80C187
80-BIT MATH COPROCESSOR

- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Upward Object-Code Compatible from 8087**
- **Fully Compatible with 387DX and 387SX Math Coprocessors. Implements all 387 Architectural Enhancements over 8087**
- **Directly Interfaces with 80C186 CPU**
- **80C186/80C187 Provide a Software/ Binary Compatible Upgrade from 80186/82188/8087 Systems**
- **Expands 80C186's Data Types to Include 32-, 64-, 80-Bit Floating-Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends 80C186's Instruction Set to 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**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in 40-Pin CERDIP and 44-Pin PLCC Package**

  (See Packaging Outlines and Dimensions, Order #231369)

The Intel 80C187 is a high-performance math coprocessor that extends the architecture of the 80C186 with floating-point, extended integer, and BCD data types. A computing system that includes the 80C187 fully conforms to the IEEE Floating-Point Standard. The 80C187 adds over seventy mnemonics to the instruction set of the 80C186, including support for arithmetic, logarithmic, exponential, and trigonometric mathematical operations. The 80C187 is implemented with 1.5 micron, high-speed CHMOS III technology and packaged in both a 40-pin CERDIP and a 44-pin PLCC package. The 80C187 is upward object-code compatible from the 8087 math coprocessor and will execute code written for the 80387DX and 80387SX math coprocessors.

