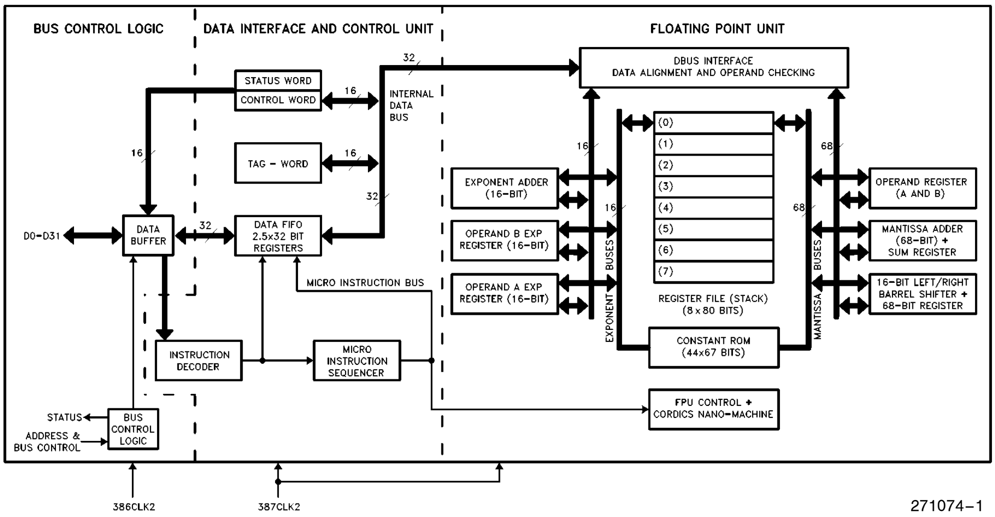
MILITARY i387™ MATH COPROCESSOR

- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Five to Seven Times M8087/M80287 Performance**
- **Upward Object-Code Compatible from M8087 and M80287**
- **Expands i386™ Microprocessor Data Types to Include 32-, 64-, 80-Bit Floating Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends i386 Processor Instruction Set to Include Trigonometric, Logarithmic, Exponential and Arithmetic Instructions for All Data Types**
- **Full-Range Transcendental Operations for SINE, COSINE, TANGENT, ARCTANGENT and LOGARITHM**
- **Built-In Exception Handling**
- **Operates Independently of Real, Protected and Virtual-8086 Modes of the i386 Microprocessor**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in 68-Pin PGA Package and 68-Lead Ceramic Quad Flat Pack**
  (See Packaging Spec: Order #231369)
- **Available in Three Product Grades:**
  — MIL-STD-883, −55°C to +125°C (T_C)
  — Military Temperature Only, −55°C to +125°C (T_C)
  — Extended Temperature, −40°C to +110°C (T_C)

The Intel i387 is a high-performance numerics processor extension that extends the i386 microprocessor architecture with floating point, extended integer and BCD data types. The i386/i387 processors' computing system fully conforms to the ANSI/IEEE floating-point standard. Using a numerics oriented architecture, the i387 processor adds over seventy mnemonics to the i386/i387 processor instruction set, making the i386/i387 processor a complete solution for high-performance numerics processing. The i387 microprocessor is implemented with 1.5 micron, high-speed CHMOS technology and packaged in a 68-pin ceramic pin grid array (PGA) package and a 68-pin ceramic quad flat pack package. The i386/i387 processor combination is upward object-code compatible from the i386/80287, 80286/80287 and 8086/8087 computing systems.



BUS CONTROL LOGIC
DATA INTERFACE AND CONTROL UNIT
FLOATING POINT UNIT
DBUS INTERFACE DATA ALIGNMENT AND OPERAND CHECKING
STATUS WORD
CONTROL WORD
INTERNAL DATA BUS
TAG - WORD
DATA FIFO 2.5x32 BIT REGISTERS
DATA BUFFER
D0-D31
MICRO INSTRUCTION BUS
INSTRUCTION DECODER
MICRO INSTRUCTION SEQUENCER
STATUS
ADDRESS & BUS CONTROL
BUS CONTROL LOGIC
EXPONENT ADDER (16-BIT)
OPERAND B EXP REGISTER (16-BIT)
OPERAND A EXP REGISTER (16-BIT)
EXPONENT BUSES
REGISTER FILE (STACK) (8x80 BITS)
CONSTANT ROM (44x67 BITS)
MANTISSA BUSES
OPERAND REGISTER (A AND B)
MANTISSA ADDER (68-BIT) + SUM REGISTER
16-BIT LEFT/RIGHT BARREL SHIFTER + 68-BIT REGISTER
FPU CONTROL + CORDICS NANO-MACHINE
386CLK2
387CLK2
271074-1

Figure 0.1. i387™ Math Coprocessor Block Diagram

Military i387™ Math Coprocessor

CONTENTS PAGE

CONTENTS PAGE

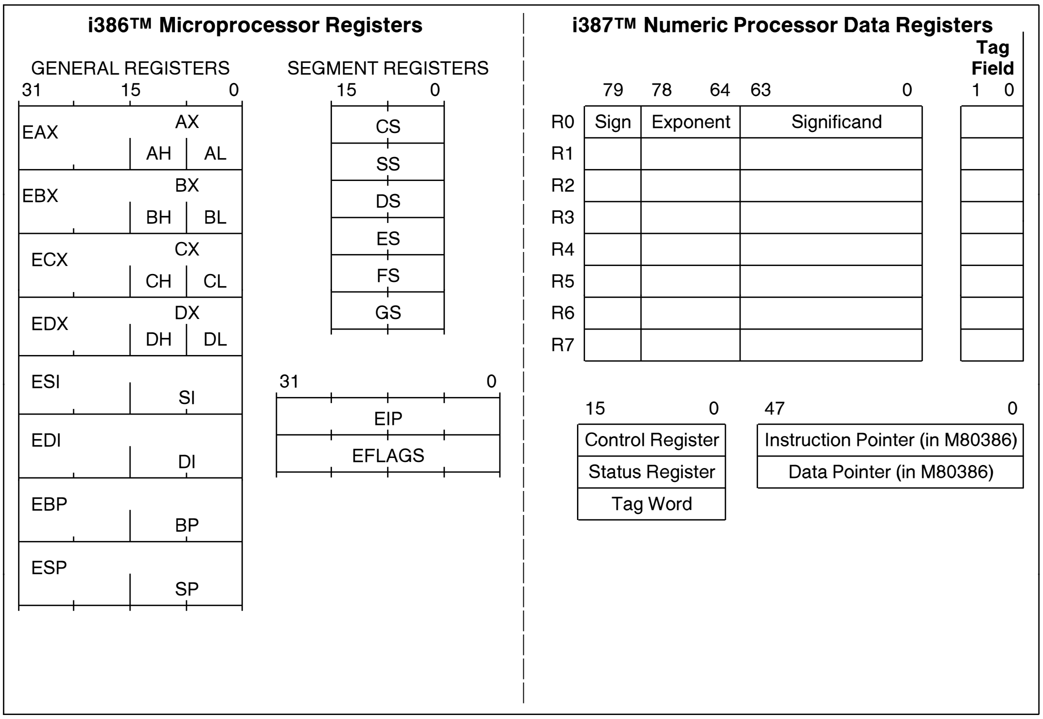


Figure 1.1. i386™/i387™ Processors Register Set

1.0 FUNCTIONAL DESCRIPTION

The i387 Numeric Processor Extension (NPX) provides arithmetic instructions for a variety of numeric data types in i386/i387 processor systems. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The i387 microprocessor effectively extends the register and instruction set of an i386 processor system for existing data types and adds several new data types as well. Figure 1.1 shows the model of registers visible to i386/i387 processor programs. Essentially, the i387 NPX can be treated as an additional resource or an extension to the i386 microprocessor. The i386 microprocessor together with an i387 NPX can be used as a single unified system.

The i387 NPX works the same whether the i386 microprocessor is executing in real-address mode, protected mode, or virtual-M8086 mode. All memory access is handled by the i386 processor; the i387 NPX merely operates on instructions and values passed to it by the i386 processor. Therefore, the i387 NPX is not sensitive to the processing mode of the i386 microprocessor.

In real-address mode and virtual-8086 mode, the i386/i387 processor combination is completely upward compatible with software for M8086/M8087, M80286/M80287 real-address mode, and i386/M80287 processor real-address mode systems.

In protected mode, the i386/i387 processor combination is completely upward compatible with software for M80286/M80287 protected mode, and i386/M80287 processor protected mode systems.

The only differences of operation that may appear when M8086/M8087 programs are ported to a protected-mode i386/i387 microprocessor system (*not* using virtual-M8086 mode), is in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR and FSAVE. These instructions are normally used only by exception handlers and operating systems, not by applications programs.

The i387 NPX contains three functional units that can operate in parallel to increase system performance. The i386 microprocessor can be transferring commands and data to the i387 NPX's *bus control logic* for the next instruction while the i387 NPX's *floating-point unit* is performing the current numeric instruction.

2.0 PROGRAMMING INTERFACE

The i387 coprocessor adds to an i386 processor system additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. To use the i387 coprocessor requires no special programming tools, because all new instructions and data types are directly supported by the i386 microprocessor assembler and compilers for high-level languages. All M8086/M8088 development tools that support the M8087 can also be used to develop software for the i386/i387 processors in real-address mode or virtual-M8086 mode. All M80286 development tools that support the M80287 can also be used to develop software for the i386/i387 processors.

All communication between the i386 microprocessor and the i387 NPX is transparent to applications software. The CPU automatically controls the i387 NPX whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the i387 NPX. All memory addressing modes, including use of displacement, base register, index register, and scaling, are available for addressing numerics operands.

Section 6 at the end of this data sheet lists by class the instructions that the i387 NPX adds to the instruction set of an i386 microprocessor system.

2.1 Data Types

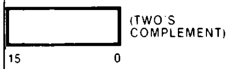
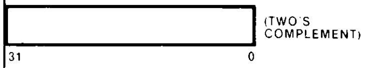
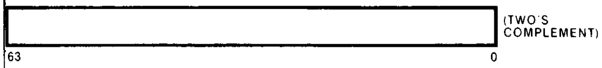
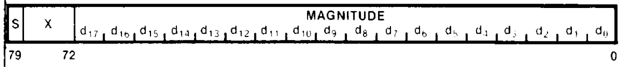
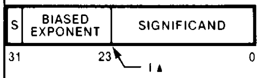
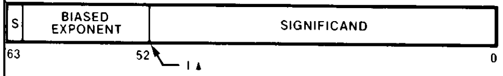
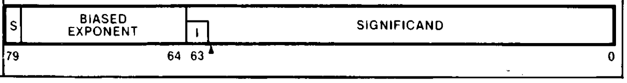
Table 2.1 lists the seven data types that the i387 NPX supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses evenly divisible by four (doubleword boundaries); operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

Internally, the i387 coprocessor holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating-point numbers, or 18-digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

2.2 Numeric Operands

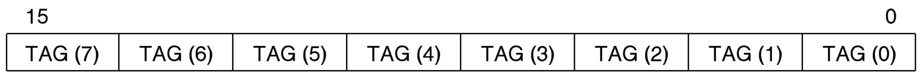
A typical NPX instruction accepts one or two operands and produces a single result. In two-operand instructions, one operand is the contents of an NPX register, while the other may be a memory location. The operands of some instructions are predefined; for example FSQRT always takes the square root of the number in the top stack element.

Table 2.1. i387™ NPX Data Type Representation in Memory

Data Formats	Range	Precision	**Most Significant Byte** HIGHEST ADDRESSED BYTE 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0
Word Integer	10^4	16 Bits	
Short Integer	10^9	32 Bits	
Long Integer	10^{19}	64 Bits	
Packed BCD	10^{18}	18 Digits	 S X MAGNITUDE d17 d16 d15 d14 d13 d12 d11 d10 d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 79 72 0
Single Precision	$10^{\pm 38}$	24 Bits	
Double Precision	$10^{\pm 308}$	53 Bits	
Extended Precision	$10^{\pm 4932}$	64 Bits	 S BIASED EXPONENT I SIGNIFICAND 79 64 63 0

271074–2

NOTES:
(1) S = Sign bit (0 = positive, 1 = negative)
(2) d_n = Decimal digit (two per byte)
(3) X = Bits have no significance; i387 NPX ignores when loading, zeros when storing
(4)▲ = Position of implicit binary point
(5) I = Integer bit of significand; stored in temporary real, implicit in single and double precision
(6) Exponent Bias (normalized values):
Single: 127 (7FH)
Double: 1023 (3FFH)
Extended Real: 16383 (3FFFH)
(7) Packed BCD: $(-1)^S (D_{17}...D_0)$
(8) Real: $(-1)^S (2^{E-BIAS}) (F_0\ F_1...)$



NOTE:
The index i of tag(i) is **not** top-relative. A program typically uses the "top" field of Status Word to determine which tag(i) field refers to logical top of stack.
TAG VALUES:

00 =	Valid
01 =	Zero
10 =	QNaN, SNaN, Infinity, Denormal and Unsupported Formats
11 =	Empty

Figure 2.1. i387™ NPX Tag Word

2.3 Register Set

Figure 1.1 shows the i387 NPX register set. When an i387 NPX is present in a system, programmers may use these registers in addition to the registers normally available on the i386 processor.

