counter will be incremented twice. The counter will incorrectly not get incremented if the matched instruction generates an exception and the exception handler does an IRET which sets the resume flag. The counter will also not get incremented for a data breakpoint match on a u-pipe instruction if the paired instruction in the v-pipe generates an exception.

The "Hardware interrupts" (event 100111) event counter counts the number of **taken** INTR and NMIs. In the event that both INTR/NMI and a higher priority interrupt are present on the same instruction boundary, the higher priority interrupt correctly gets processed first. However, the counter prematurely counts the INTR/NMI as taken and the count incorrectly gets incremented.

The "Code breakpoint match" (event 100011, 100100, 100101 or 100110) event counter may also fail to be incremented in some cases. If there is a code breakpoint match on an instruction and there is also a single-step or data breakpoint interrupt pending, the code breakpoint match counter will not be incremented.

The "Non-cacheable memory reads" (event 0111110) event counter is defined to count non-cacheable instruction or data memory read bus cycles. Reads to I/O memory space are not supposed to be counted. However, the counter incorrectly gets incremented for reads to I/O memory space.

The "Instructions executed" (event 010110) and "Instructions executed in the v-pipe" (event 010111) event counters are both supposed to be incremented when any exception is recognized. However, if the instruction in the v-pipe generates an exception and a second exception occurs before execution of the first instruction of the exception handler for the first exception, the counter incorrectly does not get incremented for the first exception.

The "Stall on write to an E or M state line" (event 011011) event counter counts the number of clocks the processor is stalled on a data memory write hit to an E or M state line in the internal data cache while either the write buffers are not empty or EWBE# is not asserted. However, it does not count stalls while the write buffers are not empty, it only counts the number of clocks stalled while EWBE# is not asserted.

The "Code TLB miss" (event 001101) and "Data TLB miss" (event 000010) event counters incorrectly get incremented twice if the instruction that misses the code TLB or the data that misses the data TLB also causes an exception.

The "Data read miss" (event 000011) and "Data write miss" (event 000100) event counters incorrectly get incremented twice if the access to the cache is misaligned.

The "Bank conflicts" (event 001010) event counter may be incremented more than once if a v-pipe access takes more than 1 clock to execute.

The "Misaligned data memory or I/O References" (event 001011) incorrectly gets incremented twice if the access was caused by a FST or FSTP instruction.

The "Pipeline flushes" (event 010101) event counter may incorrectly be incremented for some segment descriptor loads and the VERR instruction.

The “Pipeline stalled waiting for data memory read” (event 011010) event counter incorrectly counts a misaligned access as 2 clocks instead of 3 clocks, unless it misses the TLB.

IMPLICATION: The event monitor counters report an inaccurate count for certain events.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

45. VERR Type Instructions Causing Page Fault Task Switch with T Bit Set May Corrupt CS:EIP

PROBLEM: This erratum can only occur during debugging with the T bit set in the Page Fault Handler’s TSS. It requires the following very specific sequence of events:

1. The descriptor read caused by a VERR type instruction must trigger a page fault. (These instructions are VERR, VERW, LAR and LSL. They each use a selector to access the selected descriptor and perform some checks on it.)
2. The OS must have the page fault handler set up as a separate task, so the page fault causes a task switch.
3. The T bit in the page fault handler’s TSS must be set, which would normally cause a branch to the interrupt 1 (debug exception) handler.
4. The interrupt 1 handler must be in a not present code segment.

The not present code segment should cause a branch to interrupt 11. However, because of this erratum, execution begins at an invalid location selected by the CS from the page fault handler TSS but with the EIP value pointing to the instruction just beyond the VERR type instruction.

IMPLICATION: This erratum will only be seen by software or hardware developers setting the T bit in the page fault handler’s TSS for debugging. It requires that the OS in use has the page fault handler set up as a separate task, which is not done in any standard OS. Even when these conditions are met, the other conditions will cause this erratum to occur only infrequently. When it does occur, the processor will execute invalid or erroneous instructions. Depending on software and system configuration, the developer will typically see an application error message or system reset.

WORKAROUND: If debugging a system in which the page fault handler is a separate task, use one of the following:

1. Do not set the T bit in the page fault handler’s TSS.
2. Ensure that the code segment where the debug exception handler starts is always present in the system memory during debugging.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

46. BUSCHK# Interrupt Has Wrong Priority

PROBLEM: Section 2.7 of the *Pentium® Processor Family Developer’s Manual* lists the priorities of the external interrupts, with BUSCHK# as the highest (if the BUSCHK# interrupt, AKA the machine check exception, is enabled by setting the MCE bit in CR4), and INTR as the lowest. It is also specified that STPCLK# is the very lowest priority external interrupt for those Pentium processors provided with it (all CPUs with a core frequency of 75 MHz and above). Consistently with this specification, the CPU blocks all other external interrupts once execution of the BUSCHK# exception handler begins.

However this erratum can change the effective priority for a given assertion of BUSCHK# in the following cases:

CASE 1: An additional external interrupt (except INTR) or a debug exception occurs during a narrow window after the CPU begins to transfer control to the BUSCHK# handler, but before the first instruction of the handler begins execution.

In this case, the other interrupt may be serviced before BUSCHK# is serviced. Thus for other interrupts that occur during this narrow window, BUSCHK# is effectively treated as the next to lowest priority interrupt instead of the highest.

CASE 2: The following conditions must all apply for this case to cause an erratum:

1. A machine check request (INT 18) is pending
2. A FLUSH# or SMI# request is pending
3. A single step or data breakpoint exception (INT 1) is pending
4. The IO_Restart feature is enabled (i.e. TR12 bit 9 is set)

Given the above set of conditions, the interrupt priority logic does not recognize the machine check exception as the highest priority. The processor will not service the FLUSH#/SMI# nor the debug exception (INT 1). Instead, it will generate an illegal opcode exception (INT 6).

IMPLICATION: Most systems do not use BUSCHK# and thus are unaffected by this erratum. For those that do use BUSCHK#, the pin allows the system to signal an unsuccessful completion of a bus cycle. This would only occur in a defective system. (Since BUSCHK# is an “abort” type exception, it cannot be used to handle a problem from which the OS intends to recover; BUSCHK# always requires a system reset.)

Due to this erratum, the BUSCHK# interrupt would either occasionally be displaced by another interrupt (which incorrectly would be serviced first) or an unexpected illegal opcode exception (INT 6) would be generated and the pending machine check would be skipped.

Depending on the system and also the severity of the defect, this delay of the BUSCHK# interrupt (case #1 above) could cause a system hang or reset before a bus cycle error message is displayed by the BUSCHK# interrupt. In case #2 above where an illegal opcode exception (INT 6) is generated instead of the machine check exception, a properly architected INT 6 handler will usually require a reset since this handler was erroneously entered without an illegal opcode. But in any event, the normal outcome of a bus cycle error is to require a system reset, so the practical result of this erratum is just the occasional loss of the proper error message in a defective system.

Another problem can occur due to this erratum if the system is using the SMM I/O instruction restart feature. This problem requires an improbable coincidence: the SMI# signal caused by an I/O restart event must occur essentially simultaneously with BUSCHK#, such that the SMI# interrupt hits the narrow window (as described above) just before the first instruction of the BUSCHK# handler begins execution. This could happen if the same I/O instruction that triggers SMI# (usually to turn back on a device that’s been turned off to save power) also generates a bus failure due to the system suddenly going defective, thus signaling BUSCHK#. The result is that the SMI# interrupt is serviced after the EIP has already been switched to point to the first instruction of the BUSCHK# handler, instead of the I/O instruction. The SMM code that services the I/O restart feature may well use the image of EIP in the SMRAM state save memory to inspect the I/O instruction, for example to determine what I/O address it’s trying to access. In this case, the I/O restart part of SMM code will not find the correct instruction. If it is well written, it will execute RSM when it determines there is no valid I/O access to service. Then execution returns to the BUSCHK# handler with no deleterious impact. But less robust code might turn on the wrong I/O device, hang up, or begin executing from a random location.

WORKAROUND: Do not design a system which relies on BUSCHK# as the highest priority interrupt. If using SMM, do not use BUSCHK# at all.

Note that Case 2 does not apply to B1, C1 or D1 steppings of the 60- and 66-MHz Pentium processors.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

47. BF and CPUTYP Buffers Sample Incorrectly During Boundary Scan Testing

PROBLEM: During boundary scan input testing, the boundary scan input paths in the BF0, BF1 and CPUTYP buffers are disabled when RESET is **low**. (Note that this is different from the BSDL testing problem with STPCLK#, documented as Erratum 41; STPCLK# is "stuck" when RESET is **high**.)

IMPLICATION: The boundary scan cells in the BF0, BF1 and CPUTYP buffers capture a "1" from their pins, regardless of the actual data on the pins, when RESET is low. This violates the IEEE specification 1149.1 which states that the value captured should always be that on the pins regardless of the state of RESET. However, the buffer functions correctly when the EXTEST instruction is used.

WORKAROUND: If testing with boundary scan in a system environment these pins can be left untested by marking them "INTERNAL" in the BSDL file.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

48. Matched But Disabled Data Breakpoint Can Be Lost By STPCLK# Assertion

PROBLEM: Assertion of STPCLK# can interfere with a feature described in the *Intel Architecture Software Developer's Manual*, Volume 3, Section 14.2.3: "The processor sets the DR6 B bits for all breakpoints which match the conditions present at the time the debug exception is generated, whether or not they are enabled." When the debug exception is generated, all breakpoints which match the conditions present at that time are flagged by a bit set in a temporary register. If STPCLK# is asserted after this, but before control is transferred to the debug exception handler (interrupt 1), a matched but disabled data breakpoint may not be transferred from the temporary register. That is, as a result of the STPCLK# assertion, the B bit corresponding to that breakpoint may not get set in DR6.

IMPLICATION: This feature (defining disabled breakpoints) can be used in debugging; e.g., one can set a disabled data breakpoint on a memory location and then check the corresponding bit in DR6, to see if the location has been accessed by the most recent (main code) instruction, any time one is in the debug handler for some other reason. This erratum will sometimes cause this debug feature to fail to set its DR6 bit, when STPCLK# is also being used.

WORKAROUND: Use one of the following:

1. Use only *enabled* data breakpoints when STPCLK# may be asserted.
2. Disable the assertion of STPCLK# while this debug feature is being used.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

49. STPCLK# Ignored In SMM When INIT or NMI Pending

PROBLEM: If an INIT or NMI is pending while in SMM mode, and STPCLK# is asserted, the stop clock interrupt is not serviced. The correct operation is for the stop clock request to be serviced while in SMM, regardless of pending NMI or INIT.

IMPLICATION: The stop clock request is blocked until after the processor exits SMM and services the pending NMI or INIT. The processor then services the lower priority stop clock interrupt.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

50. STPCLK# Pullup Not Engaged at RESET

PROBLEM: The internal pullup on the STPCLK# pin may not pullup at power on if the pin is floating at a low input level.

IMPLICATION: If the STPCLK# pin is floating at a low input level and the pin is left unconnected at bootup, the processor may inititiate the stop grant bus cycle in response to the STPCLK# request shortly after completing the reset sequence. This may result in a system hang.

WORKAROUND: Use one of the following:

1. Always drive a valid logic level on STPCLK# (including during RESET).
2. Use an external pullup.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

51. A Fault Causing a Page Fault Can Cause an Instruction To Execute Twice

PROBLEM: When the processor encounters an exception while trying to begin the handler for a prior exception, it should be able to handle the two serially (i.e. the second fault is handled and then the faulting instruction is restarted, which causes the first fault again, whose handler now should begin properly); if not, it signals the double-fault exception. A "contributory" exception followed by another contributory exception causes the double-fault, but a contributory exception followed by a page fault are both handled. (See the *Intel Architecture Software Developer's Manual*, Volume 3, Section 5.12, Interrupt 8 for the list of contributory exceptions and other details.) This erratum occurs under the following circumstances:

1. One of these three contributory faults: #12 (stack fault), #13 (General Protection), or #17 (alignment check), is caused by an instruction in the v-pipe.
2. Then a page fault occurs before the first instruction of the contributory fault handler is fetched. (This means that a page fault that occurs because the handler starts in a not present page will *not* cause this erratum.)

The result is that execution correctly branches to the page fault handler, but *an incorrect return address is pushed on the stack*: the address of the (immediately preceding) u-pipe instruction, instead of the v-pipe instruction that caused the faults. This causes the u-pipe instruction to be executed an extra time, after the page fault handler is finished.

IMPLICATION: When this erratum occurs, an instruction will be (incorrectly) executed, effectively, twice in a row. For many instructions (e.g. MOV, AND, OR) it will have no effect, but for some instructions it can cause an incorrect answer (e.g. ADD would increase the destination by double the correct amount). However, the page fault (during transfer to the handler for fault #12, #13 or #17) required for this erratum to occur can happen in only three unusual cases:

1. If the alignment check fault handler is placed at privilege level 3, the push of the return address could cause a page fault, thus causing this erratum. (Fault #17 can only be invoked from level 3, so it is legal to have its handler at level 3. Fault 12 and 13 handlers must always be at level 0 since they can be invoked from level 0. The push of a return address on the level 0 stack *must not* cause a page fault, because if the OS allowed that to happen, the push of return address for a regular page fault could cause a second page fault, which causes a double-fault and crashes the OS.)
2. If the descriptor for the fault handler's code segment (in either the GDT or the current LDT) is in a not present page, a page fault occurs which causes this erratum.
3. If the OS has defined the fault handler as a separate task, and a page fault occurs while bringing in the new LDT or initial segments, this erratum will occur.