November 1992 **Order Number: 270640-004**

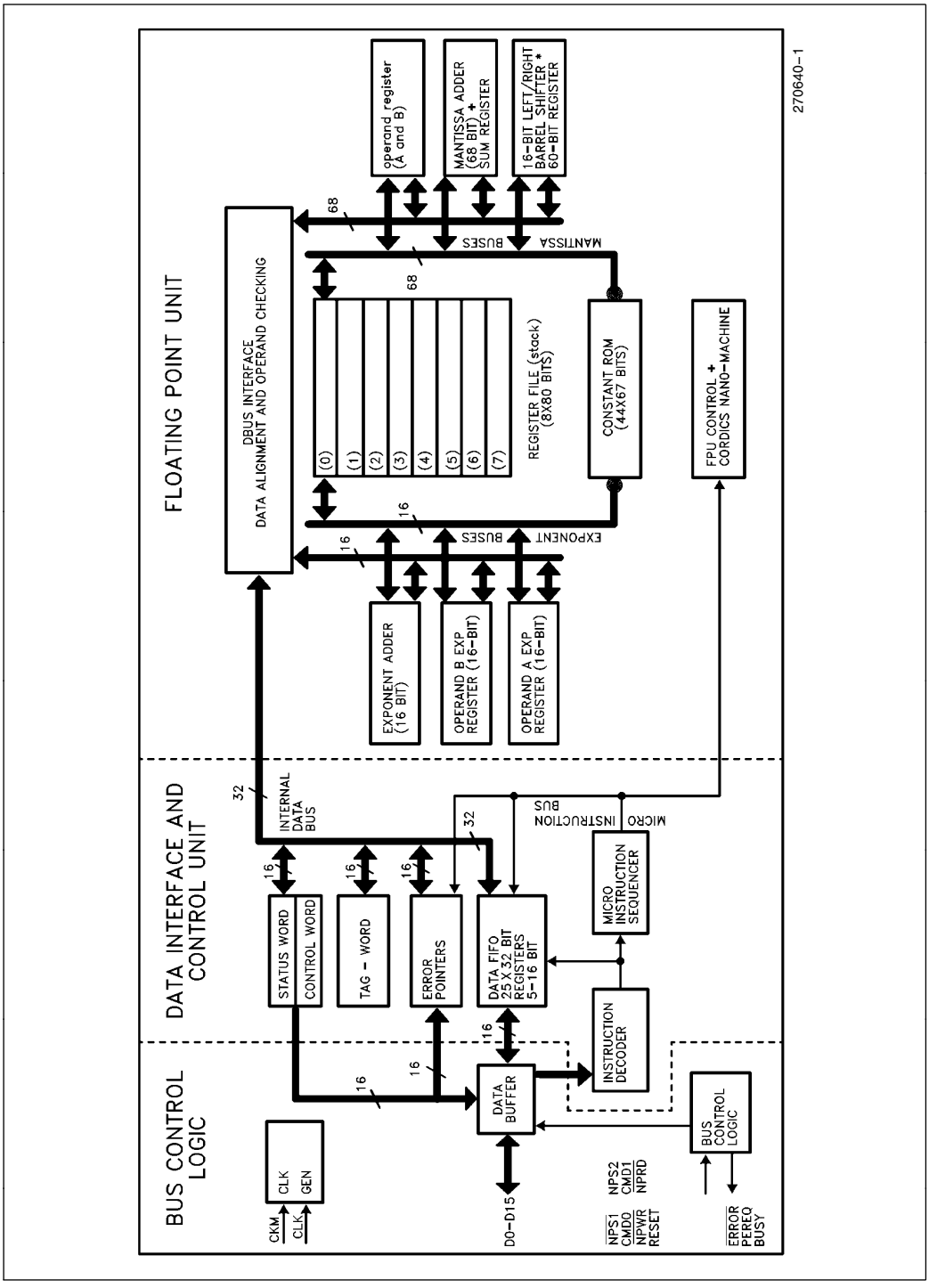


Figure 1. 80C187 Block Diagram

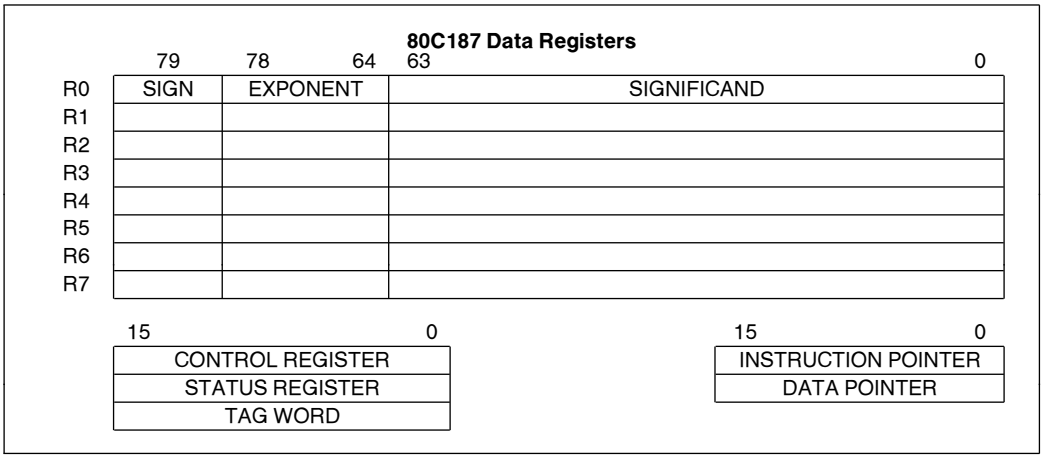


Figure 2. Register Set

FUNCTIONAL DESCRIPTION

The 80C187 Math Coprocessor provides arithmetic instructions for a variety of numeric data types. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The 80C187 effectively extends the register and instruction set of the 80C186 CPU for existing data types and adds several new data types as well. Figure 2 shows the additional registers visible to programs in a system that includes the 80C187. Essentially, the 80C187 can be treated as an additional resource or an extension to the CPU. The 80C186 CPU together with an 80C187 can be used as a single unified system.

A 80C186 system that includes the 80C187 is completely upward compatible with software for the 8086/8087.

The 80C187 interfaces only with the 80C186 CPU. The interface hardware for the 80C187 is not implemented on the 80C188.

PROGRAMMING INTERFACE

The 80C187 adds to the CPU additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. All new instructions and data types are directly supported by the assembler and compilers for high-level languages. The 80C187 also supports the full 80387DX instruction set.

All communication between the CPU and the 80C187 is transparent to applications software. The CPU automatically controls the 80C187 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 80C187. All memory addressing modes are available for addressing numerics operands.

The end of this data sheet lists by class the instructions that the 80C187 adds to the instruction set.

NOTE:
The 80C187 Math Coprocessor is also referred to as a Numeric Processor Extension (NPX) in this document.

Data Types

Table 1 lists the seven data types that the 80C187 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 even physical-memory addresses; operands may begin at odd addresses, but will require extra memory cycles to access the entire operand.

Internally, the 80C187 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.

Numeric Operands

A typical NPX instruction accepts one or two operands and produces one (or sometimes two) results. 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 (refer to the section on Data Registers).

Register Set

Figure 2 shows the 80C187 register set. When an 80C187 is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

DATA REGISTERS

80C187 computations use the extended-precision real data type.

Table 1. Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte ... HIGHEST ADDRESSED BYTE
Word Integer	$\pm 10^4$	16 Bits	(TWO'S COMPLEMENT) 15 0
Short Integer	$\pm 10^9$	32 Bits	(TWO'S COMPLEMENT) 31 0
Long Integer	$\pm 10^{18}$	64 Bits	(TWO'S COMPLEMENT) 63 0
Packed BCD	$\pm 10^{18}$	18 Digits	S X MAGNITUDE d_{17} ... d_0 79 72 0
Single Precision	$\pm 10^{\pm 38}$	24 Bits	S BIASED EXPONENT SIGNIFICAND 31 23 ▲ 0
Double Precision	$\pm 10^{\pm 308}$	53 Bits	S BIASED EXPONENT SIGNIFICAND 63 52 ▲ 0
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits	S BIASED EXPONENT I SIGNIFICAND 79 64 63 ▲ 0

270640–2

NOTES:
1. S = Sign bit (0 = Positive, 1 = Negative)
2. d_n = Decimal digit (two per byte)
3. X = Bits have no significance; 80C187 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: $(-1)^S$ $(D_{17} \ldots D_0)$
8. Real: $(-1)^S$ $(2^{E\text{-}BIAS})$ $(F_0, F_1 \ldots)$

The 80C187 register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as individually addressable 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. The 80C187 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 use. This explicit addressing is also relative to TOP.

TAG WORD

The tag word marks the content of each numeric data register, as Figure 3 shows. Each two-bit tag represents one of the eight data registers. The principal function of the tag word is to optimize the NPX's performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to identify special values (e.g. NaNs or denormals) in the contents of a stack location without the need to perform complex decoding of the actual data.

STATUS WORD

The 16-bit status word (in the status register) shown in Figure 4 reflects the overall state of the 80C187. It may be read and inspected by programs.

Bit 15, the B-bit (busy bit) is included for 8087 compatibility only. It always has the same value as the ES bit (bit 7 of the status word); it does **not** indicate the status of the BUSY output of 80C187.

Bits 13–11 (TOP) point to the 80C187 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 through 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 4 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the 80C187 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 80C187 is activated immediately.