2.3.1 DATA REGISTERS

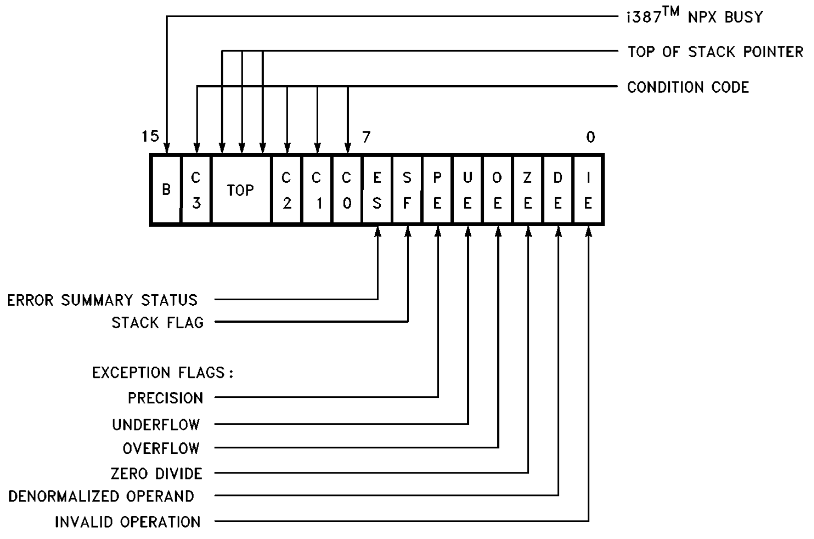
i387 processor computations use the i387 NPX's data registers. These eight 80-bit registers provide the equivalent capacity of twenty 32-bit registers. Each of the eight data registers in the i387 NPX is 80 bits wide and is divided into "fields" corresponding to the NPXs extended-precision real data type.

The i387 NPX register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by one. Like i386 microprocessor stacks in memory, the i387 NPX register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to user. This explicit register addressing is also relative to TOP.

2.3.2 TAG WORD

The tag word marks the content of each numeric data register, as Figure 2.1 shows. Each two-bit tag represents one of the eight numerics registers. The principal function of the tag word is to optimize the NPXs performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to check the contents of a stack location without the need to perform complex decoding of the actual data.



271074–3

ES is set if any unmasked exception bit is set; cleared otherwise.
See Table 2.2 for interpretation of condition code.
TOP values:
000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack
•
•
•
111 = Register 7 is Top of Stack
For definitions of exceptions, refer to the section entitled "Exception Handling"

Figure 2.2. i387™ NPX Status Word

2.3.3 STATUS WORD

The 16-bit status word (in the status register) shown in Figure 2.2 reflects the overall state of the M80387. It may be read and inspected by CPU code.

Bit 15, the B-bit (busy bit) is included for M8087 compatibility only. It reflects the contents of the ES bit (bit 7 of the status word), not the status of the $\overline{\text{BUSY}}$ output of i387/M80287 processors.

Bits 13–11 (TOP) point to the M80387 register that is the current top-of-stack.

The four numeric condition code bits (C_3–C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2.2 through 2.5.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the $\overline{\text{ERROR}}$ signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow ($C_1 = 1$) and underflow ($C_1 = 0$).

Figure 2.2 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the i387 NPX has detected an exception while executing an instruction. A later section entitled "Exception Handling" explains how they are set and used.

Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and its reflection in the B-bit (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the $\overline{\text{ERROR}}$ output of the i387 NPX is activated immediately.

Table 2.2. Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (see Table 2.3)	Three least significant bits of quotient Q2	Q0	Q1 or $\overline{\text{O/U}}$	Reduction 0 = complete 1 = incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 2.4)		Zero or $\overline{\text{O/U}}$	Operand is not comparable (Table 2.4)
FXAM	Operand class (see Table 2.5)		Sign or $\overline{\text{O/U}}$	Operand class (Table 2.5)
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or $\overline{\text{O/U}}$	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or $\overline{\text{O/U}}$	UNDEFINED
FPTAN, FSIN FCOS, FSINCOS	UNDEFINED		Roundup or $\overline{\text{O/U}}$, undefined if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

$\overline{\text{O/U}}$	When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).
Reduction	If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.
Roundup	When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.
UNDEFINED	Do not rely on finding any specific value in these bits.

Table 2.3. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interaction required for complete reduction	
0	Q1	Q0	Q2	Q MOD8	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient
	0	0	0	0	
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 2.4. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 2.5. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

2.3.4 INSTRUCTION AND DATA POINTERS

Because the NPX operates in parallel with the CPU, any errors detected by the NPX may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numeric instruction, the i386/i387 processor combination contains two pointer registers that supply the address of the failing numeric instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written error handlers. These registers are actually located in the i386 microprocessor, but appear to be located in the i387 NPX because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR. (In the M8086/M8087 and M80286/M80287, these registers are located in the NPX.) Whenever the i386 processor decodes a new ESC instruction, it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present), and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the i386 processor (protected mode or real-address mode) and depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). When the i386 microprocessor is in virtual-M8086 mode, the real-address mode formats are used. (See Figures 2.3 through 2.6.) The ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR are used to transfer these values between the M80386 registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

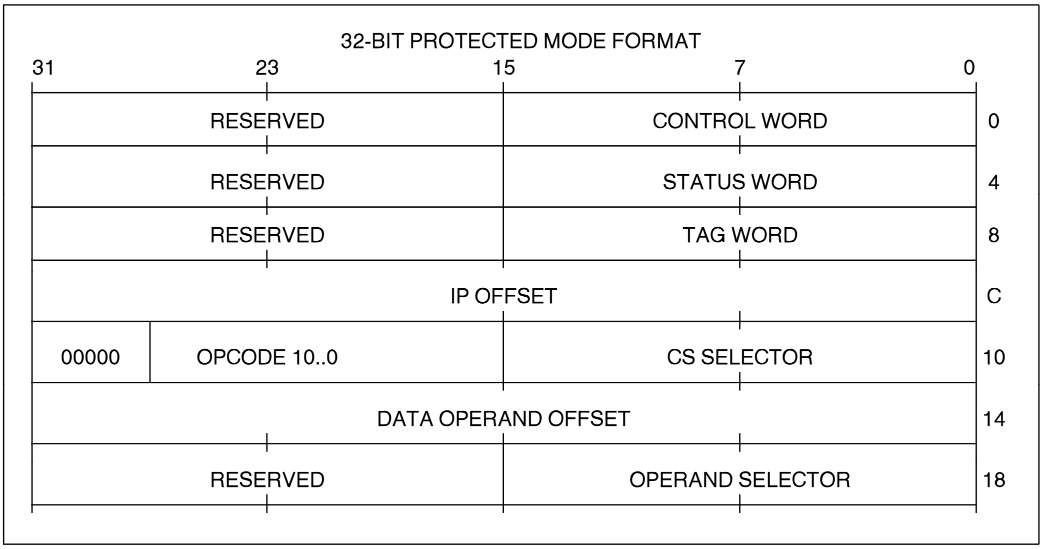


Figure 2.3. Protected Mode i387™ NPX Instruction and Data Pointer Image in Memory, 32-Bit Format

32-BIT REAL-ADDRESS MODE FORMAT

31	23	15	7 0	
RESERVED		CONTROL WORD		0
RESERVED		STATUS WORD		4
RESERVED		TAG WORD		8
RESERVED		INSTRUCTION POINTER 15..0		C
0 0 0 0	INSTRUCTION POINTER 31..16	0	OPCODE 10..0	10
RESERVED		OPERAND POINTER 15..0		14
0 0 0 0	OPERAND POINTER 31..16	0 0 0 0	0 0 0 0 0 0 0 0	18

Figure 2.4. Real Mode i387™ NPX Instruction and Data Pointer Image in Memory, 32-Bit Format

16-BIT PROTECTED MODE FORMAT

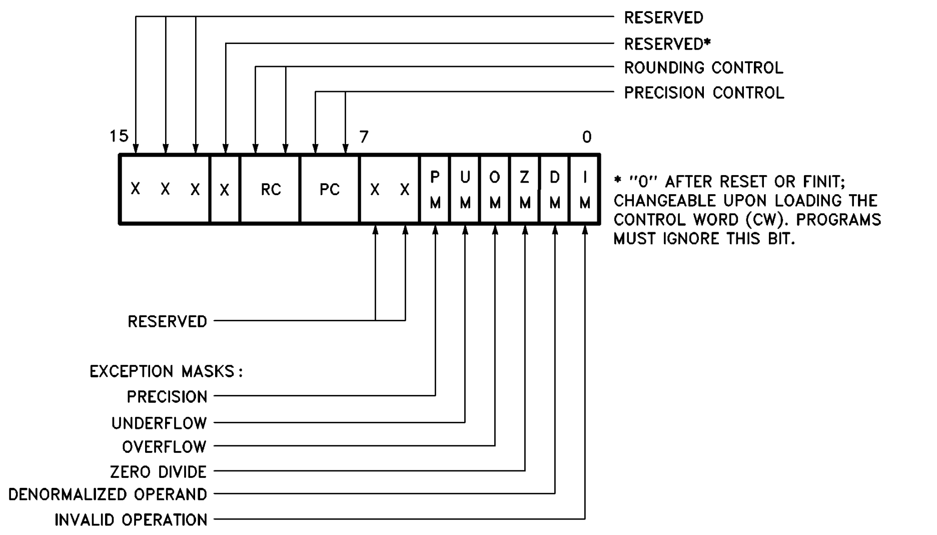
15 7 0	
CONTROL WORD	0
STATUS WORD	2
TAG WORD	4
IP OFFSET	6
CS SELECTOR	8
OPERAND OFFSET	A
OPERAND SELECTOR	C

Figure 2.5. Protected Mode i387™ NPX Instruction and Data Pointer Image in Memory, 16-Bit Format

16-BIT REAL-ADDRESS MODE AND VIRTUAL-M8086 MODE FORMAT

15		7 0	
CONTROL WORD			0
STATUS WORD			2
TAG WORD			4
INSTRUCTION POINTER 15..0			6
IP19.16	0	OPCODE 10..0	8
OPERAND POINTER 15..0			A
DP 19.16	0	0 0 0 0 0 0 0 0 0 0 0	C

Figure 2.6. Real Mode i387™ NPX Instruction and Data Pointer Image in Memory, 16-Bit Format



Precision Control
00—24 bits (single precision)
01—(reserved)
10—53 bits (double precision)
11—64 bits (extended precision)

Rounding Control
00—Round to nearest or even
01—Round down (toward $-\infty$)
10—Round up (toward $+\infty$)
11—Chop (truncate toward zero)

Figure 2.7. i387™ NPX Control Word

2.3.5 CONTROL WORD

The NPX provides several processing options that are selected by loading a control word from memory into the control register. Figure 2.7 shows the format and encoding of fields in the control word.

The low-order byte of this control word configures the i387 NPX error and exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the i387 processor recognizes.

The high-order byte of the control word configures the i387 NPX operating mode, including precision and rounding.

- Bit 12 no longer defines infinity control and is a reserved bit. Only affine closure is supported for infinity arithmetic. The bit is initialized to zero after RESET or FINIT and is changeable upon loading the CW. Programs must ignore this bit.
- The rounding control (RC) bits (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS, and FCHS), and all transcendental instructions.
- The precision control (PC) bits (bits 9–8) can be used to set the i387 NPX internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

2.4 Interrupt Description

Several interrupts of the i386 processor are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 2.6 shows these interrupts and their causes.

Table 2.6. i386™ Microprocessor Interrupt Vectors Reserved for NPX

Interrupt Number	Cause of Interrupt
7	An ESC instruction was encountered when EM or TS of i386 processor control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current NPX context may not belong to the current task.
9	An operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for small segments, 0FFFFFFFFH for big segments, zero for expand-down segments) and spanned inaccessible addresses[a]. The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. As with the M80286/M80287, the segment overrun exception should be handled by executing an FNINIT instruction (i.e. an FINIT without a preceding WAIT). The return address on the stack does not necessarily point to the failing instruction nor to the following instruction. The interrupt can be avoided by never allowing numeric data to start within 108 bytes of the end of a segment.
13	The first word or doubleword of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The M80387 has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The M80386 return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the NPX. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

a. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFC will span addresses FFFC–FFFF and 0000-0003; however addresses FFFE and FFFF are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible pages but intermediate bytes of the operand fall in a not-present page or a page to which the procedure does not have access rights.