WORKAROUND: All of the following steps must be taken (but 2 & 3 are part of normal OS strategy, done in order to optimize speed of access to key OS elements, and minimize chances for bugs): 1). If allowing the alignment fault (#17), place its handler at level 0. 2). Do not allow any of the GDT or current LDT to be "swapped out" during virtual memory management by paging. 3). Do not use a separate task for interrupts 12, 13 or 17.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

52. Machine Check Exception Pending, then HLT, Can Cause Skipped or Incorrect Instruction, or CPU Hang

PROBLEM: This erratum can occur if a machine check exception is pending when the CPU encounters a HLT instruction, or occurs while the CPU is in the HLT state. (E.g. the BUSCHK# error could be caused by executing the previous instruction, or by a code prefetch.) Before checking for pending interrupts, the HLT instruction issues its special bus cycle, and sets an special internal flag to indicate that the CPU is in the HLT state. The machine check exception (MCE) can then be detected, and if it is present the CPU branches to the MCE handler, but *without clearing the special HLT flag - the source of this erratum*. As when other interrupts break into HLT, the return address is that of the next instruction after HLT, so execution continues there after return from the MCE handler.

Except for MCE (and some cases of the debug interrupt), interrupts clear the special HLT flag before executing their handlers. The erratum that causes the MCE logic to not clear the HLT flag in this case can have the following consequences:

1. If NMI, or INTR if enabled, occurs while the HLT flag is set, the CPU logic assumes the instruction immediately following the interrupt is an HLT. So it places the address of the instruction after that on the stack, which means that upon return from the interrupt, the instruction immediately following the interrupt occurrence is skipped over.

2. If FLUSH # is asserted while the HLT flag is set, the CPU flushes the L1 cache and then returns to the HLT state. If the CPU is extracted from the HLT state by NMI or INTR, as in 1), the CPU logic assumes that the current CS:EIP points to an HLT instruction, and pushes the address of the *next* instruction on the stack, so the instruction immediately following the FLUSH# assertion is skipped over.

3. If RSM is executed while the HLT flag is set, again the CPU logic assumes that the CPU must have been interrupted (by SMI, in this case) while in the HLT state. Normally, RSM would cause the CPU to branch back to the instruction that was aborted when entering SMM. But in this case, the CPU branches to the address of the *next* instruction minus one byte. If the aborted instruction is one byte long, this is fine. If it is longer, the CPU executes effectively a random opcode: the last byte of the aborted instruction is interpreted as the first byte of the next instruction.

IMPLICATION: In cases 1 and 2, skipping an instruction can have no noticeable effect, or it could cause some obvious error condition signaled by a system exception, or it could cause an error which is not easily detected. In case 3, executing a random opcode is most likely to cause a system exception like #6 (invalid opcode), but it could cause either of the other results as with cases 1 and 2. Case 2 can also cause an indefinite CPU hang, if the problem occurred when INTR was disabled. However, in order to encounter any of these problems, the system has to continue on with program execution after servicing the MCE. Since the MCE is an abort type exception, the handler for it cannot rely on a valid return address. Also MCE usually signals a serious system reliability problem. For both these reasons, the usual protocol is to require a system reset to terminate the MCE handler. If this usual protocol is followed successfully, it will clear the HLT flag and thus always prevent the above problems. However, there is an additional complication: the cases 1, 2 and 3 above can occur *inside* the MCE handler, possibly preventing its completion.

WORKAROUND: The problems caused by this erratum will be prevented if the Machine Check Exception handler (if invoked) always forces a CPU RESET or INIT (which it should do anyway, for reasons given above). Since the problems can occur *inside* the MCE handler, the IF should be left zero to prevent INTR from interrupting. Also, NMI, SMI and FLUSH could be blocked inside the MCE handler. The most secure strategy is to force INIT immediately upon entrance to the MCE handler.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

53. FBSTP Stores BCD Operand Incorrectly If Address Wrap & FPU Error Both Occur

PROBLEM: This erratum occurs only if a program does all of the following:

1. The program uses 16 bit addressing inside a USE32 segment (requiring the 67H addressing override prefix) in order to wrap addresses at offsets above 64K back to the bottom of the segment.
2. The 10 byte BCD operand written to memory by the FBSTP instruction must actually straddle the 64K boundary. If all 10 bytes are either above or below 64K, the wrap works normally.
3. The FBSTP instruction whose operand straddles the boundary must also generate an FPU exception. (e.g. Overflow if the operand is too big, or Precision if the operand must be rounded, to fit the BCD format.)

The result is that some of the 10 bytes of the stored BCD number will be located incorrectly if there is an FPU exception. They will be nearby, in the same segment, so no protection violation occurs from this erratum.

The erratum is caused by the fact that when an FPU exception occurs due to FBSTP, a different internal logic sequence is used by the CPU, which incidentally sends the bytes to memory in different groupings. Normally this does not affect the result, but when address wrap occurs in the middle of the operand, the different groupings can cause different destination addresses to be calculated for some bytes.

IMPLICATION: Code which relies on this address wrap with a straddled FBSTP operand may not store the operand correctly if FBSTP also generates an FPU exception. Intel recommends not to straddle segment or addressing boundaries with operands for several reasons, including (see the *Intel Architecture Software Developer's Manual*, Volume 3, Chapter 17) the chance of losing data if a memory fault interrupts an access to the operand. Also there is variation between generations of Intel processors in how straddled operands are handled.

WORKAROUND: Use one of the following:

Do not use 16 bit addressing to cause wraps at 64K inside a USE32 segment.

Follow Intel's recommendation and do not straddle an addressing boundary with an operand.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

54. V86 Interrupt Routine at Illegal Privelege Level Can Cause Spurious Pushes to Stack

PROBLEM: By architectural definition, V86 mode interrupts must be executed at privilege level 0. If the target CPL (Current Privilege Level) in the interrupt gate in the IDT (Interrupt Descriptor Table) and the DPL (Descriptor Privilege Level) of the selected code segment are not 0 when an interrupt occurs in V86 mode, then interrupt 13 (GP fault) occurs. This is described in the *Intel Architecture Software Developer's Manual*, Volume 3, Section 15.3. The architectural definition says that execution transfers to the GP fault routine (which must be at level 0) with nothing done at the privilege level (call it level N) where the interrupt service

routine is illegally located. In fact (this erratum) the Pentium® Processor incorrectly pushes the segment registers GS and FS on the stack at level N, before correctly transferring to the GP fault routine at level 0 (and pushing GS and FS again, along with all the rest that's specified for a V86 interrupt).

IMPLICATION: When this erratum occurs, it will place a few additional bytes on the stack at the level (1, 2 or 3) where the interrupt service routine is illegally located. If the stack is full or does not exist, the erratum will cause an unexpected exception. But this problem will have to be fixed during the development process for a V86 mode OS or application, because otherwise the interrupt service routine can never be accessed by V86 code. Thus this erratum can only be seen during the debugging process, and only if the software violates V86 specifications.

WORKAROUND: Place all code for V86 mode interrupt service routines at privilege level 0, per specification.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

55. Corrupted HLT Flag Can Cause Skipped or Incorrect Instruction, or CPU Hang

PROBLEM: The Pentium processor sets an internal HLT flag while in the HLT state. There are some specific instances where this HLT flag can be incorrectly set when the CPU is not in the HLT state.

1. A POP SS which generates a data breakpoint, and is immediately followed by a HLT. Any interrupt which is pending during an instruction which changes the SS, is delayed until after the next instruction (to allow atomic modification of SS:ESP). In this case, the breakpoint is therefore correctly delayed until after the HLT instruction is executed. The processor waits until after the HLT cycle to honor the breakpoint, but in this case when the processors branches to the interrupt 1 handler, it fails to clear the HLT flag. The interrupt 1 handler will return to the instruction following the HLT, and execution will proceed, but with the HLT flag erroneously set.

2. A code breakpoint is placed on a HLT instruction, and an SMI# occurs while processor is in the HLT state (after servicing the code breakpoint). The SMI handler usually chooses to RSM to the HLT instruction, rather than the next one, in order to be transparent to the rest of the system. In this case, on returning from the SMI# handler, the code breakpoint is typically re-triggered (SMI# handler does not typically set the RF flag in the EFLAGS image in the SMM save area). The processor branches to the interrupt 1 handler again, but without clearing the HLT flag. The interrupt 1 handler will return to the instruction following the HLT, and execution will proceed, but with the HLT flag erroneously set.

3. A machine check exception just before, or during, a HLT instruction can leave the HLT flag erroneously set. This is described in detail in erratum #52: *Machine Check Exception Pending, then HLT, Can Cause Skipped or Incorrect Instruction, or CPU Hang.*

IMPLICATION: For cases 1 and 2, the CPU will proceed with the HLT flag erroneously set. The following problematic conditions may then occur.

   a. If NMI, or INTR if enabled, occurs while the HLT flag is set, the CPU logic assumes the instruction immediately following the interrupt is a HLT. It therefore places the address of the instruction after that on the stack, which means that upon return from the interrupt, the instruction immediately following the interrupt occurrence is skipped over.

   b. If FLUSH # is asserted while the HLT flag is set, the CPU flushes the L1 cache and then incorrectly returns to the HLT state, which will hang the system if INTR is blocked (IF = 0) and NMI does not occur. If the CPU is extracted from the HLT state by NMI or INTR, as in a), the CPU logic assumes that the current CS:EIP points to an HLT instruction, and pushes the address of the *next* instruction on the stack, so the instruction immediately following the FLUSH# assertion is skipped over.

   c. If RSM is executed while the HLT flag is set, again the CPU logic assumes that the CPU must have been interrupted (by SMI#, in this case) while in the HLT state. Normally, RSM would cause the CPU to branch back to the instruction that was aborted when entering SMM. But in

this case, the CPU branches to the address of the *next* instruction minus one byte. If the aborted instruction is one byte long, this is fine. If it is longer, the CPU executes effectively a random opcode: the last byte of the aborted instruction is interpreted as the first byte of the next instruction.

d. If STPCLK# is asserted to the CPU while the HLT flag is incorrectly set, the CPU will hang such that a CPU reset is required to continue execution.

Cases 1 and 2 of this erratum occur only during code development work, and only with the unusual combination of data breakpoint triggered by POP SS followed by HLT or code breakpoint on HLT followed by SMI#.

WORKAROUND:

CASE 1: Avoid following POP SS with a HLT instruction. POP SS should always be followed by POP ESP anyway, to finish switching stacks without interruption. Following POP SS with HLT instead would normally be a program logic error (the interrupt that breaks the CPU out of HLT will not have a well defined stack to use).

CASE 2: Do not place code breakpoints on HLT instructions. Or: Modify the SMI# handler slightly for debugging purposes by adding instructions to set the RF flag in the EFLAGS image in the SMM save area.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

56. Benign Exceptions Can Erroneously Cause Double Fault

PROBLEM: The double-fault counter can be incorrectly incremented in the following cases:

CASE 1: An instruction generates a benign exception (for example, a FP instruction generates an INT 7) and this instruction causes a segment limit violation (or is paired with a v-pipe instruction which causes a segment limit violation)

CASE 2: A machine check exception (INT 18) is generated.

The initial benign exception will be serviced properly. However, if while trying to begin execution of the benign exception handler, the processor gets an additional contributory exception, the processor will trigger a double fault (and start to service the double fault handler) instead of servicing the new contributory fault. (See Table 5-3 in the *Intel Architecture Software Developer's Manual*, Volume 3 for a complete list of benign/contributory exceptions).

IMPLICATION: Contributory exceptions generated while servicing benign exceptions can erroneously cause the processor to execute the double fault handler instead of the contributory exception handler.

WORKAROUND: Use benign exception handlers that do not generate additional exceptions. Operating systems designed such that benign exception handlers do not generate additional exceptions will be immune to this erratum. In general, most operating system exception handlers are architected accordingly.

Note that Case 2 does not apply to OverDrive processors.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

57. Double Fault Counter May Not Increment Correctly

PROBLEM: In some cases a double fault exception is not generated when it should have been because the internal double fault counter does not correctly get incremented.

When the processor encounters a contributory exception while attempting to begin execution of the handler for a prior contributory exception (for example, while fetching the interrupt vector from the IDT or accessing

the GDT/LDT) it should signal the double fault exception. Due to this erratum, however, the CPU will incorrectly service the new exception instead of going to the double fault handler.

In addition, if the first contributory fault is the result of an instruction executed in the v-pipe, a second contributory fault will cause the processor to push an incorrect EIP onto the stack before entering the second exception handler. Upon completion of the second exception handler, this incorrect EIP gets popped from the stack and the processor resumes execution from the wrong address.

IMPLICATION: The processor could incorrectly service a second contributory fault instead of going to the double fault handler. The resulting system behavior will be operating system dependent. Additionally, an inconsistent EIP may be pushed on to the stack.

Robust operating systems should be immune to this erratum because their exception handlers are designed such that they do not generate additional contributory exceptions. This erratum is only of concern during operating system development and debug.

WORKAROUND: Use contributory exception handlers that do not generate additional contributory exceptions. Operating systems which are designed such that their contributory exception handlers do not generate additional contributory exceptions will not be affected by this erratum. In general, most operating system exception handlers are architected accordingly.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

58. Some Input Pins May Float High Erroneously When Core V_{cc} Powers Up After I/O V_{cc} (Mobile CPU)

PROBLEM: Unused input signals are typically tied off (either high or low). Low inputs can be provided with a hard V_{SS} strap or a pulldown resistor. If any of the input pins AHOLD, KEN#, WB/WT#, NA#, INV, BRDY#, or EWBE# are not driven by system logic, and are tied to ground via a weak pulldown resistor (i.e. >2KOhms), and CPU I/O power supply (V_{CC3}) ramps before CPU core power supply (V_{CC2}), these input pins may float high and be erroneously latched high by the processor during boot. The effect of this erratum depends on the usage of each pin. For example, if EWBE# gets latched high, the processor may hang indefinitely.

IMPLICATION: The input pins AHOLD, KEN#, WB/WT#, NA#, INV, BRDY#, or EWBE# may register a false start up state. In some cases, the processor may erroneously hang while waiting for an input response. For example, the EWBE# being sampled high may cause the system to hang while waiting for the processor to sample EWBE# low.