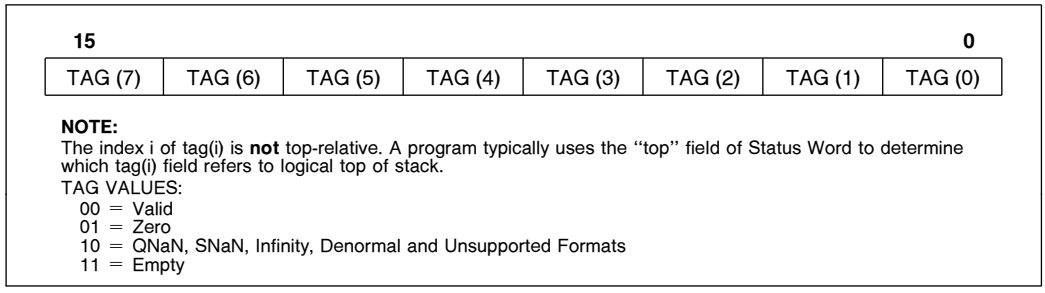


Figure 3. Tag Word

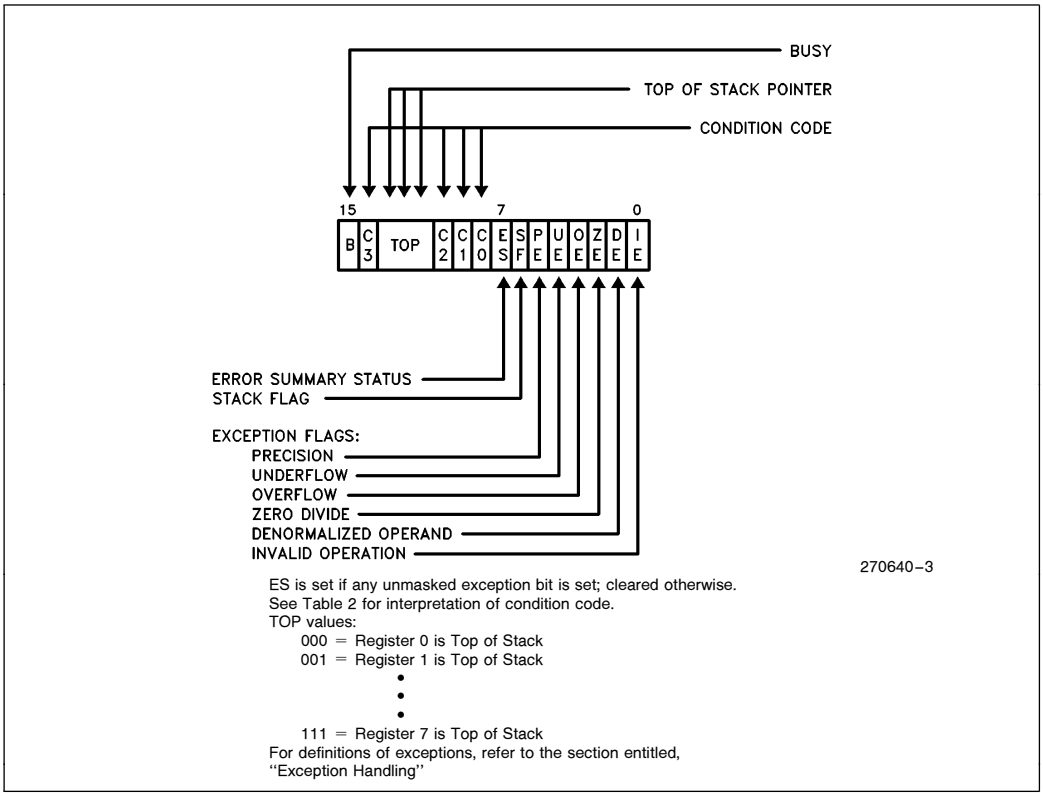


Figure 4. Status Word

CONTROL WORD

The NPX provides several processing options that are selected by loading a control word from memory into the control register. Figure 5 shows the format and encoding of fields in the control word.

Table 2. Condition Code Interpretation

Instruction	C0(S)	C3(Z)	C1(A)	C2(C)
FPREM, FPREM1 (See Table 3)	Three Least Significant Bits of Quotient Q2 Q0		Q1 or O/$\overline{U}$	Reduction 0 = Complete 1 = Incomplete
FCOM, FCOMP, FCOMPP, FTST FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of Comparison (See Table 4)		Zero or O/$\overline{U}$	Operand is not Comparable (Table 4)
FXAM	Operand Class (See Table 5)		Sign or O/$\overline{U}$	Operand Class (Table 5)
FCHS, FABS, FXCH, FINCSTP, FDECSTP, Constant Loads, FXTRACT, FLD, FILD, FBLD, FSTP (Ext Real)	UNDEFINED		Zero or O/$\overline{U}$	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/$\overline{U}$	UNDEFINED
FPTAN, FSIN, FCOS, FSINCOS	UNDEFINED		Roundup or O/$\overline{U}$, Undefined if C2 = 1	Reduction 0 = Complete 1 = Incomplete
FLDENV, FRSTOR	Each Bit Loaded from Memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

O/$\overline{U}$ 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 one was added to the least significant bit of the result during the last rounding.

UNDEFINED Do not rely on finding any specific value in these bits.

The low-order byte of this control word configures exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 80C187 recognizes.

The high-order byte of the control word configures the 80C187 operating mode, including precision, rounding, and infinity control.

- The ''infinity control bit'' (bit 12) is not meaningful to the 80C187, and programs must ignore its value. To maintain compatibility with the 8087, this bit can be programmed; however, regardless of its value, the 80C187 always treats infinity in the affine sense ($-\infty < +\infty$). This bit is initialized to zero both after a hardware reset and after the FINIT instruction.
- 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 80C187 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.

Table 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 Iteration Required for Complete Reduction	
	Q1	**Q0**	**Q2**	**Q MOD 8**	
0	0	0	0	0	Complete Reduction: C0, C3, C1 Contain Three Least Significant Bits of Quotient
	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 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 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	1	− Denormal

INSTRUCTION AND DATA POINTERS

Because the NPX operates in parallel with the CPU, any exceptions detected by the NPX may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numerics instruction, the 80C187 contains registers that aid in diagnosis. These registers supply the opcode of the failing numerics instruction, the address of the instruction, and the address of its numerics memory operand (if appropriate).

The instruction and data pointers are provided for user-written exception handlers. Whenever the 80C187 executes 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 the format shown by Figure 6. The ESC instruction FLDENV, FSTENV, FSAVE and FRSTOR are used to transfer these values between the registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

Interrupt Description

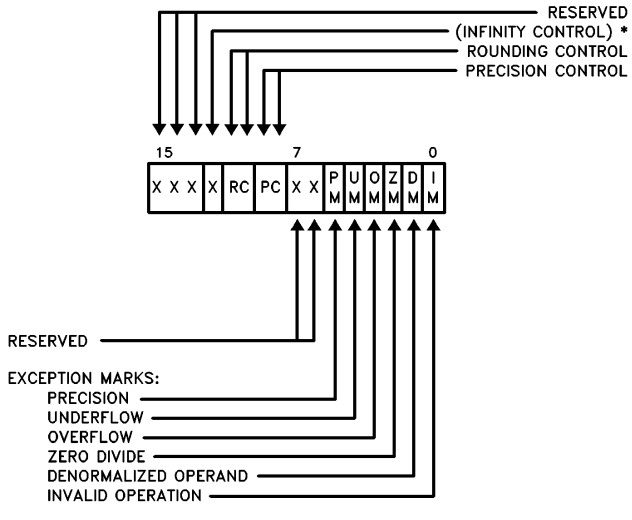
CPU interrupt 16 is used to report exceptional conditions while executing numeric programs. Interrupt 16 indicates that the previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC instructions can cause this interrupt. The CPU return address pushed onto the stack of the exception handler points to an 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.