2.5 Exception Handling

The i387 NPX detects six different exception conditions that can occur during instruction execution. Table 2.7 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the i387 NPX if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the $\overline{\text{ERROR}}$ signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 16 occurs. The exception condition must be resolved via an interrupt service routine. The i386/i387 processor combination saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

2.6 Initialization

i387 NPX initialization software must execute an FNINIT instruction (i.e. an FINIT without a preceding WAIT) to clear $\overline{\text{ERROR}}$. The FNINIT is not required for the M80287, though Intel documentation recommends its use (refer to the Numerics Supplement to the *80286 Programmer's Reference Manual*). After a hardware RESET, the $\overline{\text{ERROR}}$ output is asserted to indicate that an M80387 is present. To accomplish this, the IE and ES bits of the status word are set, and the IM bit in the control word is reset. After FNINIT, the status word and the control word have the same values as in an M80287 after RESET.

2.7 M8087 and M80287 Compatibility

This section summarizes the differences between the i387 NPX and the M80287. Any migration from the M8087 directly to the i387 NPX must also take into account the differences between the M8087 and the M80287 as listed in Appendix A.

Many changes have been designed into the i387 NPX to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

2.7.1 GENERAL DIFFERENCES

The i387 processor supports only affine closure for infinity arithmetic, not projective closure. Bit 12 of the Control Word (CW) no longer defines infinity control. It is a reserved bit; but it is initialized to zero after RESET or FINIT and is changeable upon loading the CW. Programs must ignore this bit.

Operands for FSCALE and FPATAN are no longer restricted in range (except for $\pm\infty$); F2XM1 and FPTAN accept a wider range of operands.

The results of transcendental operations may be slightly different from those computed by M80287.

In the case of FPTAN, the i387 NPX supplies a true tangent result in ST(1), and (always) a floating point 1 in ST.

Rounding control is in effect for FLD *constant*.

Software cannot change entries of the tag word to values (other than empty) that do not reflect the actual register contents.

After reset, FINIT, and incomplete FPREM, the i387 NPX resets to zero the condition code bits C_3-C_0 of the status word.

In conformance with the IEEE standard, the i387 NPX does not support the special data formats: pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

Table 2.7. Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signaling NaN, unsupported format, indeterminate form ($0*\infty$, 0/0, $(+\infty)+(-\infty)$, etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e. it has the smallest exponent but a nonzero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g. 1/3); the result is rounded according to the rounding mode.	Normal processing continues

2.7.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the M80387:

1. When the overflow or underflow exception is masked, the i387 NPX differs from the M80287 in rounding when overflow or underflow occurs. The i387 NPX produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the i387 NPX sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the i387 NPX automatically normalizes denormal operands. The M8087/M80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized operand exception. When loading a signaling NaN, FLD *single/double precision* signals an invalid-operand exception.
12. The i387 NPX only generates quiet NaNs (as on the M80287); however, the i387 NPX distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The M80287/M8087 leaves the original operand in ST(1) intact.
14. When the scaling factor is $\pm\infty$, the FSCALE (ST(0), ST(1)) instruction behaves as follows (ST(0) and ST(1) contain the scaled and scaling operands respectively):
    - FSCALE($0,\infty$) generates the invalid operation exception.
    - FSCALE(finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE(finite, $+\infty$) generates ∞ with the same sign as the scaled operand.

    The M8087/M80287 returns zero in the first case and raises the invalid-operation exception in the other cases.
15. The i387 NPX returns signed infinity/zero as the unmasked response to massive overflow/underflow. The M8087 and M80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

3.0 HARDWARE INTERFACE

3.1 Signal Description

In the following signal descriptions, the i387 coprocessor pins are grouped by function as follows:

1. Execution control—386CLK2, 387CLK2, CKM, RESETIN
2. NPX handshake—PEREQ, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$
3. Bus interface pins—D31–D0, $\overline{\text{W/R}}$, $\overline{\text{ADS}}$, $\overline{\text{READY}}$, $\overline{\text{READYO}}$
4. Chip/Port Select—STEN, $\overline{\text{NPS1}}$, NPS2, $\overline{\text{CMD0}}$
5. Power supplies—V_{CC}, V_{SS}

Table 3.1 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. All output signals are tristate; they leave floating state only when STEN is active. The output buffers of the bidirectional data pins D31–D0 are also tristate; they leave floating state only in read cycles when the i387 NPX is selected (i.e. when STEN, $\overline{\text{NPS1}}$, and NPS2 are all active).

Figure 3.1 and Table 3.2 together show the location of every pin in the pin grid array.

Table 3.1. i387™ NPX Pin Summary

Pin Name	Function	Active State	Input/ Output	Referenced To
386CLK2	i386 Microprocessor CLocK 2		I	
387CLK2	i387 NPX CLocK 2		I	
CKM	i387 NPX CLocKing Mode		I	
RESETIN	System reset	High	I	386CLK2
PEREQ	Processor Extension REQuest	High	O	386CLK2/STEN
$\overline{\text{BUSY}}$	Busy status	Low	O	386CLK2/STEN
$\overline{\text{ERROR}}$	Error status	Low	O	387CLK2/STEN
D31–D0	Data pins	High	I/O	386CLK2
$\overline{\text{W/R}}$	Write/Read bus cycle	Hi/Lo	I	386CLK2
$\overline{\text{ADS}}$	ADdress Strobe	Low	I	386CLK2
$\overline{\text{READY}}$	Bus ready input	Low	I	386CLK2
$\overline{\text{READYO}}$	Ready output	Low	O	386CLK2/STEN
STEN	STatus ENable	High	I	386CLK2
$\overline{\text{NPS1}}$	NPX select #1	Low	I	386CLK2
NPS2	NPX select #2	High	I	386CLK2
$\overline{\text{CMD0}}$	CoMmanD	Low	I	386CLK2
V_{CC}			I	
V_{SS}			I	

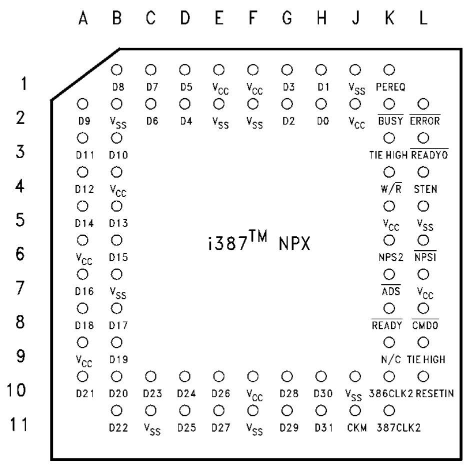
NOTE:
STEN is referenced to only when getting the output pins into or out of tristate mode.

Table 3.2a. i387™ NPX PGA Pin Cross-Reference

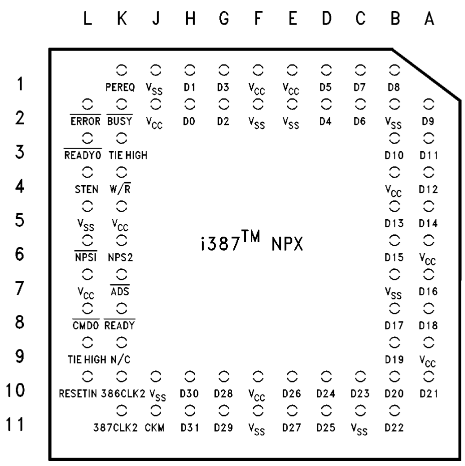
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
A2	D9	B9	D19	F10	V_{CC}	K4	W/$\overline{\text{R}}$
A3	D11	B10	D20	F11	V_{SS}	K5	V_{CC}
A4	D12	B11	D22	G1	D3	K6	NPS2
A5	D14	C1	D7	G2	D2	K7	$\overline{\text{ADS}}$
A6	V_{CC}	C2	D6	G10	D28	K8	$\overline{\text{READY}}$
A7	D16	C10	D23	G11	D29	K9	No Connect
A8	D18	C11	V_{SS}	H1	D1	K10	386CLK2
A9	V_{CC}	D1	D5	H2	D0	K11	387CLK2
A10	D21	D2	D4	H10	D30	L2	$\overline{\text{ERROR}}$
B1	D8	D10	D24	H11	D31	L3	$\overline{\text{READYO}}$
B2	V_{SS}	D11	D25	J1	V_{SS}	L4	STEN
B3	D10	E1	V_{CC}	J2	V_{CC}	L5	V_{SS}
B4	V_{CC}	E2	V_{SS}	J10	V_{SS}	L6	$\overline{\text{NPS1}}$
B5	D13	E10	D26	J11	CKM	L7	V_{CC}
B6	D15	E11	D27	K1	PEREQ	L8	$\overline{\text{CMD0}}$
B7	V_{SS}	F1	V_{CC}	K2	$\overline{\text{BUSY}}$	L9	Tie High
B8	D17	F2	V_{SS}	K3	Tie High	L10	RESETIN

Table 3.2b. i387™ Math Coprocessor CQFP Pin Cross-Reference

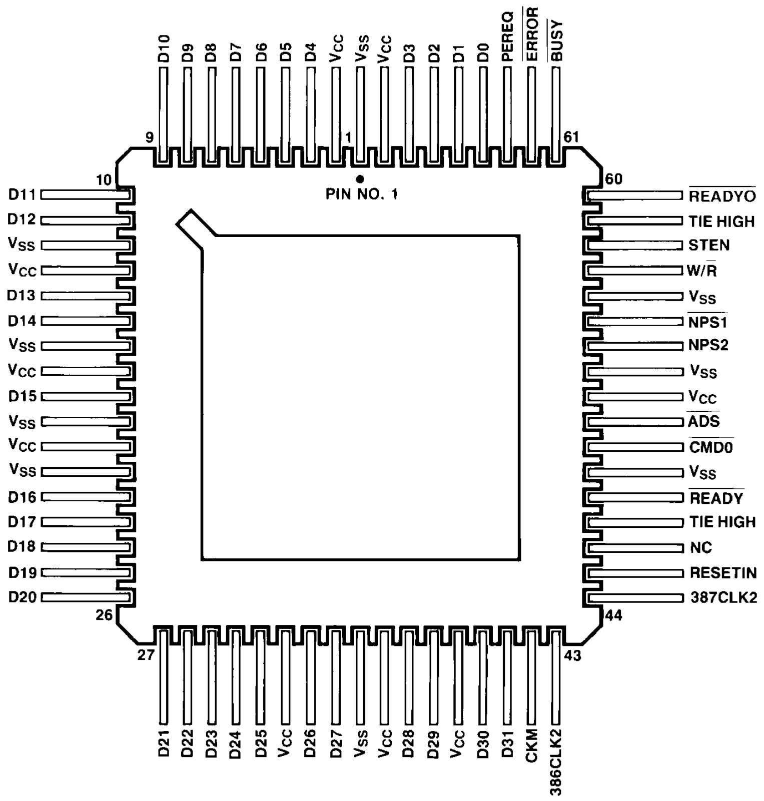
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	V_{SS}	18	D15	35	V_{SS}	52	V_{CC}
2	V_{CC}	19	V_{SS}	36	V_{CC}	53	V_{SS}
3	D4	20	V_{CC}	37	D28	54	NPS2
4	D5	21	V_{SS}	38	D29	55	$\overline{\text{NPS1}}$
5	D6	22	D16	39	V_{CC}	56	V_{SS}
6	D7	23	D17	40	D30	57	W/$\overline{\text{R}}$
7	D8	24	D18	41	D31	58	STEN
8	D9	25	D19	42	CKM	59	Tie High
9	D10	26	D20	43	386CLK2	60	$\overline{\text{READYO}}$
10	D11	27	D21	44	387CLK2	61	$\overline{\text{BUSY}}$
11	D12	28	D22	45	RESETIN	62	$\overline{\text{ERROR}}$
12	V_{SS}	29	D23	46	NC	63	PEREQ
13	V_{CC}	30	D24	47	Tie High	64	D0
14	D13	31	D25	48	$\overline{\text{READY}}$	65	D1
15	D14	32	V_{CC}	49	V_{SS}	66	D2
16	V_{SS}	33	D26	50	$\overline{\text{CMD0}}$	67	D3
17	V_{CC}	34	D27	51	$\overline{\text{ADS}}$	68	V_{CC}


A B C D E F G H J K L
1 D8 D7 D5 VCC VCC D3 D1 VSS PEREQ
2 D9 VSS D6 D4 VSS VSS D2 D0 VCC BUSY ERROR
3 D11 D10 TIE HIGH READYO
4 D12 VCC W/R STEN
5 D14 D13 VCC VSS
6 VCC D15 NPS2 NPS1
7 D16 VSS ADS VCC
8 D18 D17 READY CMD0
9 VCC D19 N/C TIE HIGH
10 D21 D20 D23 D24 D26 VCC D28 D30 VSS 386CLK2 RESETIN
11 D22 VSS D25 D27 VSS D29 D31 CKM 387CLK2
i387™ NPX