WORKAROUND: If the signal is not driven by system logic and is pulled low, a pulldown resistor of 2K Ohms or less should be used to guarantee logic level zero.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

59. Short Form of MOV EAX/ AX/ AL May Not Pair

PROBLEM: The MOV data instruction forms (excluding MOV using Control, Debug or Segment registers) are intended to be pairable, unless there is a register dependency between the two instructions considered for pairing. (e.g. MOV EAX, mem1 followed by MOV mem2, EAX: here the 2nd instruction cannot be completed until after the first has put the new value in EAX.) This pairing for MOV data is documented by the UV symbol in the Pairing column in the table of Pentium processor instruction timings in the *Optimizations for Intel's 32-Bit Processors* application note (Order # 243195). This erratum is that the instruction unit under some conditions fails to pair the special short forms of MOV mem, EAX /AX /AL, when no register dependency exists.

The Intel Architecture includes special instructions to MOV EAX /AX /AL to a memory offset (opcodes 0A2H & 0A3H). These instructions don't have a MOD/RM byte (and so are shortened by one byte). Instead, the opcode is followed immediately by 1/2/4 bytes giving the memory offset (displacement). This erratum occurs specifically when a MOV mem, EAX /AX /AL instruction using opcode 0A2H or 0A3H is followed by an instruction that uses the EAX /AX /AL register as a source (register source, or as base or index for the address of a memory source) or a destination register. Then the instruction unit detects a (false) dependency and it doesn't allow pairing. For example, the following two instructions are not paired:

```
A340000000   MOV DWORD PTR 40H, EAX ;  memory DS:[40H] <- EAX  [goes into u-pipe ]
A160000000   MOV EAX, DWORD PTR 60H ;  EAX <- memory DS:[60H]  [does NOT go into
v-pipe]
```

IMPLICATION: The only result of this erratum is a very small performance impact due to the non-pairing of the above instructions under the specified conditions. The impact was evaluated for SPECint92* and SPECfp92* and was estimated to be much smaller than run-to-run measurement variations.

WORKAROUND: For the Pentium processor, use the normal MOV instructions (with the normal MOD/RM byte) for EAX /AX /AL instead of the short forms, when writing optimizing compilers and assemblers or hand assembling code for maximum speed. However, as documented above, the performance improvement from avoiding this erratum will be quite small for most programs.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

60. Turning Off Paging May Result In Prefetch To Random Location

PROBLEM: When paging is turned off a small window exists where the BTB has not been flushed and a speculative prefetch to a random location may be performed. The *Intel Architecture Software Developer's Manual,* Volume 3, Section 8.8.2, lists a sequence of nine steps for switching from protected mode to real-address mode. Listed here is step 1.

1. If paging is enabled, perform the following sequence:
   - Transfer control to linear addresses which have an identity mapping (i.e., linear addresses equal physical addresses). Ensure the GDT and IDT are identity mapped.
   - Clear the PG bit in the CR0 register.
   - Move zero into the CR3 register to flush the TLB.

With paging enabled, linear addresses are mapped to physical addresses using the paging system. In step a above the executing code transfers control to code located where the linear addresses are mapped directly to physical addresses. Step b turns off paging followed by step c which writes zero to CR3 which flushes the TLB (and BTB). A small window exists (after clearing the PG bit and before zeroing CR3) where the BTB has not been flushed, and a BTB hit may cause a prefetch to an unintended physical address.

IMPLICATION: A prefetch to an unintended physical address could potentially cause a problem if this prefetch was to a memory mapped I/O address. If reading a memory mapped I/O address changes the state of a memory mapped I/O device, this unintended access may cause a system problem.

WORKAROUND: Flush the BTB just before turning paging off. This can be done by reading the contents of CR3 and writing it back to CR3 prior to clearing the PG bit in CR0.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

61. STPCLK# or FLUSH# After STI

PROBLEM: The STI specification says that external interrupts are enabled at the end of the next instruction after STI. However external interrupts may be enabled before the next instruction is executed following STI if a STPCLK# or FLUSH# is asserted and serviced before the instruction boundary of this next instruction.

IMPLICATION: External interrupts assumed blocked until after the instruction following STI may be recognized before this instruction executes. No operating system is known by Intel to be affected.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

62. REP String Instruction Not Interruptable by STPCLK#

PROBLEM: The *Intel Architecture Software Developer's Manual,* Volume 2, Chapter 3 under the REP string instruction, states that any pending interrupts are acknowledged during a string instruction. On the Pentium processor there is one exception. STPCLK# is not able to interrupt a REP string instruction. It is only recognized on an instruction boundary (as stated in Volume 1, Section 21.1.36). However, if any other interrupt is recognized during a REP string instruction, this will allow STPCLK# to be serviced before returning to execution of the REP string instruction.

IMPLICATION: A system that uses stop clock frequently can not interrupt the REP string instruction in the middle and must wait until it completes or another interrupt is recognized before STPCLK# is recognized. Note that in standard PC-AT architecture, the real time clock interrupt will interrupt a long string instruction allowing STPCLK# to be recognized.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

63. Single Step May Not be Reported on First Instruction After FLUSH#

PROBLEM: The single step trap should cause an exception to occur upon completion of all instructions. However, in some cases when ITR (bit 9 of TR12) = '1', a single step exception may not be reported for the first instruction following FLUSH#. The Single Step exception will be skipped for this instruction. Note that subsequent single step exceptions will be reported correctly.

IMPLICATION: A single step breakpoint may be missed when a FLUSH# request is presented to the processor. This erratum will only affect software developers while debugging code.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

64. Double Fault May Generate Illegal Bus Cycle

PROBLEM: A double fault condition may generate an illegal bus cycle (a cacheable line-fill with a lock attribute). This scenario is caused by following sequence of events:

1. A contributory fault occurs.

2. The processor begins to service this fault by reading the appropriate trap/interrupt gate from the IDT. However, this gate points to a segment descriptor (in the GDT) whose “accessed” bit is not set.
3. The segment descriptor is modified and marked “not present/not valid” by another processor in the system
4. A locked Read-Modify-Write cycle is generated to update the “accessed” bit.

The erratum condition is encountered if the segment descriptor was modified and marked “not present/not valid” by another processor in the system before the locked read cycle (step #4 above). The processor will begin to execute the locked read. Since the descriptor is marked invalid, the processor should go to the exception handler to service a specific exception and clear the bus-lock (through a write operation). However, since a contributory fault has already occurred, the processor will interpret this condition as a double fault. The double fault logic incorrectly generates a cacheable line-fill with a lock attribute.

IMPLICATION: This erratum can only occur in DP and MP systems.

Cacheable line-fills with a lock attribute are “illegal” bus cycles. Exact operation under this condition is chipset dependent. It may cause the system to hang.

Note that this erratum will only occur in the case of a double fault, which are rare events for well architected operating systems. Also, the double fault condition is not generally recoverable, implying that the system will need to be rebooted anyway.

Finally, this erratum can only happen on the first pass through the interrupt handler. After that, the “accessed” bit of the code descriptor will be set, eliminating a prerequisite for occurrence of this erratum.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

65. TRST# Not Asynchronous

PROBLEM: TRST# is not an asynchronous input as specified in Section 5.1.67 of the *Pentium® Processor Family Developer’s Manual*.

IMPLICATION: TRST# will not be recognized in cases where it does not overlap a rising TCK# clock edge. This violates the IEEE 1149.1 specification on Boundary Scan.

WORKAROUND: TRST# should be asserted for a minimum of two TCK periods to ensure recognition by the processor.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

66. STPCLK# on RSM to HLT Causes Non-Standard Behavior

PROBLEM: This problem will occur if STPCLK# is asserted during the execution of an RSM instruction which is returning from SMM to a HALT instruction (Auto HALT restart must be enabled for this to happen). The RSM instruction will be completed, and then the CPU correctly issues, in response to the STPCLK# assertion, a Stop Grant special cycle, and goes into the Stop Grant state. However, following this, behavior occurs which deviates from the CPU specifications in one of two ways, depending on whether STPCLK# is de-asserted before any (enabled) external interrupt occurs (Case 1), or an (enabled) external interrupt occurs while STPCLK# is still active (Case 2).

CASE 1: After STPCLK# is de-asserted, no HALT special cycle is issued, and the CPU effectively stays in the Stop Grant state until an external interrupt is asserted (to which the CPU responds normally). However, a HALT cycle **should** be issued when STPCLK# is de-asserted, because it is stated that a HALT cycle will be issued upon an RSM to the HALT state.

CASE 2: When the (enabled) external interrupt is asserted while STPCLK# is still active, the CPU should remain in the Stop Grant state. But when the conditions have been met for this erratum, the CPU comes out of the Stop Grant state and starts the internal interrupt service process. This includes issuing the interrupt acknowledge cycles, reading the selected entry from the interrupt descriptor table, and fetching the first instruction of the requested interrupt service routine (I.S.R.). However, before the CPU executes that first instruction, STPCLK# is recognized again, execution halts, and a Stop Grant cycle is issued. The erratum condition is cleared by one of the steps which the CPU performs to prepare for the I.S.R., so any further interrupts (while STPCLK# remains asserted) will not remove the CPU from the Stop Grant state. When STPCLK# is de-asserted, the CPU begins executing the requested I.S.R.

IMPLICATION:

CASE 1: The absence of the usual HALT special cycle upon a RSM to a HLT instruction in this rare case should have no impact, unless the system is looking for the HALT cycle after RSM and would normally make some response to it. The system will have received the HALT cycle upon initial entry to the HALT state. To expect another HALT cycle after RSM, the system would have to be tracking the fact that the SMI occurred during a HLT.

CASE 2: This case of the erratum means that some cycles preparatory to executing the I.S.R. are issued when the interrupt is received, rather than waiting until after STPCLK# is de-asserted. Also, an extra Stop Grant cycle is issued just after these premature cycles. However, all of the I.S.R. itself is executed at the correct time. This difference in the bus cycles has no known system implications.

WORKAROUND: None required for any known implementations.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

67. Code Cache Dump May Cause Wrong IERR#

PROBLEM: When using the test registers to read a cache line that is not initialized, the data array may indicate a wrong parity, which may cause IERR# to be asserted. It may also cause a shutdown.

IMPLICATION: A code cache dump through test registers may cause a parity check when reading an uninitialized cache entry, resulting in a shutdown.

WORKAROUND: Set TR[1] to 1 to ignore IERR#, so that shutdown during a code cache dump can be avoided, or ensure that all cache lines have been initialized prior to a code cache dump.

68. Asserting TRST# Pin or Issuing JTAG Instructions Does not Exit TAP Hi-Z State

PROBLEM: The *Pentium® Processor Family Developer's Manual,* Section 11.3.2.1 states that the TAP Hi-Z state can be terminated by resetting the TAP with the TRST# pin, by issuing another TAP instruction, or by entering the Test_Logic_Reset state. However, the indication that the processor has entered the TAP Hi-Z state is maintained until the next RESET. Therefore by using the above methods alone, the TAP Hi-Z state can not be terminated.

IMPLICATION: When the TAP Hi-Z instruction is enabled and executed, the processor may not terminate the Hi-Z state.

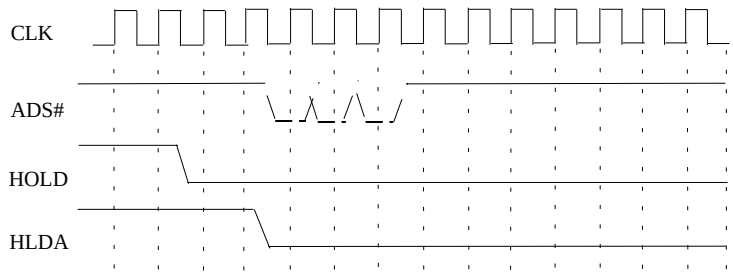
WORKAROUND: To exit TAP Hi-Z state, in addition to the methods described above, the processor needs to be RESET as well.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

69. ADS# May be Delayed After HLDA Deassertion

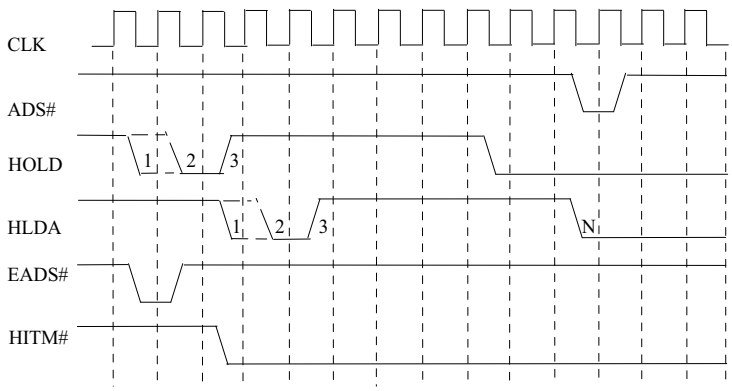
PROBLEM: The Pentium processor typically starts a pending bus cycle on the same clock that HLDA is deasserted and the Pentium processor with MMX technology typically starts the cycle one clock after HLDA is deasserted. However, in both processors it may be delayed by as many as two clocks. See the diagram below:

Pending Cycle May Be Delayed



In two cases, for example, if HOLD is deasserted for one clock (i.e., clock 2) or two clocks (i.e., clocks 1 & 2) and then reasserted, the window may not be large enough to start a pending snoop writeback cycle. The writeback cycle may be delayed until the HLDA is deasserted again (i.e., clock N). See diagram below.

Snoop Writeback Cycle Delayed



IMPLICATION: If the system expects a cycle, for example a writeback cycle, and depends on this cycle to commence within the HLDA deassertion window, then the system may not complete the handshake and cause a hang.