Exception Handling

The 80C187 detects six different exception conditions that can occur during instruction execution. Table 6 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the 80C187 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, interrupt 16 occurs. The exception condition must be resolved via an interrupt service routine. The return address pushed onto the CPU stack upon entry to the service routine does not necessarily point to the failing instruction nor to the following instruction. The 80C187 saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

If error trapping is required at the end of a series of numerics instructions (specifically, when the last ESC instruction modifies memory data and that data is used in subsequent nonnumerics instructions), it is necessary to insert the FNOP instruction to force the 80C187 to check its $\overline{\text{ERROR}}$ input.



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)

*The ''infinity control'' bit is not meaningful to the 80C187. To maintain compatibility with the 8087, this bit can be programmed; however, regardless of its value, the 80C187 treats infinity in the affine sense ($-\infty < +\infty$).

Figure 5. Control Word

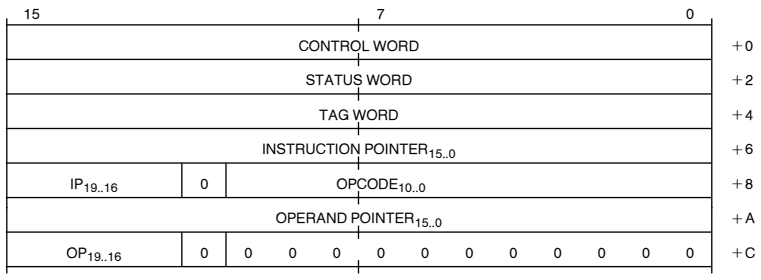


Figure 6. Instruction and Data Pointer Image in Memory

Table 6. Exceptions

Exception	Cause	Default Action (If Exception is Masked)
Invalid Operation	Operation on a signalling NaN, unsupported format, indeterminate form (0*∞, 0/0), (+∞) + (−∞), 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	The operand is normalized, and 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

Initialization

After FNINIT or RESET, the control word contains the value 037FH (all exceptions masked, precision control 64 bits, rounding to nearest) the same values as in an 8087 after RESET. For compatibility with the 8087, the bit that used to indicate infinity control (bit 12) is set to zero; however, regardless of its setting, infinity is treated in the affine sense. After FNINIT or RESET, the status word is initialized as follows:

- All exceptions are set to zero.
- Stack TOP is zero, so that after the first push the stack top will be register seven (111B).
- The condition code C_3–C_0 is **undefined**.
- The B-bit is zero.

The tag word contains FFFFH (all stack locations are empty).

80C186/80C187 initialization software should execute an FNINIT instruction (i.e. an FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for 80C187 software, but Intel recommends its use to help ensure upward compatibility with other processors.

8087 Compatibility

This section summarizes the differences between the 80C187 and the 8087. Many changes have been designed into the 80C187 to directly support the IEEE standard in hardware. These changes result in increased performance by elminating the need for software that supports the standard.

GENERAL DIFFERENCES

The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80C187 Numeric Processor Extension. They do not alter the state of the 80C187 Numeric Processor Extension. (They are treated similarly to FNOP, except that $\overline{\text{ERROR}}$ is not checked.) While 8086/8087 code containing these instructions can be executed on the 80C186/80C187, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the 80C187 Numeric Processor Extension.

The 80C187 differs from the 8087 with respect to instruction, data, and exception synchronization. Except for the processor control instructions, all of the 80C187 numeric instructions are automatically synchronized by the 80C186 CPU. When necessary, the

80C186 automatically tests the BUSY line from the 80C187 Numeric Processor Extension to ensure that the 80C187 Numeric Processor Extension has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used with 8086 and 8088 CPUs, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute on the 80C186/80C187, these WAIT instructions are unnecessary.

The 80C187 supports only affine closure for infinity arithmetic, not projective closure.

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

Rounding control is in effect for FLD *constant*.

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

After reset, FINIT, and incomplete FPREM, the 80C187 resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the 80C187 does not support the special data formats pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

The denormal exception has a different purpose on the 80C187. A system that uses the denormal-exception handler solely to normalize the denormal operands, would better mask the denormal exception on the 80C187. The 80C187 automatically normalizes denormal operands when the denormal exception is masked.

EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the 80C186/80C187:

1. The 80C186/80C187 traps exceptions only on the next ESC instruction; i.e. the 80C186 does not notice unmasked 80C187 exceptions on the 80C186 $\overline{\text{ERROR}}$ input line until a later numerics instruction is executed. Because the 80C186 does not sample $\overline{\text{ERROR}}$ on WAIT and FWAIT instructions, programmers should place an FNOP instruction at the end of a sequence of numerics instructions to force the 80C186 to sample its $\overline{\text{ERROR}}$ input.
2. The 80C187 Numeric Processor Extension signals exceptions through a dedicated $\overline{\text{ERROR}}$ line to the CPU. The 80C187 error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt-controller-oriented instructions in numerics exception handlers for the 8086/8087 should be deleted.
3. Interrupt vector 16 must point to the numerics exception handling routine.
4. The ESC instruction address saved in the 80C187 Numeric Processor Extension includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. When the overflow or underflow exception is masked, the 80C187 differs from the 8087 in rounding when overflow or underflow occurs. The 80C187 produces results that are consistent with the rounding mode.
6. When the underflow exception is masked, the 80C187 sets its underflow flag only if there is also a loss of accuracy during denormalization.
7. 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.
8. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
9. The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
10. The denormal exception no longer takes precedence over all other exceptions.
11. When the denormal exception is masked, the 80C187 automatically normalizes denormal operands. The 8087 performs unnormal arithmetic, which might produce an unnormal result.
12. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
13. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
14. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
15. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized oper-

and exception. When loading a signalling NaN, FLD *single/double precision* signals an invalid-operand exception.