271074–5
i387 Math Coprocessor PGA Pinout—View from Pin Side


L K J H G F E D C B A
1 PEREQ VSS D1 D3 VCC VCC D5 D7 D8
2 ERROR BUSY VCC D0 D2 VSS VSS D4 D6 VSS D9
3 READYO TIE HIGH D10 D11
4 STEN W/R VCC D12
5 VSS VCC D13 D14
6 NPS1 NPS2 D15 VCC
7 VCC ADS VSS D16
8 CMD0 READY D17 D18
9 TIE HIGH N/C D19 VCC
10 RESETIN 386CLK2 VSS D30 D28 VCC D26 D24 D23 D20 D21
11 387CLK2 CKM D31 D29 VSS D27 D25 VSS D22
i387™ NPX

271074–20
i387 Math Coprocessor PGA Pinout—View from Top Side


D10 D9 D8 D7 D6 D5 D4 VCC VSS VCC D3 D2 D1 D0 PEREQ ERROR BUSY
9
1
61
10
60
PIN NO. 1
D11 D12 VSS VCC D13 D14 VSS VCC D15 VSS VCC VSS D16 D17 D18 D19 D20
READYO TIE HIGH STEN W/R VSS NPS1 NPS2 VSS VCC ADS CMD0 VSS READY TIE HIGH NC RESETIN 387CLK2
26
44
27
43
D21 D22 D23 D24 D25 VCC D26 D27 VSS VCC D28 D29 VCC D30 D31 CKM 386CLK2

271074–19

MQ80387
68-Lead Quad Flat Pack
Top View

Figure 3.1. i387™ NPX Pin Configuration

3.1.1 M80386 CLOCK 2 (386CLK2)

This input uses the M80386 CLK2 signal to time the bus control logic. Several other M80387 signals are referenced to the rising edge of this signal. When CKM = 1 (synchronous mode) this pin also clocks the data interface and control unit and the floating-point unit of the M80387. This pin requires MOS-level input. The signal on this pin is divided by two to produce the internal clock signal CLK.

3.1.2 M80387 CLOCK 2 (387CLK2)

When CKM = 0 (asynchronous mode) this pin provides the clock for the data interface and control unit and the floating-point unit of the M80387. In this case, the ratio of the frequency of 387CLK2 to the frequency of 386CLK2 must lie within the range 10:16 to 14:10. When CKM = 1 (synchronous mode) this pin is ignored; 386CLK2 is used instead for the data interface and control unit and the floating-point unit. This pin requires TTL-level input.

3.1.3 M80387 CLOCKING MODE (CKM)

This pin is a strapping option. When it is strapped to V_{CC}, the i387 NPX operates in synchronous mode; when strapped to V_{SS}, the i387 NPX operates in asynchronous mode. These modes relate to clocking of the data interface and control unit and the floating-point unit only; the bus control logic always operates synchronously with respect to the i386 processor.

3.1.4 SYSTEM RESET (RESETIN)

A LOW to HIGH transition on this pin causes the i387 NPX to terminate its present activity and to enter a dormant state. RESETIN must remain HIGH for at least 40 387CLK2 periods. The HIGH to LOW transitions of RESETIN must be synchronous with 386CLK2, so that the phase of the internal clock of the bus control logic (which is the 386CLK2 divided by 2) is the same as the phase of the internal clock of the i386 microprocessor. After RESETIN goes LOW, at least 50 387CLK2 periods must pass before the first NPX instruction is written into the i387 coprocessor. This pin should be connected to the i386 microprocessor RESET pin. Table 3.3 shows the status of other pins after a reset.

Table 3.3. Output Pin Status during Reset

Pin Value	Pin Name
HIGH	$\overline{\text{READYO}}$, $\overline{\text{BUSY}}$
LOW	PEREQ, $\overline{\text{ERROR}}$
Tri-State OFF	D31–D0

3.1.5 PROCESSOR EXTENSION REQUEST (PEREQ)

When active, this pin signals to the i386 CPU that the i387 NPX is ready for data transfer to/from its data FIFO. When all data is written to or read from the data FIFO, PEREQ is deactivated. This signal always goes inactive before $\overline{\text{BUSY}}$ goes inactive. This signal is referenced to 386CLK2. It should be connected to the i386 microprocessor PEREQ input. Refer to Figure 3.7 for the timing relationships between this and the $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ pins.

3.1.6 BUSY STATUS ($\overline{\text{BUSY}}$)

When active, this pin signals to the i386 CPU that the i387 NPX is currently executing an instruction. This signal is referenced to 386CLK2. It should be connected to the i386 microprocessor $\overline{\text{BUSY}}$ pin. Refer to Figure 3.7 for the timing relationships between this and the PEREQ and $\overline{\text{ERROR}}$ pins.

3.1.7 ERROR STATUS ($\overline{\text{ERROR}}$)

This pin reflects the ES bits of the status register. When active, it indicates that an unmasked exception has occurred (except that, immediately after a reset, it indicates to the i386 microprocessor that an i387 NPX is present in the system). This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, and FNSAVE. This signal is referenced to 387CLK2. It should be connected to the i386 microprocessor $\overline{\text{ERROR}}$ pin. Refer to Figure 3.7 for the timing relationships between this and the PEREQ and $\overline{\text{BUSY}}$ pins.

3.1.8 DATA PINS (D31–D0)

These bidirectional pins are used to transfer data and opcodes between the i386 microprocessor and i387 NPX. They are normally connected directly to the corresponding i386 processor data pins. HIGH state indicates a value of one. D0 is the least significant data bit. Timings are referenced to 386CLK2.

3.1.9 WRITE/READ BUS CYCLE (W/$\overline{\text{R}}$)

This signal indicates to the i387 NPX whether the i386 processor bus cycle in progress is a read or a write cycle. This pin should be connected directly to the i386 processor W/$\overline{\text{R}}$ pin. HIGH indicates a write cycle; LOW, a read cycle. This input is ignored if any of the signals STEN, $\overline{\text{NPS1}}$, or NPS2 is inactive. Setup and hold times are referenced to 386CLK2.

3.1.10 ADDRESS STROBE ($\overline{\text{ADS}}$)

This input, in conjunction with the $\overline{\text{READY}}$ input indicates when the i387 NPX bus-control logic may sample W/$\overline{\text{R}}$ and the chip-select signals. Setup and hold times are referenced to 386CLK2. This pin should be connected to the i386 processor $\overline{\text{ADS}}$ pin.

3.1.11 BUS READY INPUT ($\overline{\text{READY}}$)

This input indicates to the i387 NPX when an i386 processor bus cycle is to be terminated. It is used by the bus-control logic to trace bus activities. Bus cycles can be extended indefinitely until terminated by $\overline{\text{READY}}$. This input should be connected to the same signal that drives the i386 processor $\overline{\text{READ}}$ input. Setup and hold times are referenced to 386CLK2.

3.1.12 READY OUTPUT ($\overline{\text{READYO}}$)

This pin is activated at such a time that write cycles are terminated after two clocks and read cycles after three clocks. In configurations where no extra wait states are required, it can be used to directly drive the i386 processor $\overline{\text{READY}}$ input. Refer to section 3.4 "Bus Operation" for details. This pin is activated only during bus cycles that select the i387 NPX. This signal is referenced to 386CLK2.

3.1.13 STATUS ENABLE (STEN)

This pin serves as a chip select for the i387 NPX. When inactive, this pin forces $\overline{\text{BUSY}}$, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{READYO}}$ outputs into floating state. D31–D0 are normally floating and leave floating state only if STEN is active and additional conditions are met. STEN also causes the chip to recognize its other chip-select inputs. STEN makes it easier to do on-board testing (using the overdrive method) of other chips in systems containing the i387 NPX. STEN should be pulled up with a resistor so that it can be pulled down when testing. In boards that do not use on-board testing, STEN should be connected to V_{CC}. Setup and hold times are relative to 386CLK2. Note that STEN must maintain the same setup and hold times as $\overline{\text{NPS1}}$, NPS2, and $\overline{\text{CMD0}}$ (i.e. if STEN changes state during an M80387 bus cycle, it should change state during the same CLK period as the $\overline{\text{NPS1}}$, NPS2, and $\overline{\text{CMD0}}$ signals).

3.1.14 NPX SELECT #1 ($\overline{\text{NPS1}}$)

When active (along with STEN and NPS2) in the first period of an i386 microprocessor bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the i387 NPX. This pin should be connected directly to the i386 processor M/$\overline{\text{IO}}$ pin, so that the i387 NPX is selected only when the M80386 performs I/O cycles. Setup and hold times are referenced to 386CLK2.

3.1.15 NPX SELECT #2 (NPS2)

When active (along with STEN and $\overline{\text{NPS1}}$) in the first period of an i386 processor bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the i387 NPX. This pin should be connected directly to the i386 microprocessor A31 pin, so that the i387 NPX is selected only when the i386 processor uses one of the I/O addresses reserved for the i387 NPX (800000F8 or 800000FC). Setup and hold times are referenced to 386CLK2.

3.1.16 COMMAND ($\overline{\text{CMD0}}$)

During a write cycle, this signal indicates whether an opcode ($\overline{\text{CMD0}}$ active) or data ($\overline{\text{CMD0}}$ inactive) is being sent to the i387 NPX. During a read cycle, it indicates whether the control or status register ($\overline{\text{CMD0}}$ active) or a data register ($\overline{\text{CMD0}}$ inactive) is being read. $\overline{\text{CMD0}}$ should be connected directly to the A2 output of the i386 microprocessor. Setup and hold times are referenced to 386CLK2.

3.2 Processor Architecture

As shown by the block diagram on the front page, the NPX is internally divided into three sections: the bus control logic (BCL), the data interface and control unit, and the floating point unit (FPU). The FPU (with the support of the control unit which contains the sequencer and other support units) executes all numerics instructions. The data interface and control unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, and sequencing the microinstructions, and for handling some of the administrative instructions. The BCL is responsible for i386 processor bus tracking and interface. The BCL is the only unit in the i387 NPX that must run synchronously with the i386 processor; the rest of the i387 NPX can run asynchronously with respect to the i386 processor.

3.2.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from memory to the i387 NPX and transferring outputs from the i387 NPX to memory.

3.2.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, or FSTCW, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the $\overline{\text{BUSY}}$, PEREQ and $\overline{\text{ERROR}}$ signals that synchronize i387 NPX activities with the i386 processor. It also supports the FPU in all operations that it cannot perform alone (e.g. exceptions handling, transcendental operations, etc.).

3.2.3 FLOATING POINT UNIT

The FPU executes all instructions that involve the register stack, including arithmetic, logical, transcendental, constant, and data transfer instructions. The data path in the FPU is 84 bits wide (68 significant bits, 15 exponent bits, and a sign bit) which allows internal operand transfers to be performed at very high speeds.

3.3 System Configuration

As an extension to the i386 processor, the i387 NPX can be connected to the CPU as shown by Figure 3.2. A dedicated communication protocol makes

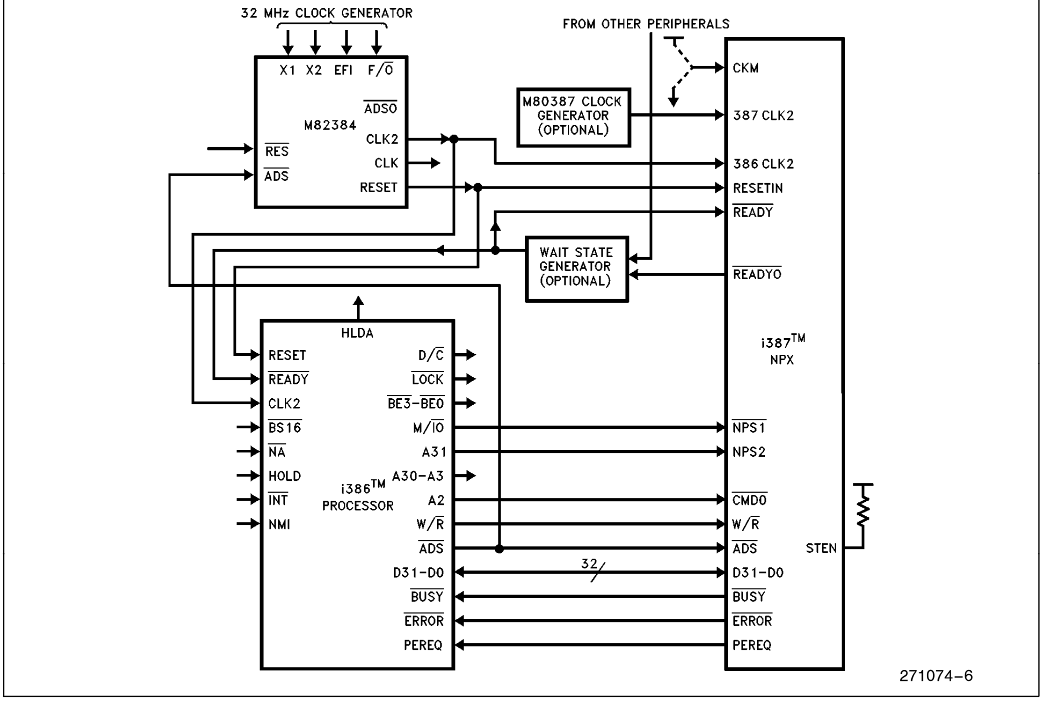


Figure 3.2. i386™/i387™ Processors System Configuration

Table 3.4. Bus Cycles Definition

STEN	$\overline{\text{NPS1}}$	NPS2	$\overline{\text{CMD0}}$	$\overline{\text{W}}/\text{R}$	Bus Cycle Type
0	x	x	x	x	i387 NPX not selected and all outputs in floating state
1	1	x	x	x	i387 NPX not selected
1	x	0	x	x	i387 NPX not selected
1	0	1	0	0	CW or SW read from i387 NPX
1	0	1	0	1	Opcode write to i387 NPX
1	0	1	1	0	Data read from i387 NPX
1	0	1	1	1	Data write to i387 NPX

possible high-speed transfer of opcodes and operands between the i386 microprocessor and i387 NPX. The i387 NPX is designed so that no additional components are required for interface with the i386 processor. The i387 NPX shares the 32-bit wide local bus of the i386 processor and most control pins of the i387 NPX are connected directly to pins of the i386 processor.