WORKAROUND:

1. Deassert HOLD for at least 3 clocks (i.e., clocks 1, 2, and 3 shown in figure) before reasserting HOLD again. This ensures that the Pentium processor initiates any pending cycles before reasserting HLDA.
2. If the system is waiting for the snoop writeback cycle to commence, for instance if HITM# is asserted, the system should wait for the ADS# before reasserting HOLD.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

70. Stack Underflow in IRET Gives #GP, Not #SS

PROBLEM: The general Intel architecture rule about accessing the stack beyond either its top or bottom is that the stack fault error (#SS) will be generated. However, if during the execution of the IRET instruction there are insufficient bytes for an interlevel IRET to pop from the stack (stack underflow), the general protection (#GP) fault is generated instead of #SS.

IMPLICATIONS: This can only occur if the stack has been modified since the interrupt stored its return address, flags etc. such that there is no longer room on the stack for all of the stored information when IRET tries to access it. This would constitute a serious programming error that would cause problems more obvious than this erratum, and would normally be corrected during debugging. If this erratum did occur during regular execution of a program, the normal O/S response to a task causing either a #GP or #SS exception is to terminate the task, and so this erratum (#GP instead of #SS) would normally have no effect. If however the O/S is to be programmed to try to correct #GP and #SS problems and allow the task to continue execution, the workaround should be used.

WORKAROUND: In order for the O/S code to correctly analyze this case of stack limit violation, the #GP code must include a test for stack underflow when #GP occurs during the IRET instruction.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

71. Performance Monitoring Pins PM[1:0] May Count The Events Incorrectly

PROBLEM: The performance monitoring pins PM[1:0] can be used to indicate externally the status of event counters CTR1 and CTR0. While events are generated at the rate of the CPU clock, the PM[1:0] pins toggle at the rate of the I/O bus clock. However in some cases, the PM[1:0] pins may toggle twice when the event counters increment twice in one I/O clock, while in some cases, the PM[1:0] pins may toggle only once even when the event counters increment twice in two consecutive I/O clocks.

IMPLICATION: The performance monitoring pins PM[1:0] may not be relied upon to reflect the correct number of events that have occurred.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

72. BIST Is Disabled

PROBLEM: Built in Self Test (BIST) is disabled and will not be run when the proper sequence is initiated. BIST can be initiated in two different ways - by asserting INIT when RESET transitions from high to low or by using the RUNBIST command through the TAP port. In both cases, BIST will not be run and the value returned in either EAX or the Runbist register will be zero indicating BIST has passed.

IMPLICATION: If BIST is initiated, the system should not depend on the result since the processor always reports that BIST has completed successfully. BIST is normally only run as part of a manufacturing test or as part of a power on test.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

73. Branch Trace Messages May Cause System Hang

PROBLEM: In a system with branch trace messages enabled, certain semaphore signaling sequences may cause the system to hang. Branch trace messages have the highest priority bus cycle in the Pentium processor with MMX technology, unlike previous Pentium processors, and take precedence over any other write cycle. A sequence of code where the processor writes to another processor or a controller, and then locks into a tight loop while waiting for the other processor or the controller to respond to the write, is susceptible to a hang, if branch trace messages are enabled. The problem is that the unending branch trace messages from the loop take priority over the previous write cycle. The write cycle never occurs and the other processor or the controller never responds. However, the processor will be pulled out of the hanging situation if an interrupt occurs.

IMPLICATION: This erratum only affects operation of the processor during instruction execution tracing which is normally only done during code development and debug. In addition, this erratum would typically only occur in an MP system, with short code sequences used for message passing. Also since interrupts pull the processor out of the hanging condition and they normally occur frequently, there should not be any noticeable system hang.

WORKAROUND: Disable the branch trace message feature by setting TR12 bit 1 to 0 (the default is disabled).

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

74. Enabling RDPMC in CR4 And Also Using SMM May Cause Shutdown

PROBLEM: The Pentium processor with MMX technology implements a new instruction RDPMC (read performance monitoring counters), and a new control bit, CR4.PCE in the CR4 register. When CR4.PCE is one, the RDPMC instruction is not O/S protected and may be executed at the O/S level (level 0), or at the application level (level 1, 2 or 3) . When the processor enters SMM mode, it dumps the contents of all the control registers, including CR4, to the SMM dump area. Upon leaving SMM mode, the RSM instruction restores the control registers from the SMM dump area. The RSM instruction checks the dumped CR4 value at the SMM dump area before it loads the data back to CR4 to ensure the reserved bits are all zero. The PCE bit incorrectly gets flagged as reserved and the processor enters into shutdown mode.

IMPLICATION: This erratum only affects systems that use SMM and also enable the RDPMC instruction to be executed at all privilege levels by setting the CR4.PCE bit. If the CR4.PCE bit is set and SMM mode is used, executing the RSM instruction will cause the processor to enter shutdown mode.

WORKAROUND: Disable SMM when CR4.PCE is set.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

75. Event Monitor Counting Discrepancies (Fix)

PROBLEM: The Pentium processor with MMX technology added several performance monitoring events to those defined in the Pentium processor (75/90/100/120/133/166/200). There are several conditions where the counters do not operate as specified.

The “Writes to non-cacheable memory” (event 101110) event counter counts the number of writes to non-cacheable memory including non-cacheable writes caused by MMX™ instructions. In some cases the counter fails to get incremented for a non-cacheable memory write caused by an MMX instruction.

The “Stall on MMX instruction write to an E or M state line” (event 111011) event counter counts the number of clocks the processor is stalled on a data memory write hit to an E or M state line in the internal

data cache caused by a MMX instruction while either the write buffers are not empty or EWBE# is not asserted. However, it does not count stalls while the write buffers are not empty, it only counts the number of clocks stalled while EWBE# is not asserted.

IMPLICATION: The event monitor counters report an inaccurate count for certain events.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

76. Event Monitor Counting Discrepancies (NoFix)

PROBLEM: The Pentium processor with MMX technology added several performance monitoring events to those defined in the Pentium processor (75/90/100/120/133/166/200). There are several conditions where the counters do not operate as specified.

The “MMX instruction data read misses” (event 110001) and “MMX instruction data write misses” (event 110100) event counters get incorrectly incremented twice if the access to the cache is misaligned. The “Pipeline stalled waiting for MMX instruction data memory read” (event 110110) event counter incorrectly counts a misaligned access as 2 clocks instead of 3 clocks unless it misses the TLB.

The “MMX instruction multiply unit interlock” (event 111011) event counter counts the number of clocks the pipe is stalled because the destination of a previous MMX multiply instruction is not ready. However, if there is a multiply instruction followed by a branch instruction followed by a dependent multiply instruction, the counter incorrectly gets incremented when the branch is taken.

IMPLICATION: The event monitor counters report an inaccurate count for certain events.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

77. INVD May Leave Valid Entries In The Cache Due To Snoop Interaction

PROBLEM: If the processor is snooped (external snoop or private snoop in a DP implementation) during execution of the INVD instruction, some cache entries may be left in a valid state.

IMPLICATION: There are no known system implications. INVD is a privileged instruction, and hence can not be used by application software in a protected mode O/S. In any case, executing INVD on a writeback processor will not guarantee cache coherency. If the O/S or the BIOS wishes to invalidate the cache, it should use the WBINVD instruction which is not affected by this erratum.

WORKAROUND: If the INVD instruction must be used in a cache test or some other (BIOS) code, set CD and NW in CR0 to ‘1’ before executing the INVD instruction, so that a snoop cycle has no effect on the state of the cache. After the INVD instruction, re-enable the cache by setting CD and NW to “0”. Note that this workaround does not apply to DP systems since external snoops can not be inhibited in a DP system.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

78. TLB Update Is Blocked After A Specific Sequence Of Events With A Misaligned Descriptor

PROBLEM: An obscure sequence of events may cause the TLB replacement mechanism to fail. This specific sequence must contain **all** of the following:

1. A specific setup of the data TLB: all 64 entries must be valid and one entry must contain the page where the IDT is located.
2. A REP-MOVS accesses a string that is at least 62-pages long.
3. A MOVS results in a GP fault in the 62nd page.
4. A gate in the IDT points to a descriptor and the descriptor is misaligned and crosses a page boundary.
5. The descriptor causes a TLB miss.

When the TLB miss discussed in condition 5 above occurs, the processor starts a split locked read-modify-write sequence to update the descriptor access or busy bit. During this split locked cycle, the address of the low bytes of the descriptor is loaded into a slot in the TLB. The address of the high bytes of the descriptor is then put into the same slot of the TLB causing the address of the low bytes to be overwritten (this is caused by conditions 1-3 above). The address of the low bytes of the descriptor then needs to be re-read from memory. However, since the bus is now locked, this cannot occur and the processor hangs waiting for the sequence to complete.

IMPLICATION: If all of the above conditions occur, the processor may hang.

WORKAROUND: Ensure that the base address of the GDT or LDT is aligned. This will prevent the split locked cycle from occurring due to the misaligned descriptor. This is already recommended in the *Intel Architecture Software Developer's Manual,* Volume 3, Section 3.5.1 for performance reasons.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

1DP. Problem with External Snooping while Two Cycles Are Pending on the Bus

PROBLEM: In a dual processor system, the following sequence of events can cause the processors to lock up:

1. There are two cycles pending on the bus, one from each processor. In this case the first cycle is from the C and the second from CM; therefore, C is the LRM and CM is the MRM.
2. AHOLD is asserted and then an external snoop occurs (EADS# is asserted) causing a hit to a modified line in the CM. Since the CM is the MRM, it does not give an indication to the C that it has been hit.
3. Now a BOFF# is asserted and backs off the pending cycles. Once BOFF# is released, the C becomes the MRM in order to maintain cycle order.
4. C now owns the bus but cannot run its cycle because AHOLD is still active. Since C is not aware that CM has been hit by an external snoop, C is not willing to give up the bus to the CM and thus prevents the CM from performing a writeback. Since the C will not give up the bus to the CM and cannot run its own cycle, the system hangs.

IMPLICATION: If each processor in a dual processor system has a cycle pending on the bus and an external snoop results in a hit to a modified line, the processors may lock up.

WORKAROUND:

1. Disable pipelining.
2. Deassert AHOLD no earlier than one clock after BOFF# has been deasserted. Note that if this workaround is used, the system will continue to run but a re-ordering of cycles will occur. The C will run its cycle first (rather than the writeback occurring first), and then grant the bus to the CM to complete its writeback cycle and then its outstanding cycle.

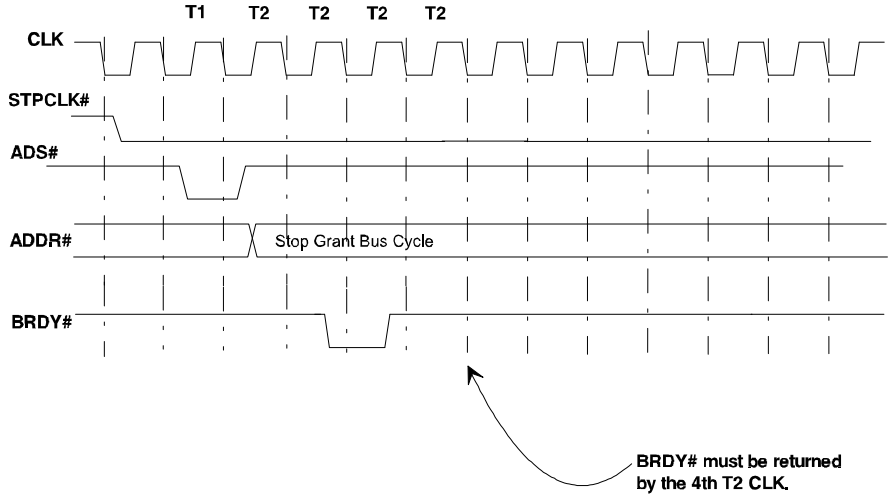
Designs based on the 82430NX PCIset and other chip sets which do not generate an external snoop when two cycles are pending on the bus are not affected.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

2DP. STPCLK# Assertion and the Stop Grant Bus Cycle

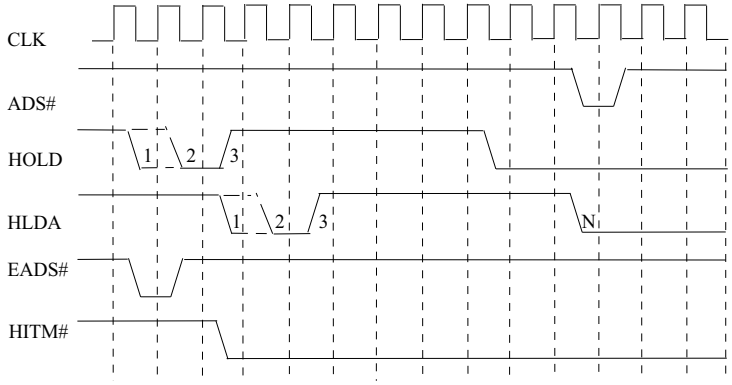
PROBLEM: If a STPCLK# interrupt occurs and BRDY# is not asserted within 4 CLKs following the Stop Grant bus cycle, then the processor which ran the Stop Grant bus cycle may hang.

IMPLICATION: This problem occurs only in dual processor systems and will cause one of the processors to hang.



In two cases, for example, if HOLD is deasserted for one clock (i.e., clock 2) or two clocks (i.e., clocks 1 & 2) and then reasserted, the window may not be large enough to start a pending snoop writeback cycle. The writeback cycle may be delayed until the HLDA is deasserted again (i.e., clock N). See diagram below.