16. The 80C187 only generates quiet NaNs (as on the 8087); however, the 80C187 distinguishes between quiet NaNs and signalling NaNs. Signalling 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).
17. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 8087 leaves the original operand in ST(1) intact.
18. 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, ∞) 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 8087 returns zero in the first case and raises the invalid-operation exception in the other cases.
19. The 80C187 returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 supports a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

Table 7. Pin Summary

Pin Name	Function	Active State	Input/ Output
CLK	CLocK		I
CKM	ClocKing Mode		I
RESET	System reset	High	I
PEREQ	Processor Extension REQuest	High	O
BUSY	Busy status	High	O
$\overline{\text{ERROR}}$	Error status	Low	O
$D_{15}-D_0$	Data pins	High	I/O
$\overline{\text{NPRD}}$	Numeric Processor ReaD	Low	I
$\overline{\text{NPWR}}$	Numeric Processor WRite	Low	I
$\overline{\text{NPS1}}$	NPX select #1	Low	I
NPS2	NPX select #2	High	I
CMD0	CoMmanD 0	High	I
CMD1	CoMmanD 1	High	I
V_{CC}	System power		I
V_{SS}	System ground		I

HARDWARE INTERFACE

In the following description of hardware interface, an overbar above a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no overbar is present above the signal name, the signal is asserted when at the high voltage level.

Signal Description

In the following signal descriptions, the 80C187 pins are grouped by function as follows:

1. Execution Control— CLK, CKM, RESET
2. NPX Handshake— PEREQ, BUSY, $\overline{\text{ERROR}}$
3. Bus Interface Pins— D_{15}–D_0, $\overline{\text{NPWR}}$, $\overline{\text{NPRD}}$
4. Chip/Port Select— $\overline{\text{NPS1}}$, NPS2, CMD0, CMD1
5. Power Supplies— V_{CC}, V_{SS}

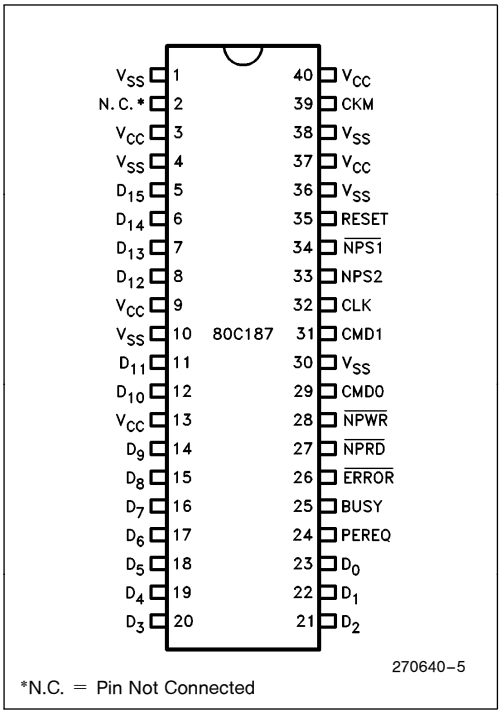
Table 7 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. Figure 7 shows the locations of pins on the CERDIP package, while Figure 8 shows the locations of pins on the PLCC package. Table 8 helps to locate pin identifiers in Figures 7 and 8.

Clock (CLK)

This input provides the basic timing for internal operation. This pin does not require MOS-level input; it will operate at either TTL or MOS levels up to the maximum allowed frequency. A minimum frequency must be provided to keep the internal logic properly functioning. Depending on the signal on CKM, the signal on CLK can be divided by two to produce the internal clock signal (in which case CLK may be up to 32 MHz in frequency), or can be used directly (in which case CLK may be up to 12.5 MHz).

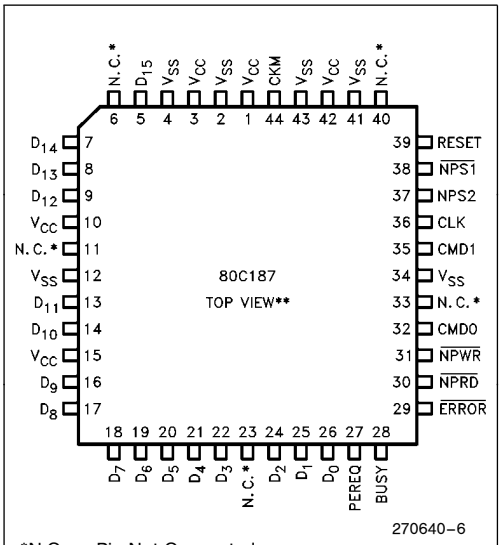
Clocking Mode (CKM)

This pin is a strapping option. When it is strapped to V_{CC} (HIGH), the CLK input is used directly; when strapped to V_{SS} (LOW), the CLK input is divided by two to produce the internal clock signal. During the RESET sequence, this input must be stable at least four internal clock cycles (i.e. CLK clocks when CKM is HIGH; 2 × CLK clocks when CKM is LOW) before RESET goes LOW.



*N.C. = Pin Not Connected

Figure 7. CERDIP Pin Configuration



*N.C. = Pin Not Connected
**"Top View" means as the package is seen from the component side of the board.