3.3.1 BUS CYCLE TRACKING

The $\overline{\text{ADS}}$ and $\overline{\text{READY}}$ signals allow the i387 NPX to track the beginning and end of i386 processor bus cycles, respectively. When $\overline{\text{ADS}}$ is asserted at the same time as the i387 NPX chip-select inputs, the bus cycle is intended for the i387 NPX. To signal the end of a bus cycle for the i387 NPX, $\overline{\text{READY}}$ may be asserted directly or indirectly by the i387 NPX or by other bus-control logic. Refer to Table 3.4 for definition of the types of i387 NPX bus cycles.

3.3.2 i387™ NPX ADDRESSING

The $\overline{\text{NPS1}}$, NPS2 and STEN signals allow the NPX to identify which bus cycles are intended for the NPX. The NPX responds only to I/O cycles when bit 31 of the I/O address is set. In other words, the NPX acts as an I/O device in a reserved I/O address space.

Because A_{31} is used to select the i387 NPX for data transfers, it is not possible for a program running on the i386 processor to address the i387 NPX with an I/O instruction. Only ESC instructions cause the i386 processor to communicate with the i387 NPX. The i386 processor $\overline{\text{BS16}}$ input must be inactive during I/O cycles when A_{31} is active.

3.3.3 FUNCTION SELECTION

The $\overline{\text{CMD0}}$ and $\overline{\text{W}}/\text{R}$ signals identify the four kinds of bus cycle: control or status register read, data read, opcode write, data write.

3.3.4 CPU/NPX Synchronization

The pin pairs $\overline{\text{BUSY}}$, PEREQ, and $\overline{\text{ERROR}}$ are used for various aspects of synchronization between the CPU and the NPX.

$\overline{\text{BUSY}}$ is used to synchronize instruction transfer from the i386 processor to the i387 NPX. When the i387 NPX recognizes an ESC instruction, it asserts $\overline{\text{BUSY}}$. For most ESC instructions, the i386 processor waits for the i387 NPX to deassert $\overline{\text{BUSY}}$ before sending the new opcode.

The NPX uses the PEREQ pin of the i386 CPU to signal that the NPX is ready for data transfer to or from its data FIFO. The NPX does not directly access memory; rather, the i386 processor provides memory access services for the NPX. Thus, memory access on behalf of the NPX always obeys the rules applicable to the mode of the i386 processor, whether the i386 processor be in real-address mode or protected mode.

Once the i386 processor initiates an i387 NPX instruction that has operands, the i386 microprocessor waits for PEREQ signals that indicate when the i387 NPX is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the i386 microprocessor continues program execution while the i387 NPX executes the ESC instruction.

In M8086/M8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In M80286/M80287 and i386/i387 processor systems, WAIT instructions are required only for operand synchronization; namely, after NPX stores to memory (except FSTSW and FSTCW) or loads from memory. Used this way, WAIT ensures that the value has already been written or read by the NPX before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred the operands from the i386 processor, the i387 NPX can process the instruction in parallel with and independent of the host CPU. When the NPX detects an exception, it asserts the $\overline{\text{ERROR}}$ signal, which causes an i386 processor interrupt.

3.3.5 SYNCHRONOUS OR ASYNCHRONOUS MODES

The internal logic of the i387 NPX (the FPU) can either operate directly from the CPU clock (synchronous mode) or from a separate clock (asynchronous mode). The two configurations are distinguished by the CKM pin. In either case, the bus control logic (BCL) of the i387 NPX is synchronized with the CPU clock. Use of asynchronous mode allows the i386 microprocessor and the FPU section of the i387 NPX to run at different speeds. In this case, the ratio of the frequency of 387CLK2 to the frequency of 386CLK2 must lie within the range 10:16 to 16:10. Use of synchronous mode eliminates one clock generator from the board design.

3.3.6 AUTOMATIC BUS CYCLE TERMINATION

In configurations where no extra wait states are required, $\overline{\text{READYO}}$ can be used to drive the i386 microprocessor $\overline{\text{READY}}$ input. If this pin is used, it should be connected to the logic that ORs all READY outputs from peripherals on the i386 microprocessor bus. $\overline{\text{READYO}}$ is asserted by the i387 NPX only during I/O cycles that select the i387 NPX. Refer to section 3.4 "Bus Operation" for details.

3.4 Bus Operation

With respect to the bus interface, the i387 NPX is fully synchronous with the i386 processor. Both operate at the same rate, because each generates its internal CLK signal by dividing 386CLK2 by two.

The i386 processor initiates a new bus cycle by activating $\overline{\text{ADS}}$. The i387 NPX recognizes a bus cycle, if, during the cycle in which $\overline{\text{ADS}}$ is activated, STEN, $\overline{\text{NPS1}}$, and NPS2 are all activated. Proper operation is achieved if $\overline{\text{NPS1}}$ is connected to the M/$\overline{\text{IO}}$ output of the i386 processor, and NPS2 to the A31 output. The i386 processor's A31 output is guaranteed to be inactive in all bus cycles that do not address the i387 NPX (i.e. I/O cycles to other devices, interrupt acknowledge, and reserved types of bus cycles). System logic must not signal a 16-bit bus cycle via the i386 processor $\overline{\text{BS16}}$ input during I/O cycles when A31 is active.

During the CLK period in which $\overline{\text{ADS}}$ is activated, the i387 NPX also examines the W/$\overline{\text{R}}$ input signal to determine whether the cycle is a read or a write cycle and examines the $\overline{\text{CMD0}}$ input to determine whether an opcode, operand, or control/status register transfer is to occur.

The i387 NPX supports both pipelined and nonpipelined bus cycles. A nonpipelined cycle is one for which the i386 processor asserts $\overline{\text{ADS}}$ when no other i387 NPX bus cycle is in progress. A pipelined bus cycle is one for which the i386 processor asserts $\overline{\text{ADS}}$ and provides valid next-address and control signals as soon as in the second CLK period after the $\overline{\text{ADS}}$ assertion for the previous i386 processor bus cycle. Pipelining increases the availability of the bus by at least one CLK period. The i387 NPX supports pipelined bus cycles in order to optimize address pipelining by the i386 processor for memory cycles.

Bus operation is described in terms of an abstract *state machine*. Figure 3.3 illustrates the states and state transitions for i387 NPX bus cycles:

- T_I is the idle state. This is the state of the bus logic after RESET, the state to which bus logic returns after evey nonpipelined bus cycle, and the state to which bus logic returns after a series of pipelined cycles.
- T_{RS} is the $\overline{\text{READY}}$ sensitive state. Different types of bus cycle may require a minimum of one or two successive T_{RS} states. The bus logic remains in T_{RS} state until $\overline{\text{READY}}$ is sensed, at which point the bus cycle terminates. Any number of wait states may be implemented by delaying $\overline{\text{READY}}$, thereby causing additional successive T_{RS} states.
- T_P is the first state for every pipelined bus cycle.

The $\overline{\text{READYO}}$ output of the i387 NPX indicates when a bus cycle for the i387 NPX may be terminated if no extra wait states are required. For all write cycles (except those for the instructions FLDENV and FRSTOR), $\overline{\text{READYO}}$ is always assert-

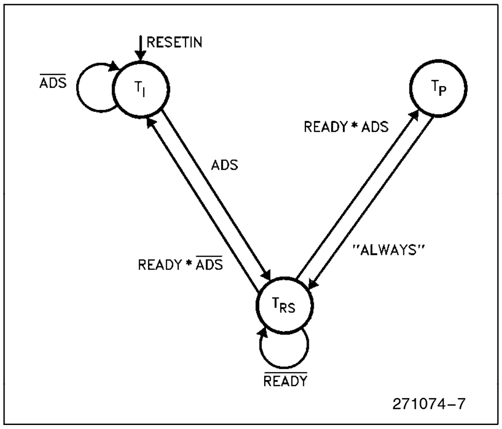


Figure 3.3. Bus State Diagram

ed in the first T_{RS} state, regardless of the number of wait states. For all read cycles and write cycles for FLDENV and FRSTOR, $\overline{\text{READYO}}$ is always asserted in the second T_{RS} state, regardless of the number of wait states. These rules apply to both pipelined and nonpipelined cycles. Systems designers may use $\overline{\text{READYO}}$ in one of three ways:

1. Leave it disconnected and use external logic to generate $\overline{\text{READY}}$ signals. When choosing this option, i387 NPX requirements for wait states in read cycles and write cycles of FLDENV and FRSTOR must be obeyed.
2. Connect it (directly or through logic that ORs READY signals from other devices) to the $\overline{\text{READY}}$ inputs of the i386 processor and i387 NPX.
3. Use it as one input to a wait-state generator.

The following sections illustrate different types of i387 NPX bus cycles.

Because different instructions have different amounts of overhead before, between, and after operand transfer cycles, it is not possible to represent in a few diagrams all of the combinations of successive operand transfer cycles. The following bus-cycle diagrams show memory cycles between i387 NPX operand-transfer cycles. Note however that, during the instructions FLDENV, FSTENV, FSAVE, and FRSTOR, some consecutive accesses to the NPX do not have intervening memory accesses. For the timing relationship between operand transfer cycles and opcode write or other overhead activities, see Figure 3.7.

3.4.1 NONPIPELINED BUS CYCLES

Figure 3.4 illustrates bus activity for consecutive nonpipelined bus cycles.

3.4.1.1 Write Cycle

At the second clock of the bus cycle, the M80387 enters the T_{RS} ($\overline{\text{READY}}$-sensitive) state. During this state, the i387 NPX samples the $\overline{\text{READY}}$ input and stays in this state as long as $\overline{\text{READY}}$ is inactive.

In write cycles, the i387 NPX drives the $\overline{\text{READYO}}$ signal for one CLK period beginning with the second CLK of the bus cycle; therefore, the fastest write cycle takes two CLK cycles (see cycle 2 of Figure 3.4). For the instructions FLDENV and FRSTOR, however, the i387 NPX forces a wait state by delaying the activation of $\overline{\text{READYO}}$ to the second T_{RS} cycle (not shown in Figure 3.4).

When $\overline{\text{READY}}$ is asserted the M80387 returns to the idle state, in which $\overline{\text{ADS}}$ could be asserted again by the i386 processor for the next cycle.