Snoop Writeback Cycle Delayed



WORKAROUND:

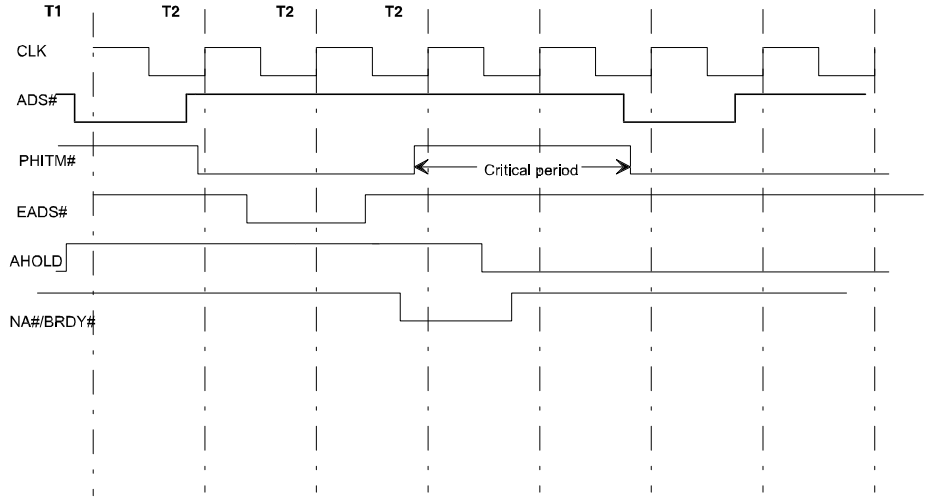
1. Do not assert STPCLK#.
2. Ensure that BRDY# is asserted within 4 CLKs after the Stop Grant bus cycle has begun.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

3DP. External Snooping with AHOLD Asserted May Cause Processor to Hang

PROBLEM: The following sequence of events may cause one of the processors in a dual processing system to hang:

1. The MRM (Most Recent Bus Master, this could be the C or CM processor) issues an ADS# of a memory-write (or memory read) cycle and an AHOLD assertion follows.
2. This ADS# causes an automatic snoop by the LRM (Least Recent Bus Master) and hits a modified line in the LRM causing PHITM#/PHIT# to be asserted.
3. After PHITM# is asserted, EADS# is asserted by the system and does **not** hit a modified line. This causes the LRM to deassert PHITM# for one or two clocks, and then assert PHITM# again.
4. One or two clocks after EADS# is deasserted, AHOLD is deasserted.
5. BRDY# or NA# is asserted within two clocks after EADS# is deasserted. With a BRDY# or NA# assertion, the MRM samples the PHITM# pin before driving the next cycle on the bus. If the MRM samples PHITM# high, during the "critical period" (shown in the figure below, the critical period is defined as the two clock periods after EADS# is sampled active.), the MRM will incorrectly issue the ADS# of its memory-write cycle again before surrendering the bus to the LRM to do its writeback.
6. After the memory-write cycle is complete, the LRM performs its writeback.
7. The MRM then re-issues the memory write-cycle again. This cycle, now being issued for the third time, causes an internal hangup in the MRM.

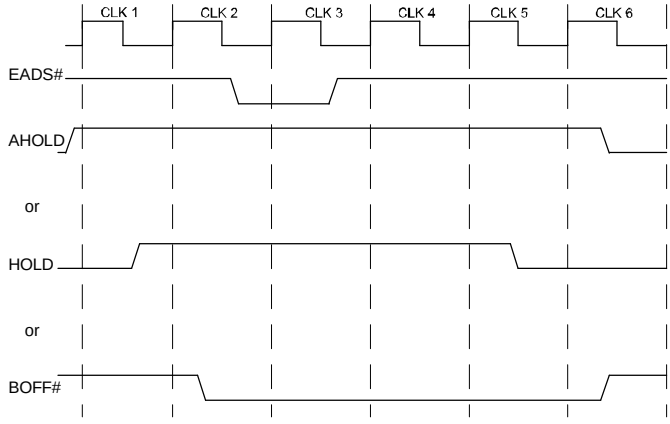


IMPLICATION: If a memory write or memory read cycle is pending on the bus and an external snoop occurs, one of the processors may hang.

WORKAROUND: Use one of the following:

1. Ensure that NA#/BRDY# is asserted before, or after the "critical period". (the critical period is the two CLK period after EADS# is sampled active.)
2. Ensure that AHOLD is held high for a minimum of three clocks after EADS# has been sampled active. See the following figure.
3. Drive HOLD for two clocks after the EADS# was sampled active. Removal of HOLD can be done without regard to the HLDA signal. See the following figure.
4. Drive BOFF# for 3 clocks after the EADS# was sampled active. See the following figure.

All of the listed workarounds prevent the MRM processor from starting a new cycle, thus preventing the third restart of the pending cycle.



STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

4DP. Address Parity Check Not Supported in Dual Processing Mode

PROBLEM: This is a dual processor erratum: There is a very short setup and hold time for the address bus lines in Dual Processing mode to support the single clock interprocessor snoop response required for cache coherency. In this case the 3ns setup specification is enough for the address to propagate to the cache but it does not allow enough time for the address to propagate to the parity calculation circuits. There is a chance that the APCHK# line will spuriously go active showing a parity error on the address bus.

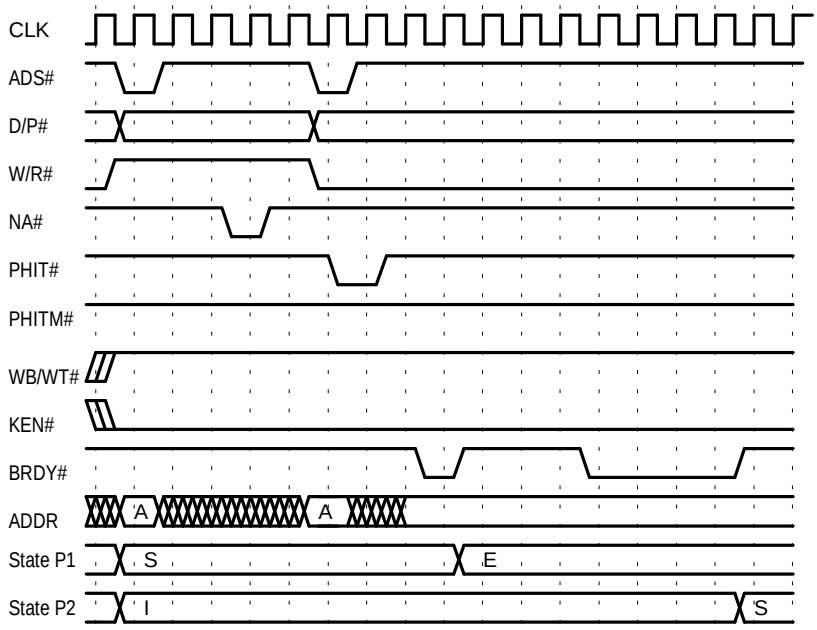
IMPLICATION: Internal measurements of these address signals show that if the 3ns spec is met, the addresses will be latched correctly. Since the parity portion of the circuitry does not meet this timing there is no way to guarantee the APCHK# output is valid. Systems using the APCHK# pin will perform the APCHK# interrupt error routines.

WORKAROUND: Designs based on the 82430NX PCIset and other chip sets which ignore APCHK# in a dual processor environment are therefore not affected. If the address setup (t_{83}) and Maximum Valid Delay (t_6) timings in the *Pentium® Processor Family Developer's Manual,* Chapter 7 are followed, this erratum will not apply to C-step components and future steppings.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section **.**

5DP. Inconsistent Cache State May Result from Interprocessor Pipelined READ into a WRITE

PROBLEM: This is a dual processor erratum: If there is a READ generated by processor 2 pipelined into a WRITE generated by processor 1, and both of these cycles are to the identical address, the cache states for these lines become inconsistent. In this case the WB/WT# pin is driven HIGH, and the KEN# pin is active. Processor 1 will have data that changes from the (S) shared state to the (E) exclusive state, and processor 2 will have data that changes from the (I) invalid state to the (S) shared state. This violates the cache coherency protocol, since any writes to the line that is in the (E) state in processor 1 will not be seen on the bus, resulting in the second processor operating with stale data. This is a symmetrical problem, such that the initiator of the WRITE cycle can either be the primary or dual processor in a two processor configuration. The reason this happens is that each ADS# causes a snoop in the LRM processor, but in this case at the time of the snoop the line state is (S) which will generate the PHIT#, and no PHITM#, the transition to the (E) state has been posted but is not performed until the BRDY# of the WRITE cycle.



IMPLICATION: If the described conditions are met, then an additional write to the same address in processor 1 will cause a cache state transition from (E) to (M) and will not generate a bus cycle. This will mean that the data in processor 2 is stale but it could continue operating with the stale data. For example: When 2 processors are sharing a cached semaphore, and processor 1 is updating the semaphore just as processor 2 is reading the semaphore, then processor 2 would eventually end up with stale data in its cache. Due to the nature of the problem, this could cause a number of unknown system problems and may or may not cause a system to hang.

WORKAROUND: Disable pipelining while using dual processing, this is done by not asserting NA#.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

6DP. Processors Hang During Zero WS, Pipelined Bus Cycles

PROBLEM: In a dual processing system when the following conditions are met: A non-bursted read or write cycle that hits a modified line in the data cache is pipelined into a data read cycle, and this dual processing system is running in zero Wait State mode to the L2 Cache, (2-1-1-1). If the ADS# for the pipelined cycle is in the same clock as the 3 or 4th BRDY# of the bursted data read then the processors will hang.

IMPLICATION: Possible system hang in dual processing operation. System restart/reboot would be required.

WORKAROUND: Use one of the following:

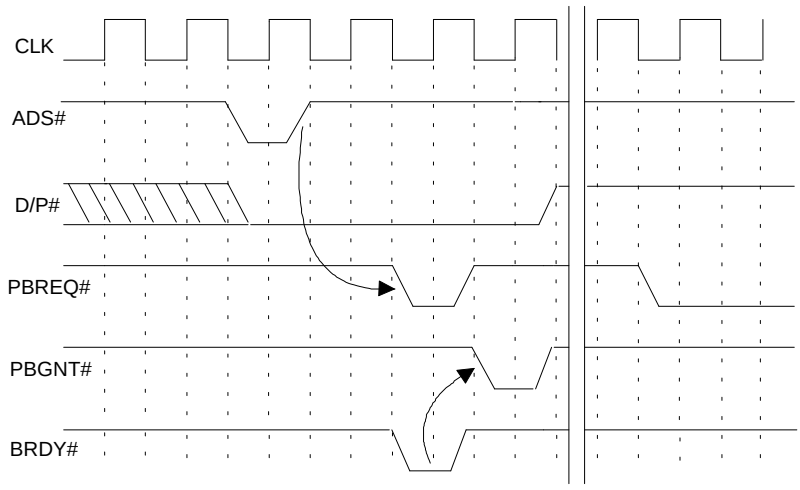
1. Run the dual processing configuration in a non-zero Wait State mode, a (3-1-1-1) burst operation has an anticipated impact of 2-4 percent performance decrease. This is how the 82430NX PCIset operates.
2. Disable pipelining by setting NA# pin high.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

7DP. Bus Lock-up Problem in a Specific Dual Processing Mode Sequence

PROBLEM: In a dual processor system with the CPUs operating in 2/3 (bus/core) Bus Fraction mode, the following sequence of events may cause the system to lock up:

1. The LRM processor is 'spinning' on a semaphore location, with interrupts disabled, waiting for the other CPU to complete its task.
2. The MRM (CPU A) issues a bus cycle by asserting ADS#.
3. The LRM (CPU B) samples the ADS# from the MRM and initiates an internal snoop.
4. Due to an internal circuit problem, the snoop may cause the LRM to assert PBREQ# for one clock erroneously even though it does not have a bus cycle to run. The LRM deasserts PBREQ# in the next clock.
5. The MRM (CPU A) on seeing PBREQ# asserted, grants the bus to the LRM (CPU B) by asserting PBGNT#.
6. Since CPU B actually does not need the bus, it does not run any bus cycle but it continues to own the bus.
7. CPU A asserts PBREQ# to CPU B in order to obtain bus ownership back to run its pending cycle.
8. CPU B, however does not grant the bus back to CPU A since it needs to run a bus cycle before relinquishing the bus ownership. Since interrupts are disabled and the processor is executing in a tight 'spin' loop, it does not have any bus cycles to run and does not relinquish the bus.



IMPLICATION: A system lockup can occur because one CPU requests the bus, while the other CPU does not relinquish bus control. Running Windows NT operating system, it has been observed that when the hang condition occurs, the code inside the Windows NT kernel is always inside a very tight code loop, with

at least one of the processors 'spinning' on a semaphore location, with interrupts disabled, waiting for the other CPU to complete its task.

WORKAROUND: Asserting STPCLK# to the processor that owns the bus will cause the system to come out of the lock up condition. Using a timer, when the PBREQ# signal is seen asserted for a few thousand clocks without the PBGNT# signal asserted, STPCLK# can then be asserted to the processor that owns the bus in order to get it out of the hang condition and resume normal operation. The D/P# pin can be used to tell which processor owns the bus. Alternatively, asserting STPCLK# to both processors will also work.

Although this problem is extremely rare, the failure rate is higher:

1. At lower temperatures, closer to approximately 30° C.
2. With pipelining enabled. Pipelining is disabled by setting NA# pin high.
3. Operating in zero or one Wait State mode.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

8DP. Incorrect Assertion of PHITM# without PHIT#

PROBLEM: In a dual processing environment, assertion of BOFF# in a pipelined bus situation may cause the same cycle to be completed twice on the bus.

This problem occurs in the following case:

1. The Primary processor issues cycle A, either a memory read or write.
2. While cycle A is in progress, the dual processor obtains the bus and issues pipelined memory cycle B, creating an internal self-snoop in the dual processor. Cycle B is of the type that creates a snoop into the data cache (TLB snoop or a code read), and this snoop hits a modified line. As a result, the dual processor has a pending internal writeback cycle.
3. BOFF# is asserted to both processors, backing off cycles A and B.
4. After deassertion of BOFF#, the Primary processor restarts cycle A.
5. The dual processor erroneously asserts PHITM# (without PHIT#) due to the pending internal writeback cycle. This causes the Primary processor to internally back off cycle A, even though there is no hit to a modified line in the dual processor and cycle A completes on the bus externally.
6. The dual processor receives the bus and issues the pending internal writeback.
7. The dual processor restarts cycle B.
8. The Primary processor receives the bus and erroneously re-issues cycle A, completing it a second time.