Figure 8. PLCC Pin Configuration

Table 8. PLCC Pin Cross-Reference

Pin Name	CERDIP Package	PLCC Package
BUSY	25	28
CKM	39	44
CLK	32	36
CMD0	29	32
CMD1	31	35
D_0	23	26
D_1	22	25
D_2	21	24
D_3	20	22
D_4	19	21
D_5	18	20
D_6	17	19
D_7	16	18
D_8	15	17
D_9	14	16
D_{10}	12	14
D_{11}	11	13
D_{12}	8	9
D_{13}	7	8
D_{14}	6	7
D_{15}	5	5
$\overline{\text{ERROR}}$	26	29
No Connect	2	6, 11, 23, 33, 40
$\overline{\text{NPRD}}$	27	30
$\overline{\text{NPS1}}$	34	38
NPS2	33	37
$\overline{\text{NPWR}}$	28	31
PEREQ	24	27
RESET	35	39
V_{CC}	3, 9, 13, 37, 40	1, 3, 10, 15, 42
V_{SS}	1, 4, 10, 30, 36, 38	2, 4, 12, 34, 41, 43

System Reset (RESET)

A LOW to HIGH transition on this pin causes the 80C187 to terminate its present activity and to enter a dormant state. RESET must remain active (HIGH) for at least four internal clock periods. (The relation of the internal clock period to CLK depends on CLKM; the internal clock may be different from that of the CPU.) Note that the 80C187 is active internally for 25 clock periods after the termination of the RESET signal (the HIGH to LOW transition of RESET); therefore, the first instruction should not be written to the 80C187 until 25 internal clocks after the falling edge of RESET. Table 9 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive states.

Table 9. Output Pin Status during Reset

Output Pin Name	Value during Reset
BUSY	HIGH
$\overline{\text{ERROR}}$	HIGH
PEREQ	LOW
D_{15}–D_0	TRI-STATE OFF

Processor Extension Request (PEREQ)

When active, this pin signals to the CPU that the 80C187 is ready for data transfer to/from its data FIFO. When there are more than five data transfers,

PEREQ is deactivated after the first three transfers and subsequently after every four transfers. This signal always goes inactive before BUSY goes inactive.

Busy Status (BUSY)

When active, this pin signals to the CPU that the 80C187 is currently executing an instruction. This pin is active HIGH. It should be connected to the 80C186's $\overline{TEST}$/BUSY pin. During the RESET sequence this pin is HIGH. The 80C186 uses this HIGH state to detect the presence of an 80C187.

Error Status ($\overline{ERROR}$)

This pin reflects the ES bit of the status register. When active, it indicates that an unmasked exception has occurred. This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, FNSAVE, FLDCW, FLDENV, and FRSTOR. This pin should be connected to the $\overline{ERROR}$ pin of the CPU. $\overline{ERROR}$ can change state only when BUSY is active.

Data Pins (D_{15}–D_0)

These bidirectional pins are used to transfer data and opcodes between the CPU and 80C187. They are normally connected directly to the corresponding CPU data pins. Other buffers/drivers driving the local data bus must be disabled when the CPU reads from the NPX. High state indicates a value of one. D_0 is the least significant data bit.

Numeric Processor Write ($\overline{NPWR}$)

A signal on this pin enables transfers of data from the CPU to the NPX. This input is valid only when $\overline{NPS1}$ and NPS2 are both active.

Numeric Processor Read ($\overline{NPRD}$)

A signal on this pin enables transfers of data from the NPX to the CPU. This input is valid only when $\overline{NPS1}$ and NPS2 are both active.

Numeric Processor Selects ($\overline{NPS1}$ and NPS2)

Concurrent assertion of these signals indicates that the CPU is performing an escape instruction and enables the 80C187 to execute that instruction. No data transfer involving the 80C187 occurs unless the device is selected by these lines.

Command Selects (CMD0 and CMD1)

These pins along with the select pins allow the CPU to direct the operation of the 80C187.

System Power (V_{CC})

System power provides the $+5V \pm 10\%$ DC supply input. All V_{CC} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

System Ground (V_{SS})

All V_{SS} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

Processor Architecture

As shown by the block diagram (Figure 1), the 80C187 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 CPU bus tracking and interface.

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 80C187 and transferring outputs from the 80C187 to memory. A dedicated communication protocol makes possible high-speed transfer of opcodes and operands between the CPU and 80C187.

Table 10. Bus Cycles Definition

$\overline{NPS1}$	NPS2	CMD0	CMD1	$\overline{NPRD}$	$\overline{NPWR}$	Bus Cycle Type
x	0	x	x	x	x	80C187 Not Selected
1	x	x	x	x	x	80C187 Not Selected
0	1	0	0	1	0	Opcode Write to 80C187
0	1	0	0	0	1	CW or SW Read from 80C187
0	1	1	0	0	1	Read Data from 80C187
0	1	1	0	1	0	Write Data to 80C187
0	1	0	1	1	0	Write Exception Pointers
0	1	0	1	0	1	Reserved
0	1	1	1	0	1	Read Opcode Status
0	1	1	1	1	0	Reserved

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, FSTCW, FSETPM, or FRSTPM, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the BUSY, PEREQ, and $\overline{ERROR}$ signals that synchronize 80C187 activities with the CPU.

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.

Bus Cycles

The pins $\overline{NPS1}$, NPS2, CMD0, CMD1, $\overline{NPRD}$ and $\overline{NPWR}$ identify bus cycles for the NPX. Table 10 defines the types of 80C187 bus cycles.

80C187 ADDRESSING

The $\overline{NPS1}$, NPS2, CMD0, and CMD1 signals allow the NPX to identify which bus cycles are intended for the NPX. The NPX responds to I/O cycles when the I/O address is 00F8H, 00FAH, 00FCH, or 00FEH. The correspondence betwen I/O addresses and control signals is defined by Table 11. To guarantee correct operation of the NPX, programs must not perform any I/O operations to these reserved port addresses.

Table 11. I/O Address Decoding

I/O Address (Hexadecimal)	80C187 Select and Command Inputs			
	NPS2	$\overline{NPS1}$	CMD1	CMD0
00F8	1	0	0	0
00FA	1	0	0	1
00FC	1	0	1	0
00FE	1	0	1	1

CPU/NPX SYNCHRONIZATION

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

BUSY is used to synchronize instruction transfer from the CPU to the 80C187. When the 80C187 recognizes an ESC instruction, it asserts BUSY. For most ESC instructions, the CPU waits for the 80C187 to deassert BUSY before sending the new opcode.