3.4.1.2 Read Cycle

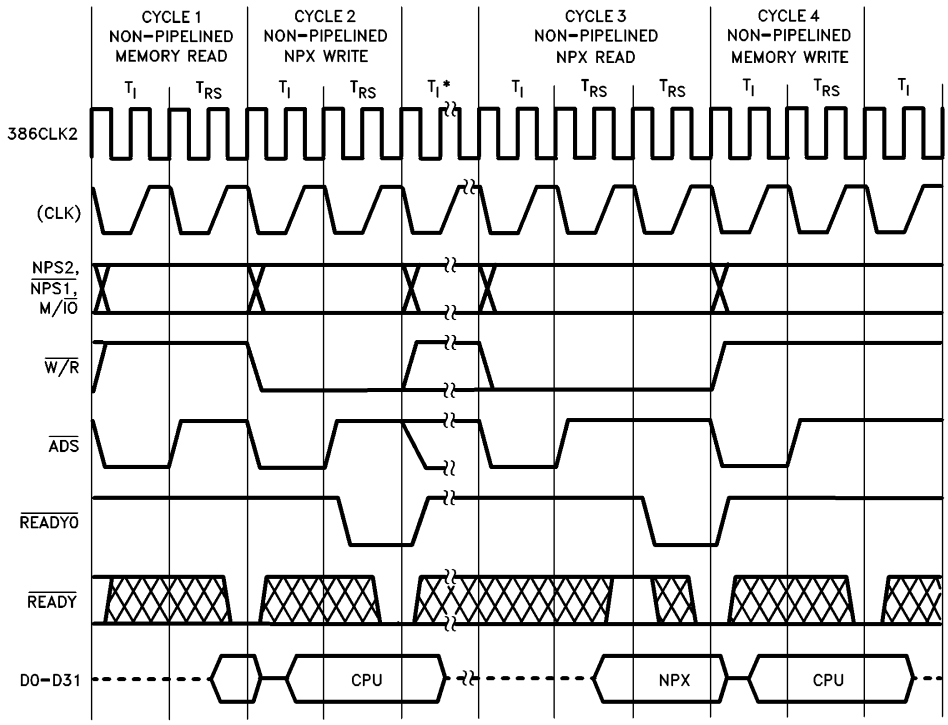
At the second clock of the bus cycle, the i387 NPX enters the T_{RS} state. See Figure 3.4. In this state, the i387 NPX samples the $\overline{\text{READY}}$ input and stays in this state as long as $\overline{\text{READY}}$ is inactive.

At the rising edge of CLK in the second clock period of the cycle, the i387 NPX starts to drive the D31–D0 outputs and continues to drive them as long as it stays in T_{RS} state.

In read cycles that address the i387 NPX, at least one wait state must be inserted to insure that the i386 microprocessor latches the correct data. Since the i387 NPX starts driving the system data bus only at the rising edge of CLK in the second clock period of the bus cycle, not enough time is left for the data signals to propagate and be latched by the i386 processor at the falling edge of the same clock period. The i387 NPX drives the $\overline{\text{READYO}}$ signal for one CLK period in the third CLK of the bus cycle. Therefore, if the $\overline{\text{READYO}}$ output is used to drive the i386 processor $\overline{\text{READY}}$ input, one wait state is inserted automatically.

Because one wait state is required for i387 NPX reads, the minimum is three CLK cycles per read, as cycle 3 of Figure 3.4 shows.

When $\overline{\text{READY}}$ is asserted the i387 NPX returns to the idle state, in which $\overline{\text{ADS}}$ could be asserted again by the i386 processor for the next cycle. The transition from T_{RS} state to idle state causes the i387 NPX to put the tristate D31–D0 outputs into the floating state, allowing another device to drive the system data bus.


CYCLE 1 NON-PIPELINED MEMORY READ
CYCLE 2 NON-PIPELINED NPX WRITE
CYCLE 3 NON-PIPELINED NPX READ
CYCLE 4 NON-PIPELINED MEMORY WRITE
T_I
T_RS
T_I*
386CLK2
(CLK)
NPS2, NPS1, M/IO
W/R
ADS
READYO
READY
D0-D31
CPU
NPX

271074-8

Cycles 1 & 2 represent part of the operand transfer cycle for instructions involving either 4-byte or 8-byte operand loads.
Cycles 3 & 4 represent part of the operand transfer cycle for a store operation.
*Cycles 1 & 2 could repeat here or T_I states for various non-operand transfer cycles and overhead.

Figure 3.4. Nonpipelined Read and Write Cycles

3.4.2 PIPELINED BUS CYCLES

Because all the activities of the i387 NPX bus interface occur either during the T_{RS} state or during the transitions to or from that state, the only difference between a pipelined and a nonpipelined cycle is the manner of changing from one state to another. The exact activities in each state are detailed in the previous section "Nonpipelined Bus Cycles".

When the i386 processor asserts $\overline{ADS}$ before the end of a bus cycle, both $\overline{ADS}$ and $\overline{READY}$ are active during a T_{RS} state. This condition causes the i387 NPX to change to a different state named T_P. The i387 NPX activities in the transition from a T_{RS} state to a T_P state are exactly the same as those in the transition from a T_{RS} state to a T_I state in non-pipelined cycles. T_P state is metastable; therefore, one clock period later the i387 NPX returns to T_{RS} state. In consecutive pipelined cycles, the i387 NPX bus logic uses only T_{RS} and T_P states.

Figure 3.5 shows the fastest transition into and out of the pipelined bus cycles. Cycle 1 in this figure represents a nonpipelined cycle. (Nonpipelined write cycles with only one T_{RS} state (i.e. no wait states) are always followed by another nonpipelined cycle, because $\overline{READY}$ is asserted before the earliest possible assertion of $\overline{ADS}$ for the next cycle.)

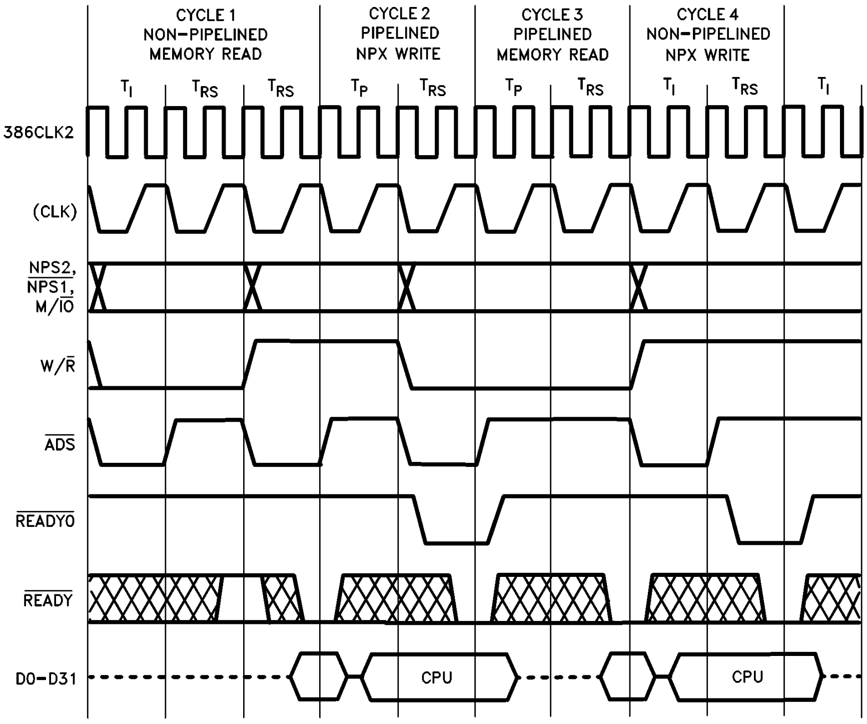
Figure 3.6 shows the pipelined write and read cycles with one additional T_{RS} states beyond the minimum required. To delay the assertion of $\overline{READY}$ requires external logic.

3.4.3 BUS CYCLES OF MIXED TYPE

When the i387 NPX bus logic is in the T_{RS} state, it distinguishes between nonpipelined and pipelined cycles according to the behavior of $\overline{ADS}$ and $\overline{READY}$. In a nonpipelined cycle, only $\overline{READY}$ is activated, and the transition is from T_{RS} to idle state. In a pipelined cycle, both $\overline{READY}$ and $\overline{ADS}$ are active and the transition is first from T_{RS} state to T_P state then, after one clock period, back to T_{RS} state.

3.4.4 $\overline{BUSY}$ AND PEREQ TIMING RELATIONSHIP

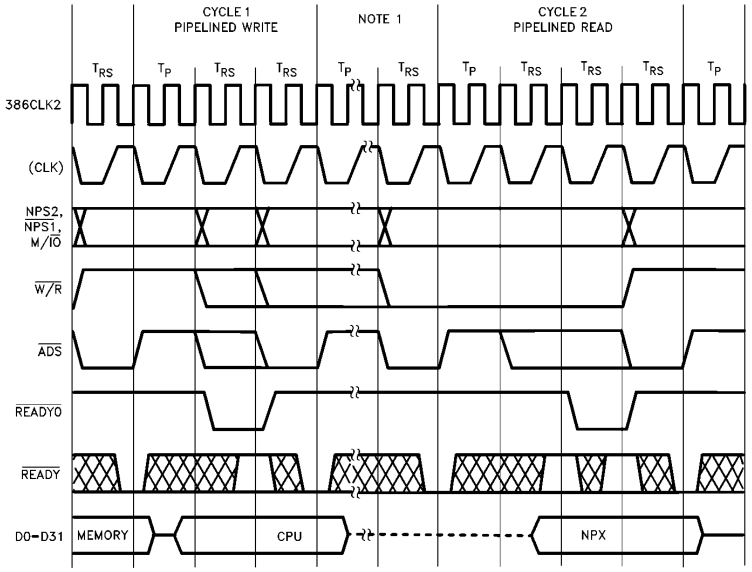
Figure 3.7 shows the activation of $\overline{BUSY}$ at the beginning of instruction execution and its deactivation after execution of the instruction is complete. PEREQ is activated in this interval. If $\overline{ERROR}$ (not shown in the diagram) is ever asserted, it would occur at least six 386CLK2 periods after the deactivation of PEREQ and at least six 386CLK2 periods before the deactivation of $\overline{BUSY}$. Figure 3.7 shows also that STEN is activated at the beginning of a bus cycle.



271074-9

Cycle 1–Cycle 4 represent the operand transfer cycle for an instruction involving a transfer of two 32-bit loads in total.
The opcode write cycles and other overhead are not shown.
Note that the next cycle will be a pipelined cycle if both $\overline{READY}$ and $\overline{ADS}$ are sampled active at the end of a T_{RS} state of the current cycle.

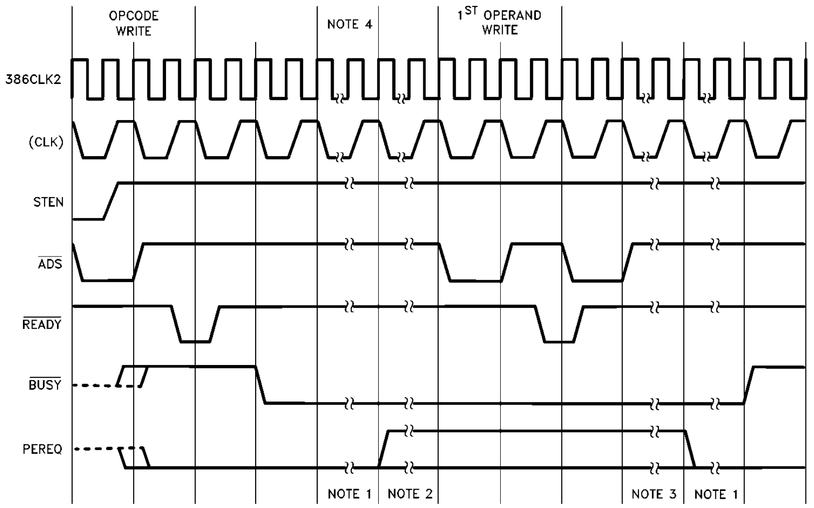
Figure 3.5. Fastest Transitions to and from Pipelined Cycles



271074–10

NOTE:
1. Cycles between operand write to the NPX and storing result.

Figure 3.6. Pipelined Cycles with Wait States



271074–11

NOTES:
1. Instruction dependent.
2. PEREQ is an asynchronous input to the i386 processor; it may not be asserted (instruction dependent).
3. More operand transfers.
4. Memory read (operand) cycle is not shown.

Figure 3.7. STEN, BUSY and PEREQ Timing Relationship

5.0 ELECTRICAL DATA

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

5.1 Absolute Maximum Ratings*

Case Temperature Under Bias	−55°C to +125°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−0.5 to V_{CC} +0.5V
Power Dissipation	1.5W

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

OPERATING CONDITIONS

MIL-STD-883

Table 5.1

Symbol	Description	Min	Max	Units
T_C	Case Temperature (Instant On)	−55	+125	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V

Extended Temperature

Symbol	Description	Min	Max	Units
T_C	Case Temperature (Instant On)	−40	+110	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V

Military Temperature Only (MTO)

Symbol	Description	Min	Max	Units
T_C	Case Temperature (Instant On)	−55	+125	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V

5.2 DC Characteristics (Over Specified Operating Conditions)

Table 5.2. DC Specifications

Symbol	Parameter	Min	Max	Units	Comments
V_{IL}	Input LO Voltage	−0.3	+0.8	V	(Note 1)
V_{IH}	Input HI Voltage	2.0	V_{CC} + 0.3	V	(Note 1)
V_{CL}	386CLK2 Input LO Voltage	−0.3	+0.8	V	
V_{CH}	386CLK2 Input HI Voltage	3.7	V_{CC} + 0.3	V	
V_{OL}	Output LO Voltage		0.45	V	(Note 2)
V_{OH}	Output HI Voltage	2.4		V	(Note 3)
I_{CC}	Power Supply Current	150 (typical)	250	mA	387CLK2 = 32 MHz(4)
		190 (typical)	310		387CLK2 = 40 MHz(4)
		250 (typical)	390		387CLK2 = 50 MHz(4)
I_{LI}	Input Leakage Current		±15	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	I/O Leakage Current		±15	μA	$0.45V \le V_O \le V_{CC}$
C_{IN}	Input Capacitance		10	pF	fc = 1 MHz
C_O	I/O or Output Capacitance		12	pF	fc = 1 MHz
C_{CLK}	Clock Capacitance		20	pF	fc = 1 MHz

NOTES:
1. This parameter is for all inputs, including 387CLK2 but excluding 386CLK2.
2. This parameter is measured at I_{OL} as follows:
   data = 4.0 mA
   $\overline{\text{READYO}}$ = 2.5 mA
   $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, PEREQ = 2.5 mA
3. This parameter is measured at I_{OH} as follows:
   data = 1.0 mA
   $\overline{\text{READYO}}$ = 0.6 mA
   $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, PEREQ = 0.6 mA
4. I_{CC} is measured at steady state, maximum capacitive loading on the outputs, and worst-case DC level at the inputs; 386CLK2 at the same frequency as 387CLK2.