IMPLICATION: Cycle A is completed twice on the system bus. In most cases the extra cycle will not create any system problems. If the cycle is to a memory-mapped I/O device, mis-operation could occur.

WORKAROUND: Either of two workarounds will avoid this erratum.

1. Disable pipelining.
2. Do not assert BOFF# during a pipelined cycle condition. Designs based on the 82430NX PCIset and other chip sets which do not assert BOFF# during a pipelined condition are therefore not affected

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

9DP. Double Issuance of Read Cycles

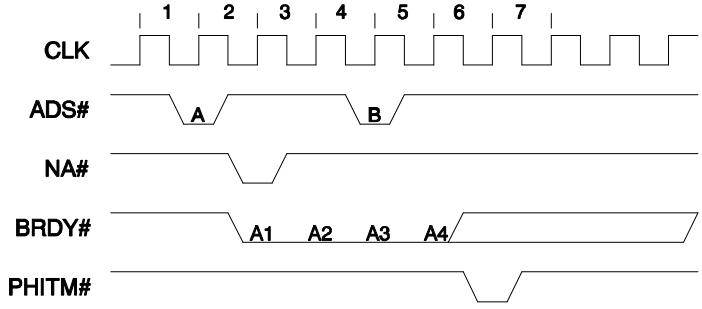
PROBLEM: In a dual processing environment, a memory read cycle (either data read or prefetch) may occur twice. The second read cycle may cause mis-operation of the CPU.

This problem occurs only in dual processing systems with a bus/core ratio of 2/3. For this to occur, pipelining must be enabled with some or all read cycles occurring in zero wait states.

For this erratum to occur, either of two specific sequence of conditions must occur on the bus as shown in the figures below:

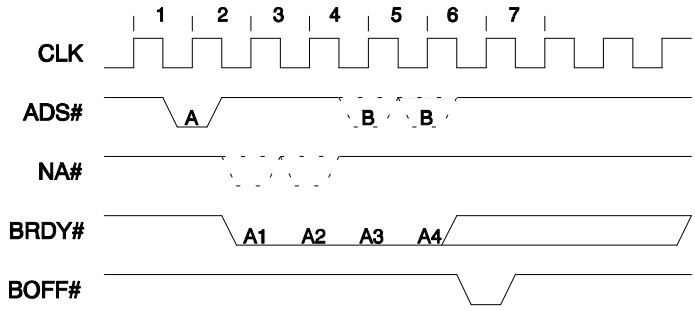
Case 1:

- The processor issues read cycle A which may be pipelined into an earlier (not shown) cycle.
- Cycle A will be completed by the system as a 2-1-1-1 (zero wait state) cycle.
- One clock after issuance of the ADS#, NA# is asserted by the system.
- A pipelined cycle (B) is asserted by the MRM in clock 4.
- Cycle A completes on the bus in clock 5.
- One clock later, PHITM# is issued by the LRM.



Case 2:

- The processor issues read cycle A which may be pipelined into an earlier (not shown) cycle.
- Cycle A will be completed by the system as a 2-1-1-1 (zero wait state) cycle.
- One or two clocks after issuance of the ADS#, NA# is asserted by the system.
- A pipelined cycle (B) is asserted by the MRM in either clock 4 or clock 5.
- Cycle A completes on the bus in clock 5.
- One clock later, BOFF# is issued by the system.



Specifically given the above conditions, the error condition occurs internally in the CPU due to the assertion of PHITM# or BOFF# one clock after the final BRDY# of cycle A. If this occurs, the CPU will respond to the PHITM# or BOFF# by subsequently re-issuing both cycles A and B, where expected operation would be that only cycle B is re-issued.

IMPLICATION: The code fetch or data read cycle (A) will be re-issued even though it is already completed. The second occurrence of the cycle, even if harmless from a system point of view, will confuse the internal state of the CPU and may cause subsequent CPU mis-operation.

WORKAROUND: Any of three workarounds will avoid this erratum:

1. Avoid assertion of BOFF# or PHITM# during the sensitive clock (clock 6). Avoiding assertion of PHITM# in clock 6 is guaranteed by asserting NA# no earlier than clock 3. Since ADS# occurs two or more clocks after NA# and since PHITM# occurs 2 clocks after ADS#, assertion of NA# in clock 3 or after will ensure that PHITM# is not driven active in clock 6.
2. Disable pipelining by not asserting NA#.
3. Do not perform zero wait-state read cycles.

Designs based on the 82430NX PCIset and other chip sets which do not perform zero wait-state read cycles are therefore not affected.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

10DP. Line Invalidation May Occur On Read or Prefetch Cycles

PROBLEM: When operating in dual processing mode, a read or prefetch cycle from either CPU may invalidate a cache line in the other.

In a dual processor environment, the LRM performs an internal snoop for each memory cycle the MRM drives onto the bus. If this snoop results in a hit in the either the code or data cache and the INV (Invalidate) signal is asserted by the system, the LRM will assert either the PHIT# or PHITM# signal and invalidate the snooped line. Expected operation is that the INV signal is not meaningful and that the LRM should respond only by asserting the PHIT# or PHITM# signal.

IMPLICATION: Unnecessary and unexpected invalidations in the LRM's caches will result. If this occurs, the only impact is to system performance; no functional problems occur with this erratum. Unnecessary invalidations in the LRM L1 caches will possibly decrease subsequent hit rate. The amount of performance degradation is a function of how many lines are shared between the two processors.

WORKAROUND: After external snoops have completed, the system should deassert the INV signal so that no invalidations are performed on subsequent private snoop operations. Designs based on the 82430NX PCIset and other chip sets which only assert INV during external snoops are therefore not affected.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

11DP. EADS# or Floating ADS# May Cause Extra Invalidates

PROBLEM: This erratum only occurs in a dual processing environment. Extra invalidates may occur into the L1 cache due to assertions of EADS#. If EADS# is asserted while the processor is driving the bus, an invalidate into the processor's L1 cache may occur.

IMPLICATION: The specification states that EADS# is ignored while the processor is driving the bus. Occurrence of this erratum means that unnecessary invalidations and writeback cycles may be performed resulting in sub-optimal performance.

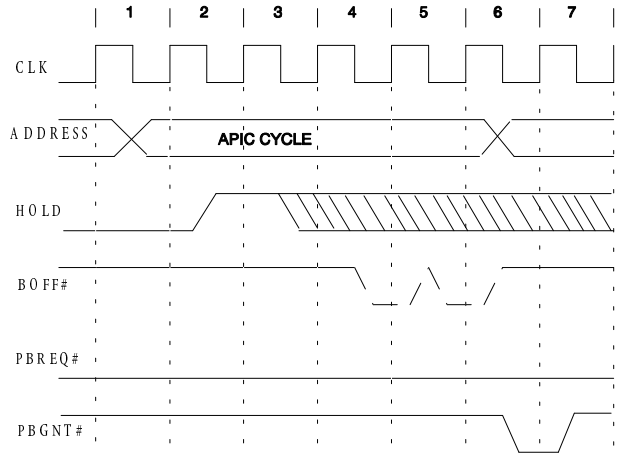
WORKAROUND: The system should not assert EADS# while the CPU owns the bus. Designs based on the 82430NX PCIset and other chip sets which do not assert EADS# while the CPU owns the bus are therefore not affected.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

12DP. HOLD and BOFF# During APIC Cycle May Cause Dual Processor Arbitration Problem

PROBLEM: In a dual processor system, the following sequence of events may cause the dual processing arbitration machines of both processors to lose synchronization and cause the system to hang:

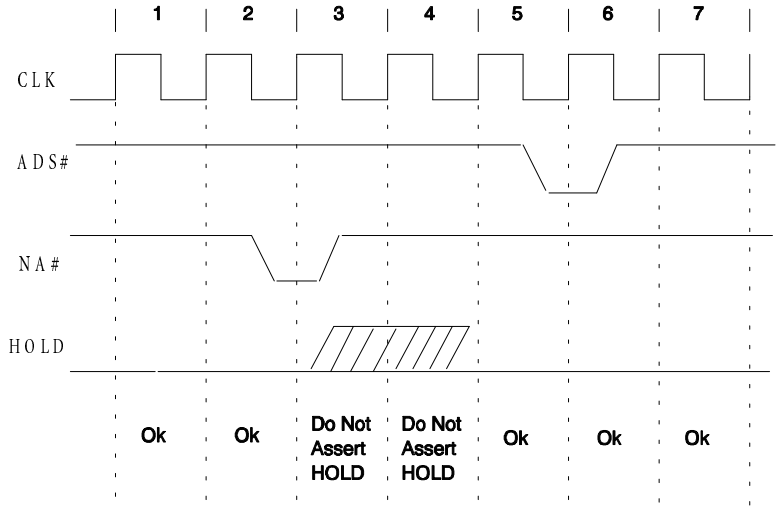
1. One of the CPUs initiates an internal APIC cycle (read or write). The LRM CPU requests ownership of the bus by asserting PBREQ#
2. HOLD is asserted during the APIC cycle
3. BOFF# is asserted before the APIC cycle gets completed, two or three clocks after HOLD is asserted (as shown in figure below)
4. Once BOFF# is deasserted, both processors may assume ownership of the bus at the same time resulting in possible contention of the CPU pins.



IMPLICATION: This problem affects dual processing (with CPU local APIC enabled) that assert HOLD and BOFF#. When the problem occurs, the dual processing arbitration machines of both processors get out of synchronization causing both processors to park on the bus. This will result in a system hang.

WORKAROUND: Use one of the following workarounds:

1. Do not use HOLD/HLDA protocol together with BOFF#. Use one or the other. Designs based on the 82430NX PCIset and other chip sets which do not use the HOLD/HLDA protocol together with BOFF# are therefore not affected.
2. In a non-pipelined system, avoid assertion of HOLD when the bus is idle. Asserting HOLD while CPU is running a bus cycle (between ADS# and last BRDY#) will ensure that HOLD does not hit an APIC cycle. Alternatively, if HOLD is asserted when the bus is idle, avoid asserting BOFF# two or three clocks after HOLD is asserted (as shown in figure above.)
3. In a pipelined system, use the same workaround as described in #2 with an additional requirement. Since an APIC cycle can be pipelined into another bus cycle, avoid assertion of HOLD in the clocks between NA# and the next ADS# (as shown in figure below.)



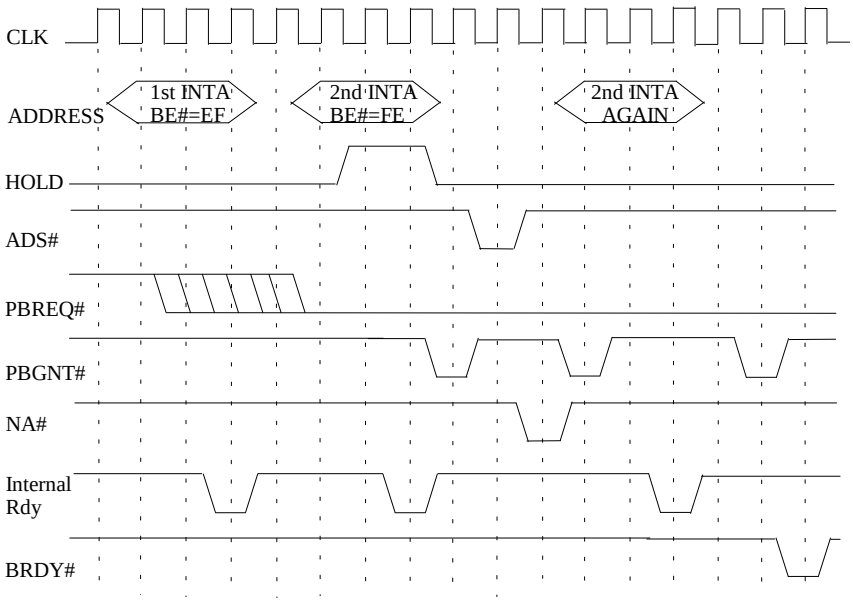
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

13DP. System Hang After Hold During Local APIC 2nd INTA Cycle

PROBLEM: In a dual processor system, the following sequence of events may cause the system to lock up:

1. One of the CPU's (i.e. CPU A) initiates the first and second internal APIC INTA cycles. The INTA cycles are not driven on the external bus. CPU B requests ownership of the bus by asserting PBREQ#.
2. HOLD is asserted during the 2nd INTA cycle and PBGNT# is asserted.
3. CPU B issues a read or write cycle and receives NA# before completion of the cycle. The bus is granted back to CPU A.
4. CPU A reissues the second INTA cycle and grants the bus to CPU B.

5. CPU A hangs because the internal Rdy# and BRDY# were recognized out of order.



IMPLICATION: Asserting HOLD during the second INTA cycle causes the processors to lose synchronization. The external cycle completes on the bus after the INTA cycle completes which hangs the system.

WORKAROUND: Use one of the following workarounds:

1. Do not use HOLD/HLDA protocol. Designs based on the 82430NX PCIset and other chip sets which do not use the HOLD/HLDA protocol are therefore not affected.
2. Disable pipelining.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

14DP. External Snoop Can Be Incorrectly Invalidated

PROBLEM: An external snoop with INV pin = 0 (non invalidating snoop) can be incorrectly treated as an invalidating snoop under the following conditions:

1. The system must use Dual Processors, operating in Intel's DP mode.
2. An external snoop occurs via EADS#, with INV = 0.
3. The previous bus master must have driven CACHE# = 1, M/IO# = 1, D/C# = 1, (W/R# = 1 or LOCK# = 0), or the pins must float to this value by the time EADS# is asserted. (This corresponds to an immediately preceding bus cycle that was non cacheable, memory, data and write or locked read.)