The NPX uses the PEREQ pin of the 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 CPU provides memory access services for the NPX.

Once the CPU initiates an 80C187 instruction that has operands, the CPU waits for PEREQ signals that indicate when the 80C187 is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the CPU continues program execution while the 80C187 executes the ESC instruction.

In 8086/8087 systems, WAIT instructions are required to achieve synchronization of both commands and operands. The 80C187, however, does not require WAIT instructions. The WAIT or FWAIT instruction commonly inserted by high-level compilers and assembly-language programmers for exception synchronization is not treated as an instruction by the 80C186 and does not provide exception trapping. (Refer to the section ''System Configuration for 8087-Compatible Exception Trapping''.)

Once it has started to execute a numerics instruction and has transferred the operands from the CPU, the 80C187 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 a CPU interrupt.

OPCODE INTERPRETATION

The CPU and the NPX use a bus protocol that adapts to the numerics opcode being executed. Only the NPX directly interprets the opcode. Some of the results of this interpretation are relevant to the CPU. The NPX records these results (opcode status information) in an internal 16-bit register. The 80C186 accesses this register only via reads from NPX port 00FEH. Tables 10 and 11 define the signal combinations that correspond to each of the following steps.

1. The CPU writes the opcode to NPX port 00F8H. This write can occur even when the NPX is busy or is signalling an exception. The NPX does not necessarily begin executing the opcode immediately.
2. The CPU reads the opcode status information from NPX port 00FEH.
3. The CPU initiates subsequent bus cycles according to the opcode status information. The opcode status information specifies whether to wait until the NPX is not busy, when to transfer exception pointers to port 00FCH, when to read or write operands and results at port 00FAH, etc.

For most instructions, the NPX does not start executing the previously transferred opcode until the CPU (guided by the opcode status information) first writes exception pointer information to port 00FCH of the NPX. This protocol is completely transparent to programmers.

Bus Operation

With respect to bus interface, the 80C187 is fully asynchronous with the CPU, even when it operates from the same clock source as the CPU. The CPU initiates a bus cycle for the NPX by activating both $\overline{\text{NPS1}}$ and NPS2, the NPX select signals. During the CLK period in which $\overline{\text{NPS1}}$ and NPS2 are activated, the 80C187 also examines the $\overline{\text{NPRD}}$ and $\overline{\text{NPRW}}$

input signals to determine whether the cycle is a read or a write cycle and examines the CMD0 and CMD1 inputs to determine whether an opcode, operand, or control/status register transfer is to occur. The 80C187 activates its BUSY output some time after the leading edge of the NPRD or NPRW signal. Input and ouput data are referenced to the trailing edges of the NPRD and NPRW signals.

The 80C187 activates the PEREQ signal when it is ready for data transfer. The 80C187 deactivates PEREQ automatically.

System Configuration

The 80C187 can be connected to the 80C186 CPU as shown by Figure 9. (Refer to the 80C186 Data Sheet for an explanation of the 80C186's signals.) This interface has the following characteristics:

- The 80C187's NPS1, ERROR, PEREQ, and BUSY pins are connected directly to the corresponding pins of the 80C186.
- The 80C186 pin MCS3/NPS is connected to NPS1; NPS2 is connected to V_{CC}. Note that if the 80C186 CPU's DEN signal is used to gate external data buffers, it must be combined with the NPS signal to insure numeric accesses will not activate these buffers.
- The NPRD and NPRW pins are connected to the RD and WR pins of the 80C186.
- CMD1 and CMD0 come from the latched A_2 and A_1 of the 80C186, respectively.
- The 80C187 BUSY output connects to the 80C186 TEST/BUSY input. During RESET, the signal at the 80C187 BUSY output automatically programs the 80C186 to use the 80C187.
- The 80C187 can use the CLKOUT signal of the 80C186 to conserve board space when operating at 12.5 MHz or less. In this case, the 80C187 CKM input must be pulled HIGH. For operation in excess of 12.5 MHz, a double-frequency external oscillator for CLK input is needed. In this case, CKM must be pulled LOW.