5.3 AC Characteristics (Over Specified Operating Conditions)

Output Trip Level: 1.5V

Table 5.3a. Combinations of Bus Interface and Execution Speeds

Functional Block	Speed Combinations		
	i387-16	i387-20	i387-25
Bus Interface Unit (MHz)	16	20	25
Execution Unit (MHz)	16	20	25

Table 5.3b. Timing Requirements of the Execution Unit

Pin	Symbol	Parameter	16 MHz 1.5V		20 MHz 1.5V		25 MHz 1.5V		Comments
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
387CLK2	t1	Period	31	125	25	125	20	125	2.0V
387CLK2	t2a	High Time	9		8		7		2.0V
387CLK2	t2b	High Time	5		5		4		3.7V
387CLK2	t3a	Low Time	9		8		7		2.0V
387CLK2	t3b	Low Time	7		6		5		0.8V
387CLK2	t4	Fall Time		8		8		7	3.7V to 0.8V
387CLK2	t5	Rise Time		8		8		7	0.8V to 3.7V

Table 5.3c. Timing Requirements of the Bus Interface Unit

Pin	Symbol	Parameter	16 MHz 1.5V		20 MHz 1.5V		25 MHz 1.5V		Comments
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
386CLK2	t1	Period	31	125	25	125	20	125	2.0V
386CLK2	t2a	High Time	9		8		7		2.0V
386CLK2	t2b	High Time	5		5		4		3.7V
386CLK2	t3a	Low Time	9		8		7		2.0V
386CLK2	t3b	Low Time	7		6		5		0.8V
386CLK2	t4	Fall Time		8		8		7	3.7V to 0.8V
386CLK2	t5	Rise Time		8		8		7	0.8V to 3.7V
386CLK2/ 387CLK2		Ratio	10/16	14/10	10/16	14/10	10/16	14/10	
$\overline{\text{READYO}}$	t7	Out Delay	4	34	3	31	3	24	C_L = 75 pF*
$\overline{\text{READYO}}$	t7	Out Delay	4	31	3	27	3	21	C_L = 25 pF
PEREQ	t7	Out Delay	5	34	5	34	4	33	C_L = 75 pF*
$\overline{\text{BUSY}}$	t7	Out Delay	5	34	5	29	4	29	C_L = 75 pF*
$\overline{\text{BUSY}}$	t7	Out Delay	N/A	N/A	N/A	N/A	4	27	C_L = 25 pF
$\overline{\text{ERROR}}$	t7	Out Delay	5	34	5	34	4	33	C_L = 75 pF*

*C_L = 50 pF for 25 MHz timing.

NOTE:
1. Float conditions occurs when maximum output current becomes less than I_{LO} in magnitude.

Table 5.3d. Timing Requirements of the Bus Interface Unit (Continued)

Pin	Symbol	Parameter	16 MHz 1.5V		20 MHz 1.5V		25 MHz 1.5V		Comments
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
D31–D0	t8	Out Delay	1	54	1	54	0	50	C_L = 120 pF*
D31–D0	t10	Setup Time	11		11		11		
D31–D0	t11	Hold Time	11		11		11		
D31–D0	t12 (Note 1)	Float Time	6	33	6	27	5	24	C_L = 120 pF*
PEREQ	t13 (Note 1)	Float Time	1	60	1	50	1	40	C_L = 75 pF*
$\overline{\text{BUSY}}$	t13 (Note 1)	Float Time	1	60	1	50	1	40	C_L = 75 pF*
$\overline{\text{ERROR}}$	t13 (Note 1)	Float Time	1	60	1	50	1	40	C_L = 75 pF*
$\overline{\text{READYO}}$	t13 (Note 1)	Float Time	1	60	1	50	1	40	C_L = 75 pF*
$\overline{\text{ADS}}$	t14	Setup Time	26		21		16		
$\overline{\text{ADS}}$	t15	Hold Time	5		5		4		
$\overline{\text{W/R}}$	t14	Setup Time	26		21		16		
$\overline{\text{W/R}}$	t15	Hold Time	5		5		4		
$\overline{\text{READY}}$	t16	Setup Time	21		12		9		
$\overline{\text{READY}}$	t17	Hold Time	4		4		4		
$\overline{\text{CMD0}}$	t16	Setup Time	21		19		16		
$\overline{\text{CMD0}}$	t17	Hold Time	2		4		4		
$\overline{\text{NPS1}}$ NPS2	t16	Setup Time	21		19		16		
$\overline{\text{NPS1}}$ NPS2	t17	Hold Time	2		2		4		
STEN	t16	Setup Time	21		21		15		
STEN	t17	Hold Time	2		2		2		
RESETIN	t18	Setup Time	13		12		10		
RESETIN	t19	Hold Time	4		4		3		

*C_L = 50 pF for 25 MHz timing.

NOTE:
1. Float conditions occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not tested.

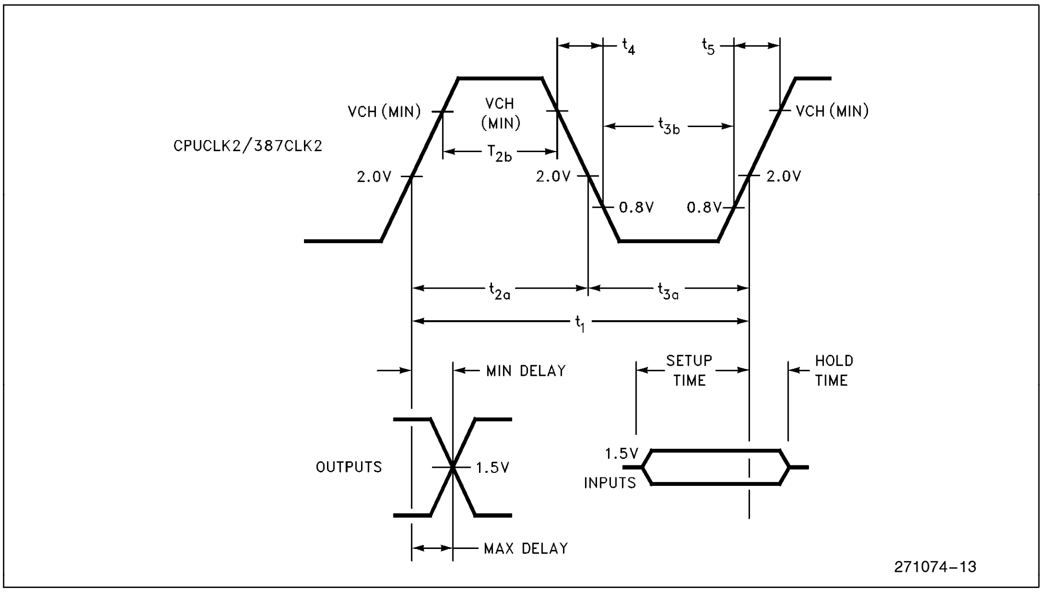


Figure 5.1. 386CLK2/387CLK2 Waveform and Measurement Points for Input/Output AC Specifications

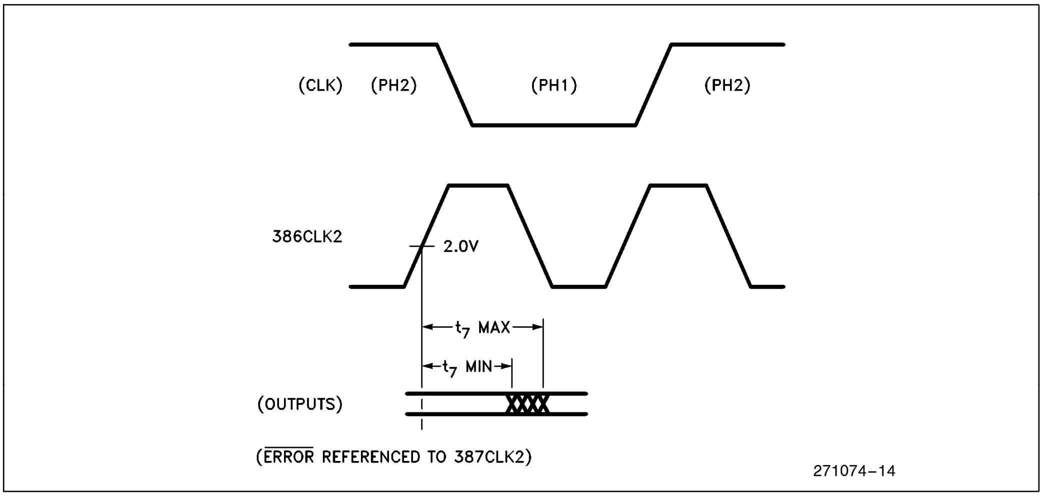
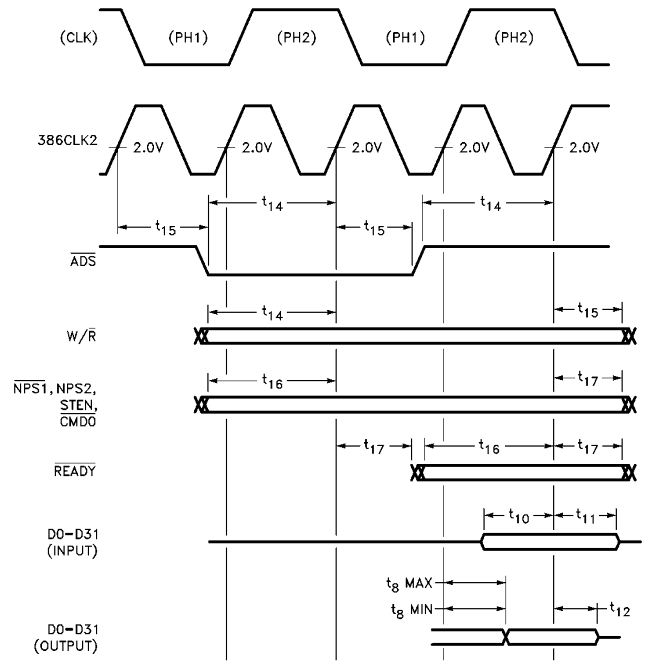
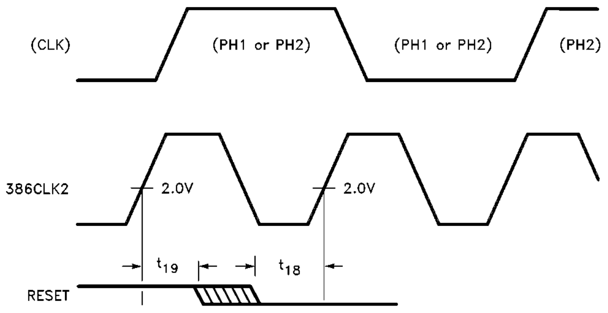


Figure 5.2. Output Signals



271074–15

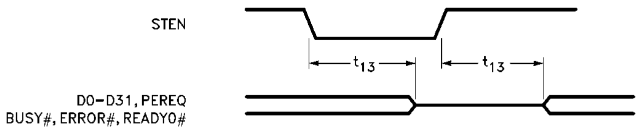
Figure 5.3. Input and I/O Signals



NOTE:
The second internal processor phase following RESET high to low transition is PH2.