IMPLICATION: A small fraction of non invalidating external snoops will be incorrectly invalidated, which in turn will cause unnecessary write back cycles, resulting in sub-optimal performance if the system uses non

invalidating external snoops frequently. The degree of sub-optimal performance will depend on the details of system hardware and software, and most importantly, on the amount of non invalidating external snoops.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

15DP. STPCLK# Re-assertion Recognition Constraint With DP

PROBLEM: The *Pentium® Processor Family Developer's Manual*, Section 14.4.2.1 describes how to assure that each assertion and de-assertion of STPCLK# is recognized. However, it is not possible to guarantee that all changes on STPCLK# will be recognized in a DP system. This is because snoops between the dual processors triggered by the PHITM# signal can delay the processor's entry into the Stop Grant state until well after the end of the Stop Grant cycle. It is specified that de-assertion of STPCLK# must be held for at least 5 clocks after the beginning of the processor's entry into the Stop Grant state to be guaranteed to be recognized by the processor. As it is not practical for the system to monitor the PHITM# signal, there is no practical way to guarantee that deassertion of STPCLK# will be recognized.

IMPLICATIONS: A DP system should not be designed to depend on every STPCLK# assertion being recognized and thus generating a Stop Grant bus cycle response, and/or on every STPCLK# de-assertion allowing execution of at least one instruction. If a system design (such as typical usage of STPCLK# for thermal control and/or power usage reduction) does not depend on either of these features, this erratum will have no effect. Aside from sometimes not displaying these two features, a Pentium Processor system will never hang or otherwise malfunction because of random assertion and de-assertion of STPCLK#.

WORKAROUND: Do not design DP systems to depend on every STPCLK# assertion being recognized and thus generating a Stop Grant bus cycle response, or to depend on every STPCLK# de-assertion allowing execution of at least one instruction.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

16DP. Second Assertion of FLUSH# During Flush Acknowledge Cycle May Cause Hang

PROBLEM: The *Pentium® Processor Family Developer's Manual,* Section 3.5.1.2 states that in a DP system the FLUSH# signal must not be asserted again until the FLUSH ACK cycle is generated. The erratum occurs when the dual processor hasn't been initialized (with the IPI), and a FLUSH# is asserted during a FLUSH ACK cycle (anytime from ADS# to 1 clock after BRDY# of FLUSH ACK cycle).

IMPLICATION: Asserting FLUSH# in a DP system with the dual processor un-initialized by an IPI during a FLUSH ACK cycle may cause a hang.

WORKAROUND: Initialize the dual processor by sending an IPI, or do not assert FLUSH# during the FLUSH ACK cycle. The FLUSH# can safely be asserted two clocks after the completion (i.e. BRDY#) of the FLUSH ACK cycle.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

1AP. Remote Read Message Shows Valid Status After a Checksum Error

PROBLEM: If an APIC Remote Read (RR) transmission suffers checksum error, the RR bits of the register are mistakenly set to valid when they should show an invalid message state.

IMPLICATION: The implication is that the data portion (cycles 21-36) of the remote read message could be corrupted, but the RR status bits (bits [17:16] of the ICR0 register) would show a valid status of '10', when they should show an invalid status of '00'.

WORKAROUND: There is no workaround for this erratum, but checksum errors on the APIC bus imply that there are more serious noise issues inherent to the system that need to be addressed. In any event the RR messages should not be used if there are noise issues on the bus.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

2AP. Chance of Clearing an Unread Error in the Error Register

PROBLEM: A normal read of the APIC Error register clears the register. The clearing process waits 3 clocks to complete due to the possibility of being backed off. In the mean time if another error is written during this 3 clock delay, this new error overwrites the originally read error, and then is cleared at the end of the original 3 clock period.

IMPLICATION: An error could be posted in the APIC Error register but cleared prior to being read.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

3AP. Writes to Error Register Clears Register

PROBLEM: The APIC Error register is intended to only be read. If there is a write to this register the data in the APIC Error register will be cleared and lost.

IMPLICATION: There is a possibility of clearing the Error register status since the write to the register is not specifically blocked.

WORKAROUND: Writes should not occur to the Pentium processor APIC Error register.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

4AP. Three Interrupts of the Same Priority Causes Lost Local Interrupt

PROBLEM: If three interrupts of the same priority level (priority is defined in the 4MSB of the interrupt vector), arrive in the following circumstance:

1. A interrupt is being serviced by the CPU, and the proper bit is set in the ISR register.
2. A second interrupt is received from the serial bus.
3. At the same time a third interrupt is received from a local interrupt source, which could include local pins (LVT), an APIC timer (Timer), self-interrupt, or an APIC error interrupt.

If the first two conditions are met the third interrupt will be lost, and not serviced.

IMPLICATION: The third interrupt will be ignored and not serviced if the specific scenario happens as listed above.

WORKAROUND: The problem can be avoided if different priority levels are assigned to serial interrupts, than to local interrupts.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

5AP. APIC Bus Synchronization Lost Due to Checksum Error on a Remote Read Message

PROBLEM: This error only occurs when a Remote Read message request is processed, and the returned data has a non-zero value in the bits [30:29], and this returned data suffers a checksum (CS) error in the transmission. When the device that generated the Remote Read responds with the end of interrupt message (EOI) or the ICR message the APIC bus will lose synchronization.

IMPLICATION: If this rare condition occurs the APIC bus will become unusable, and will impact system operation. The system will hang because there will be no service on interrupts. Since RR messages are primarily used in system debug procedures, there is no impact foreseen on normal APIC or system operation.

WORKAROUND: There is no known workaround for this erratum; Remote Read messages should not be used. This error is mainly caused by checksum errors on the APIC bus which means that there are more serious noise issues inherent to the system that need to be fixed.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

6AP. HOLD During a READ from Local APIC Register May Cause Incorrect PCHK#

PROBLEM: If the processor is reading one of its local APIC registers when the HOLD pin is asserted, PCHK# may be asserted for one clock even though there is no data parity error. PCHK# will be asserted if the values of the data bits [D31:0] and the parity bits [DP3:DP0] do not match during the HOLD/HLDA transaction.

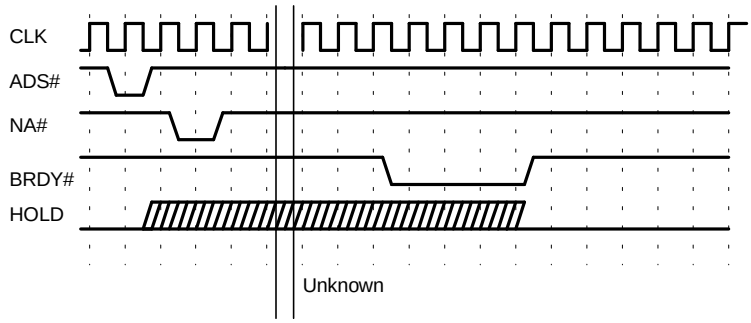
IMPLICATION: This will impact a system that implements or responds to parity checking by the CPU, the response will be specific to the parity error recovery routines implemented in the system. If parity is not implemented in a system, there will be NO adverse effect from this erratum since there is no real parity problem.

WORKAROUND: If data parity checking and the local APIC are both enabled, deassert PEN# (parity enable) during the time that HOLD is active. This signals to the processor that parity is not being driven from the system and PCHK# will never be driven in response to this data transfer.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

7AP. HOLD During an Outstanding Interprocessor Pipelined APIC Cycle Hangs Processor

PROBLEM: This is an APIC related dual processor erratum: When an APIC read cycle is interprocessor pipelined into any other allowable cycle, and HOLD is prior to the last BRDY# of the outstanding cycle, the MRM processor will hang.



IMPLICATION: A system that uses dual processor with pipelining enabled, is subject to periodic lockups if HOLD/HLDA# protocol is used.

WORKAROUND: Use one of the following:

1. Disable pipelining in dual processor operation.
2. Do not use the HOLD/HLDA# protocol, use BOFF# instead.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

8AP. PICCLK Reflection May Cause an APIC Checksum Error

PROBLEM: This is an APIC-related erratum: Even though the PICCLK signal is a slower frequency clock, it has been found to be extremely sensitive to the signal transition speeds and slopes. If the PICCLK specification for rise time is met but there is a "knee" or a reflection during a high or low going transition between 0.8V to 2.0V then the CPU may not correctly receive the APIC message, and will generate a checksum error and in addition it will try to resend the APIC message. This knee could be as small as 100-200ps, and it may still cause a problem. If the reflection occurs regularly, the resend tries on the bus could saturate the bus bandwidth and the system could lose interrupts or hang up.

IMPLICATION: Checksums happen occasionally, but if there is a knee in the PICCLK transition range then there is a much higher likelihood of the occurrence. A healthy system will only see one or two per day of operation, if this problem shows up then there is a chance that the resends of the checksum errors will saturate the APIC bus and hang the system.

WORKAROUND: Use the Intel Diagnostic tool under Microsoft Windows NT 3.1 to count the number of checksum errors that occur. This tool is available to OEMs through Intel, and only works with the latest release of the Windows NT 1.1 HAL. If you are an OEM, contact your Intel representative to get a copy.

Verify that the PICCLK signal meets the new .15ns (min), 2.5ns (max), specification for a rise from 0.8V to 2.0V or a fall between 2.0V and 0.8V. Also verify that this signal is "clean" , and there are no chances or evidence of reflection during this time. The reflection would show up as ledges in the transition of the signal in the 0.8V to 2.0V transition range. If there is any evidence of a reflection and the system shows errors on the diagnostic tool, then the PICCLK line must be reworked to clean this up. The rework could include a different clock driver, and/or rerouting the clock lines on the board. See the Specification Clarification that is part of this document for guidance on PICCLK routing. See also the Specification Change section of this document for more details on the PICCLK specification.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

9AP. Spurious Interrupt in APIC Through Local Mode

PROBLEM: This erratum affects APIC in through Local (virtual wire) mode. The system can be a uniprocessing or dual processing system that is using a Pentium processor with the APIC in through-local (virtual wire) mode. This mode is supposed to cause the processor to respond to interrupts identically to a Level triggered 8259 interrupt controller, and is typically used to provide AT compatibility mode for existing drivers. Currently it acts as an edge triggered mode interrupt controller, latching any interrupts that may be quickly asserted and then deasserted based on driver interception. The result is that some operating systems (i.e., Novell*) will report the spurious interrupt and it may impact the performance or operation of certain debug hooks for the operating system or network. Software disabling of the APIC by clearing bit 8 of the SVR (spurious vector interrupt register) will not prevent this from occurring.

IMPLICATION: Reports of the a spurious interrupt or lost interrupt message may continuously be output to the terminal connection and fill the screen of a monitoring host.

WORKAROUND: Use one of the following:

1. Ignore/disable the spurious interrupt reports. This may impact other debug hooks normally associated with the network or operating system.
2. Rewrite drivers such that they disable interrupt processing during the driver execution, and then re-enable the interrupts at the end of the procedure.
3. Disable APIC instead of running it in through Local mode.

   By Hardware: By deasserting the APICEN pin prior to the falling edge of reset.

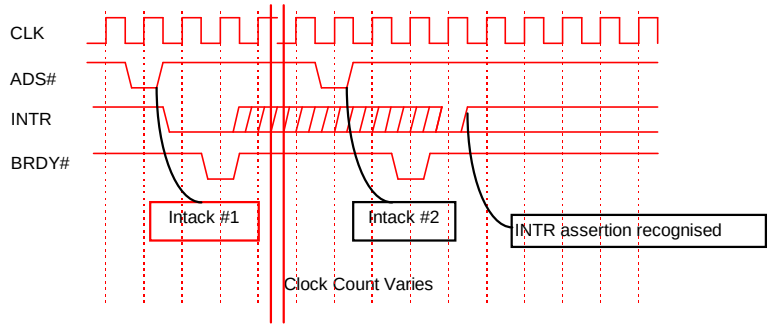
   By Software: This can be done on the B1, B3, B5, and C2-step components by using a reserved bit (bit 4) in the TR12 test register set to '1'. The use of a reserved bit is only for these steppings (B1, B3, B5, and C2) and the function of this bit may change in future steppings. When implementing this workaround ensure that the BIOS does a CPUID check looking for a specific stepping of the device. CPUIDs for the following components are B1 = 0521H, B3= 0522H, B5= 0524H and C2=0525H. If the TR12 register is used, the APIC is fully disabled. To re-enable APIC, bit 4 must be cleared to '0' and then a warm reset of the part performed prior to APIC use of any kind.

4. For cB1, cC0 and E0 steppings, a software fix can be enabled by setting bit 14 of TR12 to '1'. By enabling this bit, an interrupt that is asserted and deasserted during the window that interrupts are disabled (after CLI and before STI) is ignored. If the interrupt is asserted during this window and deasserted after interrupts are enabled (after STI sets IF), the interrupt is latched and serviced. By setting bit 14 to '0', the processor will behave as in earlier steppings.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

10AP. Potential for Lost Interrupts while Using APIC in Through Local Mode

PROBLEM: This erratum affects APIC in through Local (virtual wire) mode. If a uniprocessing or dual processing system is using a Pentium processor with the APIC in through Local (virtual wire) mode, and the chip set is able to re-assert the INTR line prior to completion of the second Interrupt acknowledge cycle (from a prior assertion of the INTR line), then the processor will neither recognize nor service the second interrupt. The assertion edge of INTR has to occur after the completion of the second IntAck BRDY#, if there is a transition and this transition is held high during the restricted time period, this INTR will not be recognized. Software disabling of the APIC by clearing bit 8 of the SVR (spurious vector interrupt register) will not prevent this from occurring.



IMPLICATION: If the conditions listed above are met the system may hang up since there would be an interrupt that would not get serviced.

WORKAROUND: Use one of the following:

1. Verify/Modify chip sets such that they cannot assert a second INTR for processing prior to the completion of both Interrupt acknowledge cycles for the first INTR.
2. Disable APIC instead of running in through Local mode.