271074–16

Figure 5.4. RESET Signal

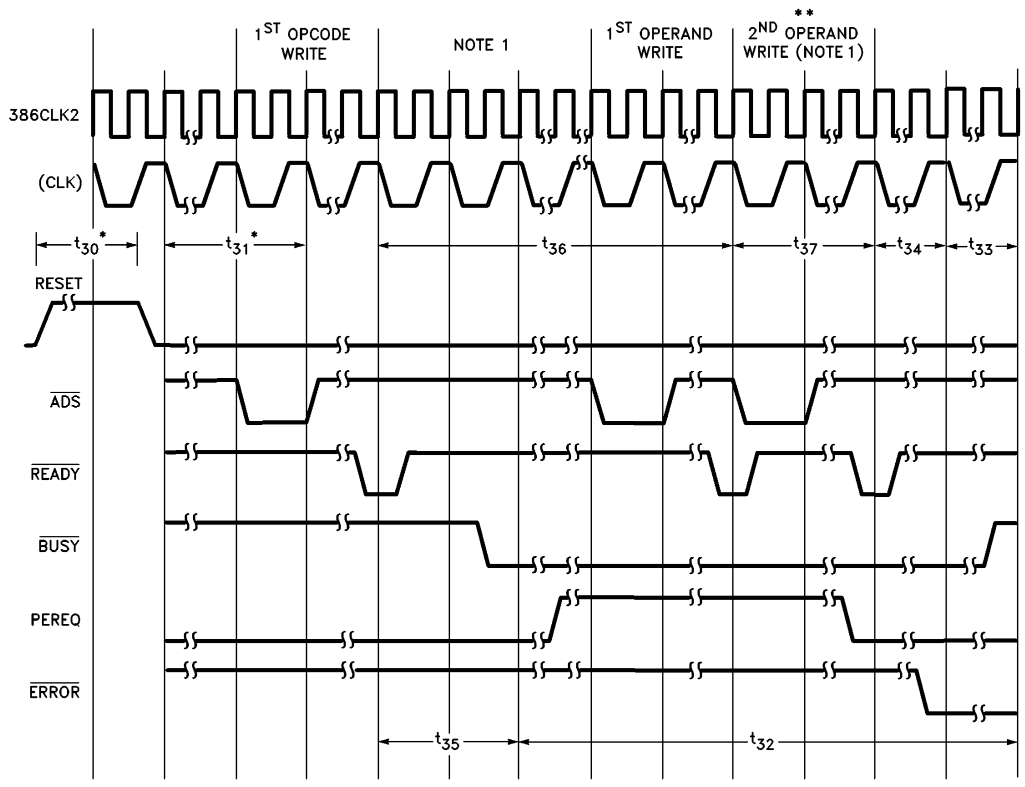


271074–17

Figure 5.5. Float from STEN

Table 5.4. Other Parameters

Pin	Symbol	Parameter	Min	Max	Units
RESETIN	t30	Duration	40		387CLK2
RESETIN	t31	RESETIN Inactive to 1st Opcode Write	50		387CLK2
$\overline{\text{BUSY}}$	t32	Duration	6		386CLK2
$\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$	t33	$\overline{\text{ERROR}}$ (In) Active to $\overline{\text{BUSY}}$ Inactive	6		386CLK2
PEREQ, $\overline{\text{ERROR}}$	t34	PEREQ Inactive to $\overline{\text{ERROR}}$ Active	6		386CLK2
$\overline{\text{READY}}$, $\overline{\text{BUSY}}$	t35	$\overline{\text{READY}}$ Active to $\overline{\text{BUSY}}$ Active	4	4	386CLK2
$\overline{\text{READY}}$	t36	Minimum Time from Opcode Write to Opcode/Operand Write	6		386CLK2
$\overline{\text{READY}}$	t37	Minimum Time from Operand Write to Operand Write	8		386CLK2



* In 387CLK2's
** or last operand

NOTE:
1. Memory read (operand) cycle is not shown.

Figure 5.6. Other Parameters

	Instruction												Optional Fields	
	First Byte				Second Byte									
1	11011	OPA		1	MOD		1	OPB		R/M			SIB	DISP
2	11011	MF		OPA	MOD		OPB			R/M			SIB	DISP
3	11011	d	P	OPA	1	1	OPB			ST(i)				
4	11011	0	0	1	1	1	1	OP						
5	11011	0	1	1	1	1	1	OP						
	15–11	10	9	8	7	6	5	4	3	2	1	0		

6.0 i387™ NUMERICS COPROCESSOR EXTENSIONS TO THE i386™ MICROPROCESSOR INSTRUCTION SET

Instructions for the i387 NPX assume one of the five forms shown in the following table. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the i386 processor addressing modes.

OP = Instruction opcode, possible split into two fields OPA and OPB

MF = Memory Format
- 00—32-bit real
- 01—32-bit integer
- 10—64-bit real
- 11—16-bit integer

P = Pop
- 0—Do not pop stack
- 1—Pop stack after operation

ESC = 11011

d = Destination
- 0—Destination is ST(0)
- 1—Destination is ST(i)

R XOR d = 0—Destination (op) Source
R XOR d = 1—Source (op) Destination

ST(i) = Register stack element *i*
- 000 = Stack top
- 001 = Second stack element
- •
- •
- •
- 111 = Eighth stack element

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of i386 processor instructions (refer to *i386 Microprocessor Programmer's Reference Manual*)

SIB (Scale Index Base) byte and DISP (displacement) are optionally present in instructions that have MOD and R/M fields. Their presence depends on the values of MOD and R/M, as for i386 processor instructions.

The instruction summaries that follow assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD request delaying processor access to the bus; and that no exceptions are detected during instruction execution. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

i387™ NPX Extensions to the i386 Processor Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load[a]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	20	45–52	25	61–65
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP	56–67			
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP	44			
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP	266–275			
ST(i) to ST(0)	ESC 001	11000 ST(i)		14			
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	44	79–93	45	82–95
ST(0) to ST(i)	ESC 101	11010 ST(i)		11			
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	44	79–93	45	82–95
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP	80–97			
ST(0) to extended real	ESC 011	MOD 111 R/M	SIB/DISP	53			
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP	512–534			
ST(0) to ST(i)	ESC 101	11001 ST (i)		12			
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)		18			
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	26	56–63	31	71–75
ST(i) to ST(0)	ESC 000	11010 ST(i)		24			
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	SIB/DISP	26	56–63	31	71–75
ST(i) to ST(0)	ESC 000	11011 ST(i)		26			
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001		26			
FTST = Test ST(0)	ESC 001	1110 0100		28			
FUCOM = Unordered compare	ESC 101	11100 ST(i)		24			
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)		26			
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001		26			
FXAM = Examine ST(0)	ESC 001	11100101		30-38			
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110		20			
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000		24			
FLDPI = Load pi into ST(0)	ESC 001	1110 1011		40			
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001		40			

Shaded areas indicate instructions not available in M8087/M80287.

NOTE:
a. When loading single- or double-precision zero from memory, add 5 clocks.

i387™ NPX Extensions to the i386™ Processor Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS (Continued)							
FLDL2E = Load $\log_2$(e) into ST(0)	ESC 001	1110 1010		40			
FLDLG2 = Load $\log_{10}$(2) into ST(0)	ESC 001	1110 1100		41			
FLDLN2 = Load $\log_e$(2) into ST(0)	ESC 001	1110 1101		41			
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	24–32	57–72	29–37	71–85
ST(i) and ST(0)	ESC d P 0	11000 ST(i)		23–31[b]			
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	24–32	57–82	28–36	71–83[c]
ST(i) and ST(0)	ESC d P 0	1110 R R/M		26–34[d]			
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	27–35	61–82	32–57	76–87
ST(i) and ST(0)	ESC d P 0	1100 1 R/M		29–57[e]			
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	89	120–127[f]	94	136–140[g]
ST(i) and ST(0)	ESC d P 0	1111 R R/M		88[h]			
FSQRT[i] = Square root	ESC 001	1111 1010		122–129			
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101		67–86			
FPREM = Partial remainder	ESC 001	1111 1000		74–155			
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101		95–185			
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100		66–80			
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100		70–76			
FABS = Absolute value of ST(0)	ESC 001	1110 0001		22			
FCHS = Change sign of ST(0)	ESC 001	1110 0000		24–25			

Shaded areas indicate instructions not available in M8087/M80287.

NOTES:
b. Add 3 clocks to the range when d = 1.
c. Add 1 clock to **each** range when R = 1.
d. Add 3 clocks to the range when d = 0.
e. typical = 52 (When d = 0, 46–54, typical = 49).
f. Add 1 clock to the range when R = 1.
g. 135–141 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq$ ST(0) $\leq +\infty$.

i387™ NPX Extensions to the i386™ Processor Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2–6	
TRANSCENDENTAL				
FCOS[k] = Cosine of ST(0)	ESC 001	1111 1111		123–772[j]
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		191–497[j]
FPATAN = Partial arctangent	ESC 001	1111 0011		314–487
FSIN[k] = Sine of ST(0)	ESC 001	1111 1110		122–771[j]
FSINCOS[k] = Sine and cosine of ST(0)	ESC 001	1111 1011		194–809[j]
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		211–476
FYL2X[m] = $ST(1) * \log_2(ST(0))$	ESC 001	1111 0001		120–538
FYL2XP1[n] = $ST(1) * \log_2(ST(0) + 1.0)$	ESC 001	1111 1001		257–547
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		33
FSTSW AX = Store status word	ESC 111	1110 0000		13
FLDCW = Load control word	ESC 001	MOD 101 R/M	SIB/DISP	19
FSTCW = Store control word	ESC 101	MOD 111 R/M	SIB/DISP	15
FSTSW = Store status word	ESC 101	MOD 111 R/M	SIB/DISP	15
FCLEX = Clear exceptions	ESC 011	1110 0010		11
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP	103–104
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP	71
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP	375–376
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP	308
FINCSTP = Increment stack pointer	ESC 001	1111 0111		21
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		22
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		18
FNOP = No operations	ESC 001	1101 0000		12

Shaded areas indicate instructions not available in M8087/M80287.

NOTES:

j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 76 additional clocks may be needed to reduce the operand.

k. $0 \leq |ST(0)| < 2^{63}$.

l. $-1.0 \leq ST(0) \leq 1.0$.

m. $0 \leq ST(0) < \infty, -\infty < ST(1) < +\infty$.

n. $0 \leq |ST(0)| < (2 - SQRT(2))/2, -\infty < ST(1) < +\infty$.

APPENDIX A COMPATIBILITY BETWEEN THE M80287 AND THE M8087

The M80286/M80287 operating in Real-Address mode will execute M8086/M8087 programs without major modification. However, because of differences in the handling of numeric exceptions by the M80287 NPX and the M8087 NPX, exception-handling routines *may* need to be changed.

This appendix summarizes the differences between the M80287 NPX and the M8087 NPX, and provides details showing how M8086/M8087 programs can be ported to the M80286/M80287.

1. The NPX signals exceptions through a dedicated $\overline{\text{ERROR}}$ line to the M80286. The NPX error signal does not pass through an interrupt controller (the M8087 INT signal does). Therefore, any interrupt-controller-oriented instructions in numeric exception handlers for the M8086/M8087 should be deleted.
2. The M8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the M80287. If the M80287 encounters one of these opcodes in its instruction stream, the instruction will effectively be ignored—none of the M80287 internal states will be updated. While M8086/M8087 containing these instructions may be executed on the M80286/M80287, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the M80287.
3. Interrupt vector 16 must point to the numeric exception handling routine.
4. The ESC instruction address saved in the M80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the M8087 does not include leading prefixes.
5. In Protected-Address mode, the format of the M80287's saved instruction and address pointers is different than for the M8087. The instruction opcode is not saved in Protected mode—exception handlers will have to retrieve the opcode from memory if needed.
6. Interrupt 7 will occur in the M80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the M80286 MSW set (TS = 1 or EM = 1). If TS is set, then a WAIT instruction will also cause interrupt 7. An exception handler should be included in M80286/M80287 code to handle these situations.
7. Interrupt 9 will occur if the second or subsequent words of a floating-point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An exception handler should be included in M80286/M80287 code to report these programming errors.
8. Except for the processor control instructions, all of the M80287 numeric instructions are automatically synchronized by the M80286 CPU—the M80286 automatically tests the $\overline{\text{BUSY}}$ line from the M80287 to ensure that the M80287 has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the M8087 used with M8086 and M8088 processors, explicit WAITs are required before each numeric instruction to ensure synchronization. Although M8086/M8087 programs having explicit WAIT instructions will execute perfectly on the M80286/M80287 without reassembly, these WAIT instructions are unnecessary.
9. Since the M80287 does not require WAIT instructions before each numeric instruction, the ASM286 assembler does not automatically generate these WAIT instructions. The ASM86 assembler, however, automatically precedes every ESC instruction with a WAIT instruction. Although numeric routines generated using the ASM86 assembler will generally execute correctly on the M80286/M80287, reassembly using ASM286 may result in a more compact code image.

   The processor control instructions for the M80287 may be coded using either a WAIT or No-WAIT form of mnemonic. The WAIT forms of these instructions cause ASM286 to precede the ESC instruction with a CPU WAIT instruction, in the identical manner as does ASM86.