   By Hardware: This can be done through hardware by deasserting the APICEN pin prior to the falling edge of reset.

   By Software: This can be done on the B-step components by using a reserved bit (bit 4) in the TR12 test register set to '1'. The use of a reserved bit is only for the B-steppings (B1, B3, or B5) of the 75-, 90-, and 100-MHz Pentium processors and the function of this bit may change in future steppings. When implementing this workaround ensure the BIOS does a CPUID check looking for a specific B-stepping of the device. CPUIDs for the following components are B1 = 0521H, B3= 0522H and B5= 0524H. If the TR12 register is used APIC is fully disabled. To re-enable APIC, bit 4 must be cleared to '0' and then a warm reset of the part performed prior to APIC use of any kind.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

11AP. Back to Back Assertions of HOLD or BOFF# May Cause Lost APIC Write Cycle

PROBLEM: If the processor is writing to one of its local APIC registers when the HOLD or BOFF# pin is asserted, then the next assertion of HOLD or BOFF# can potentially cause a subsequent APIC write cycle to be lost. This may occur as a result of the following sequence of events:

1. The CPU issues a write cycle to one of its local APIC registers (this cycles runs internally to the CPU but the corresponding APIC address is observed on the address bus while the cycle is executing)
2. HOLD or BOFF# is asserted while this APIC cycle is executing. (The write cycle to the local APIC register does complete but internal logic remembers to restart this cycle. For HOLD case, the cycle restarts upon deassertion of HOLD)
3. The processor asserts HLDA (if HOLD is asserted)
4. HOLD or BOFF# is deasserted
5. The processor deasserts HLDA (if HOLD is asserted)

6. The first APIC write cycle appears to restart
7. Another HOLD or BOFF# is asserted while this restarted APIC write cycle is executing internally
8. The processor asserts HLDA (if HOLD is asserted)
9. HOLD or BOFF# is deasserted
10. The processor deasserts HLDA (if HOLD is asserted)
11. The CPU issues the next APIC write cycle to one of its local APIC registers and there is no bus activity prior to this cycle and the previous restarted APIC write cycle.
12. This subsequent APIC write cycle is observed to start (the correct address is observed on the bus), however it fails to complete internally. In other words, from a software perspective this APIC write instruction is lost.

IMPLICATION: This problem affects systems that use HOLD/HLDA or BOFF# and enable the local APIC of the CPU. If the second APIC write cycle is an EOI (End of Interrupt) cycle, the CPU will stop servicing subsequent interrupts of equal or less priority. This may cause the system to hang. If the second APIC write cycle is not an EOI, the failure mode would depend on the particular APIC register that is not updated correctly.

WORKAROUND: Using one of the following workarounds will avoid this erratum:

1. This problem will not occur if an instruction in between the two APIC write commands in the code results in a bus cycle. This may also be achieved by inserting an APIC read instruction (reading one of the local APIC registers) before every APIC write instruction. Other instructions such as I/O or locked instructions would also force bus activity prior to executing an APIC write and will avoid this erratum.
2. Disable the local APIC if running in Uniprocessor mode.
3. In Dual processor mode, delay the next assertion of HOLD or BOFF# to allow the restarted APIC write cycle to complete.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

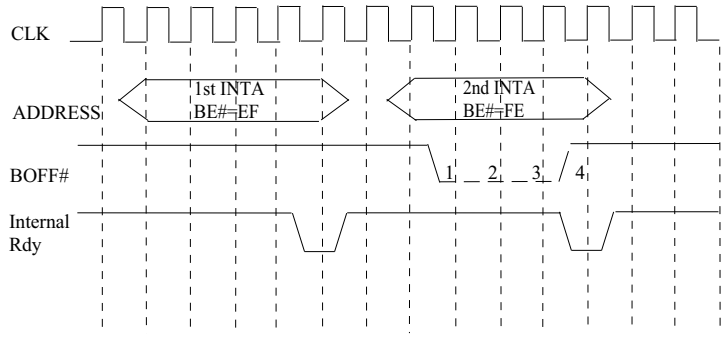
12AP. System May Hang When BOFF# Is Asserted During the Second Internal APIC INTA Cycle

PROBLEM: The processor will hang if BOFF# is asserted and deasserted during the second internal APIC INTA cycle. The erratum will occur if BOFF# is sampled during the interval (i.e. clocks 1, 2, or 3) shown in the figure below, although keeping BOFF# asserted through the last cycle (i.e. cycle 4) of the second INTA will prevent this erratum from occurring

If these conditions occur while a remote read message is being sent or received, the second INTA cycle may take up to 8 clocks to complete (counting from clock 1). BOFF# is not latched and must remain asserted until after the 2nd INTA completes (e.g. for 8 clocks).

Similarly, if HOLD is asserted anytime during the second INTA cycle, and during a remote read, the processor will hang.

BOFF# During 2nd INTA (Non-remote Read)



IMPLICATION: If the system does not assert BOFF# when the processor bus is idle, then this problem will not occur. Internal APIC INTA cycles are run only when the bus is idle, thus asserting BOFF# during an external bus cycle (e.g. started by an ADS#) will avoid these circumstances. In systems that can assert BOFF# when the bus is idle, asserting BOFF# for a least 4 clocks will avoid the problem for non-remote read cases. Note that there may be other implementations that guarantee BOFF# is not asserted in the problematic window. For example, if BOFF# is only asserted with AHOLD active and AHOLD always precedes BOFF# by at least 4 clocks, the erratum is avoided.

If a remote read is occurring (e.g. between two processors) this will delay completion of the second INTA cycle by up to 8 clocks (i.e. bus clocks), and BOFF# asserted during this time may hang the system. Remote reads are typically performed during system debug and not in normal operation. Not performing remote reads will avert this case.

Asserting HOLD anytime during the second INTA cycle during a remote read cycle will cause the system to hang. Not asserting HOLD or not performing APIC remote read cycles will avert this case.

WORKAROUND: Use one of the following to avoid the BOFF# case:

1. For non-remote read case, do not assert BOFF# when the bus is idle, or assert BOFF# for at least 4 clocks during idle bus cycles. For the remote read case, assert BOFF# for at least 8 clocks.
2. Use APIC in through local mode.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

13AP. APIC Pipeline Cycle During Cache Linefill Causes Restarted Cycle to Lose Its Attribute

PROBLEM: When a read or write cycle to an APIC register is pipelined into a cache linefill and both cycles get backed off (by assertion of BOFF# for **one clock** only), the cache linefill that is restarted loses its attributes. When the cache linefill is restarted, although the processor does assert CACHE#, the processor loses track of the cacheability of the cycle and treats the burst linefill as a single cycle read.

IMPLICATION: The processor reads only the first quad-word (indicated by the first BRDY#), but ignores the following three transfers of the burst linefill. However, the internal APIC cycle is allowed to restart after the first BRDY#. When the APIC cycle completes and another bus cycle is started by the processor (indicated by an ADS#) before the last BRDY# from the burst linefill is returned, the leftover BRDY#s could incorrectly

terminate the new cycle and the processor could lose synchronization with the bus, causing the processor to hang or get corrupted data.

It is unlikely this erratum will occur for systems using zero wait states (i.e. 2111 burst read) or one wait state lead off (i.e. 3111 burst read). A high-latency memory subsystem or I/O subsystem would increase the exposure of the new bus cycle to a leftover BRDY# (i.e. 3222 burst read).

WORKAROUND: Use one of the following:

1. Always assert BOFF# for more than one clock.
2. Disable pipelining when using the APIC.
3. Avoid asserting BOFF# during pipelined linefill cycles when using the APIC.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

14AP. INIT and SMI# Via the APIC Three-Wire Bus May Be Lost

PROBLEM: If the INIT and SMI# pins are kept asserted once they are recognized and then another INIT or SMI# is asserted to the processor via the APIC three-wire bus, the processor will not recognize this second assertion of INIT or SMI#.

INIT and SMI# are edge triggered interrupts and are only recognized on the rising edge (falling edge for SMI#). Since the processor only detects the edges on these pins, it is possible to hold the levels on these pins in the asserted state (logic 1 for INIT and logic 0 for SMI#). When another INIT or SMI# is required, the levels at these pins can be deasserted for several clocks and reasserted to generate the edge which triggers the interrupt. However, if the levels on these pins are kept asserted, and the APIC three-wire bus is also used to assert INIT and SMI# to the processor, the INIT and SMI# interrupts via the APIC three-wire bus are lost.

IMPLICATION: If the above conditions are met, INIT and SMI# interrupts via the APIC three-wire bus will be lost. Designs which do not use the APIC three-wire bus to assert INIT and SMI# will not be affected by this erratum.

WORKAROUND: To avoid this erratum, use one of the following:

1. Assert INIT or SMI# to trigger the interrupt and then deassert the INIT or SMI# thereafter to avoid conflict with the APIC serial bus INIT or SMI# messages.
2. Do not send an INIT or SMI# message via the APIC three-wire bus.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

15AP. IERR# in FRC Lock-Step Mode During APIC Write

PROBLEM: When an APIC write is pipelined into a memory write, IERR# is incorrectly asserted one clock after the BRDY# of the memory write for a duration of one clock. (Note that APIC write cycles are not driven on the external bus). This problem is a subset of the problem described in Erratum 29.

IMPLICATION: This will cause an inadvertent IERR# to occur for one clock.

WORKAROUND: Disable pipelining in FRC lock-step mode.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

16AP. Inadvertent BRDY# During External INTA Cycle With BOFF#

PROBLEM: The *Pentium® Processor Family Developer's Manual* states that BRDY# is ignored during assertion of ADS#. There are two cases when using the APIC in through local mode where an inadvertent BRDY# asserted during the ADS# of an external INTA cycle, in combination with a subsequent BOFF#, can cause the processor to hang.

1. If during the first INTA cycle an inadvertent BRDY# is asserted with ADS#, followed by the real BRDY# for that cycle, and then the 2nd INTA cycle is backed off, the processor loses synchronization. Instead of restarting the 2nd INTA cycle externally, the processor ends the cycle internally without reading a valid interrupt number (0-255) which hangs the interrupt handler. The window that BOFF# can cause this erratum is after (the valid BRDY#) completion of the 1st INTA cycle and before completion of the 2nd INTA cycle.
2. If the 1st INTA cycle completes correctly (with only one valid BRDY#), and an inadvertent BRDY# is asserted during the ADS# of the 2nd INTA cycle, and then the 2nd INTA cycle is backed off before its completion, the processor again loses synchronization and hangs.

IMPLICATION: The interrupt will not be serviced and the system hangs waiting for the processor to complete its 2nd INTA cycle.

WORKAROUND: Use one of the following:

1. Do not assert BRDY# during ADS#.
2. Do not assert BOFF# during an external INTA cycle.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

17AP. APIC Read Cycle Doesn't Complete Upon Assertion of BOFF# and HOLD

PROBLEM: During an APIC cycle (read or second INTA), if BOFF# is asserted for one clock, and HOLD is asserted in the following cycle (at the rising edge of BOFF#) for one clock, the APIC cycle may either not complete or not complete correctly. Either the processor is waiting for the APIC cycle to complete before issuing any new cycles and the processor hangs (APIC read), or the cycle does complete but with the incorrect Interrupt Vector being recognized (APIC second INTA). Note that this erratum does not occur when BOFF# and HOLD are asserted simultaneously for one clock.

IMPLICATION: Systems typically use either BOFF# or HOLD (but not both) to gain control of the bus. If a system were to assert this sequence of BOFF# and HOLD for one clock each, the system may be susceptible to a hang.

WORKAROUND: Do not assert BOFF# for one clock immediately followed by HOLD for one clock. If HOLD must follow BOFF# by one clock, assert one of the signals (BOFF# or HOLD) for more than one clock.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

18AP. PICCLK Must Toggle For at Least Twenty Cycles Before RESET

PROBLEM: In order for the internal circuitry of the local APIC to initialize properly, PICCLK must toggle at least twenty times (1.2μS – 10μS depending on the PICCLK frequency) before the falling edge of RESET.

IMPLICATION: An improper initialization of the internal APIC circuits may cause, for example, the APICEN/PICD1 pin to be erroneously driven low; thus, the on-chip APIC would not be enabled. In such a scenario, the dual-processor in DP systems and all messages sent on the serial APIC bus would not be recognized.

WORKAROUND: Ensure that PICCLK toggles for at least twenty cycles before the falling edge of RESET.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

19AP. APIC ID Can Not Be Changed

PROBLEM:: BE[3:0]# can be used on the Pentium processor as APIC ID inputs and are sampled at RESET. Due to the strong internal pulldown on these pins, it may not be possible to change the default APIC ID when the pins are sampled at RESET.

IMPLICATION: Only the default APIC ID (0000 for OEM and 0001 for DP upgrade) can be set at RESET.

WORKAROUND: The APIC ID can also be configured through software. To change the APIC ID after RESET, write the value to Local APIC Unit ID Register.

1TCP. CPU May Not Reset Correctly Due to Floating FRCMC# Pin

PROBLEM: The functional redundancy master/checker (FRCMC#) input is sampled by the processor during reset (it is ignored after reset). If it is sampled active, then it tri-states the outputs. In the TCP package, this input is not bonded out, and is therefore floating internally. The possibility exists that the processor will sample this input low during reset and tri-state the outputs.

IMPLICATION: The system may fail to boot up.

WORKAROUND: If CPU fails to reset, reboot the system.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

2TCP. BRDY# Does Not Have Buffer Selection Capability

PROBLEM: The capability of configuring selectable buffer sizes via the BRDY# and BUSCHK# pins is not available in the TCP package; only the Typical Stand Alone Component strength is available.

IMPLICATION: Only the Typical Stand Alone Component buffer size (the smallest, # EB2) is available in the TCP package. This erratum is expected to impact few if any notebook designs; the higher buffer strengths are helpful only for large designs, which are normally built into server and some desktop systems.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.