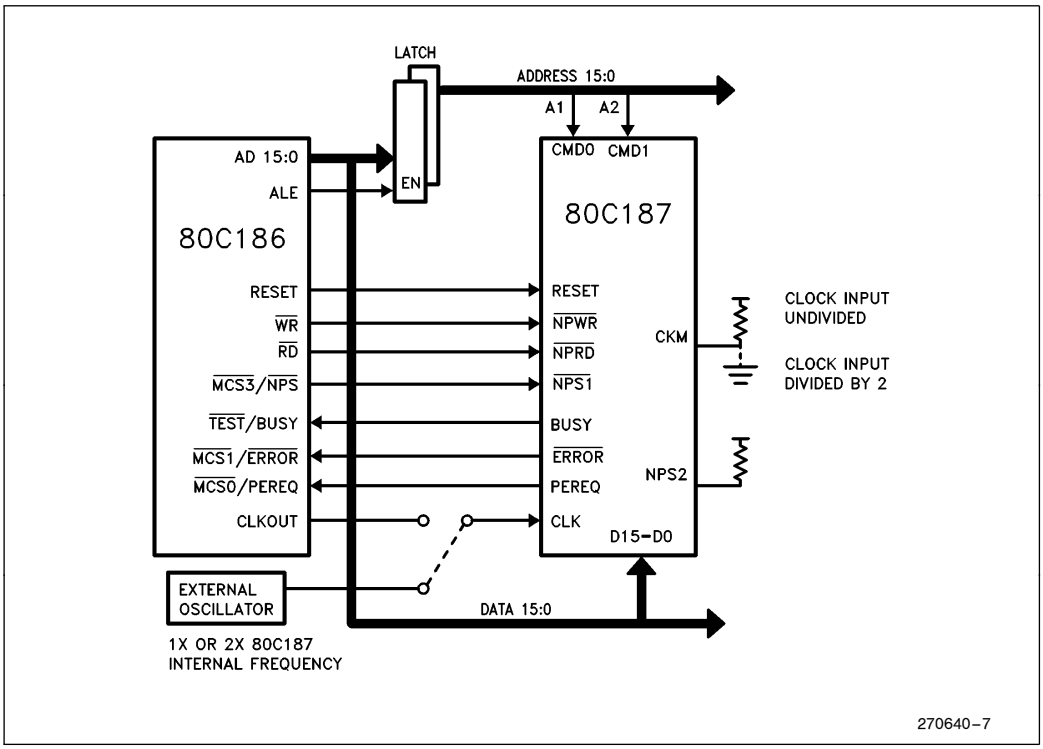
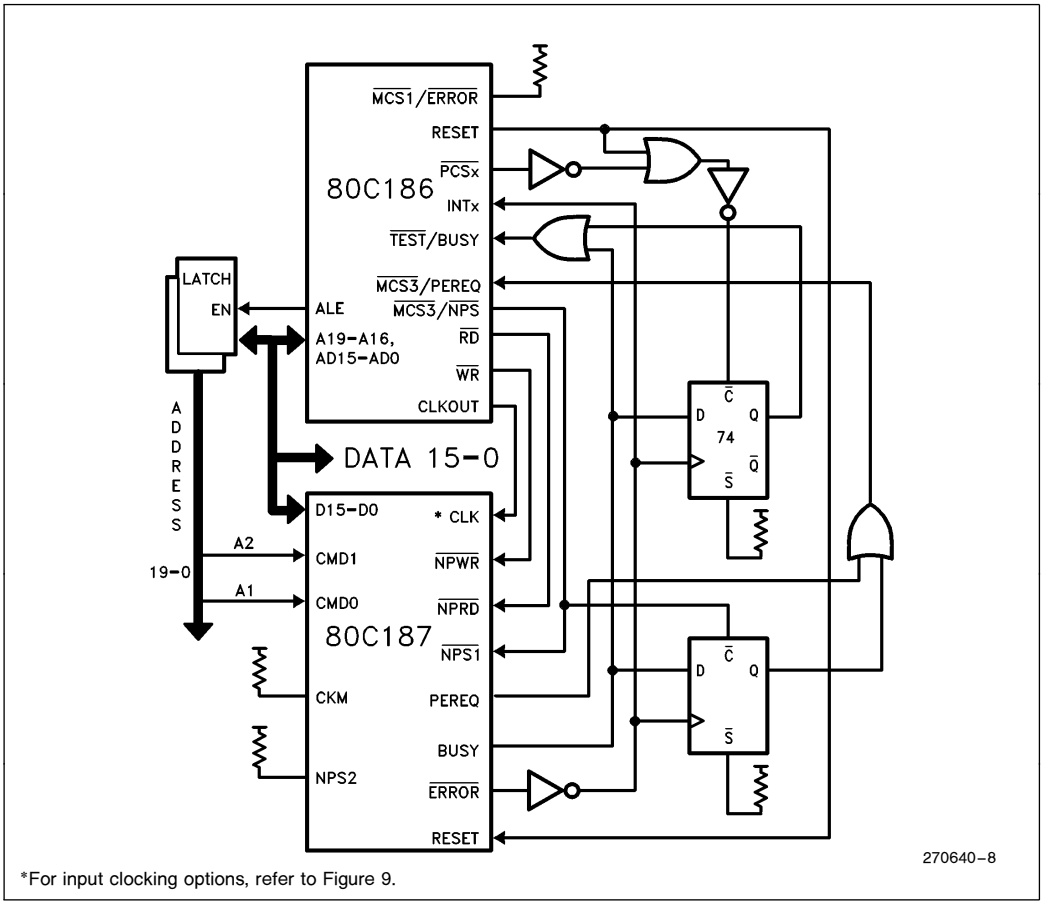


Figure 9. 80C186/80C187 System Configuration

System Configuration for 80186/ 80187-Compatible Exception Trapping

When the 80C187 $\overline{\text{ERROR}}$ output signal is connected directly to the 80C186 $\overline{\text{ERROR}}$ input, floating-point exceptions cause interrupt #16. However, existing software may be programmed to expect floating-point exceptions to be signalled over an external interrupt pin via an interrupt controller.

For exception handling compatible with the 80186/ 82188/8087, the 80C186 can be wired to recognize exceptions through an external interrupt pin, as Figure 10 shows. (Refer to the 80C186 Data Sheet for an explanation of the 80C186's signals.) With this arrangement, a flip-flop is needed to latch BUSY upon assertion of $\overline{\text{ERROR}}$. The latch can then be cleared during the exception-handler routine by forcing a $\overline{\text{PCS}}$ pin active. The latch must also be cleared at RESET in order for the 80C186 to work with the 80C187.



*For input clocking options, refer to Figure 9.

Figure 10. System Configuration for 8087-Compatible Exception Trapping

ELECTRICAL DATA

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

Absolute Maximum Ratings*

Case Temperature Under Bias (T_C) . . . 0°C to +85°C

Storage Temperature . . . . . −65°C to +150°C

Voltage on Any Pin
with Respect to Ground . . . . −0.5V 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.

Power and Frequency Requirements

The typical relationship between I_{CC} and the frequency of operation F is as follows:

$I_{CC_{typ}} = 55 + 5 * F$ mA where F is in MHz.

When the frequency is reduced below the minimum operating frequency specified in the AC Characteristics table, the internal states of the 80C187 may become indeterminate. The 80C187 clock cannot be stopped; otherwise, I_{CC} would increase significantly beyond what the equation above indicates.

DC Characteristics T_C = 0°C to +85°C, V_{CC} = +5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LOW Voltage	−0.5	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.5	V	
V_{ICL}	Clock Input LOW Voltage	−0.5	+0.8	V	
V_{ICH}	Clock Input HIGH Voltage	2.0	V_{CC} +0.5	V	
V_{OL}	Output LOW Voltage		0.45	V	I_{OL} = 3.0 mA
V_{OH}	Output HIGH Voltage	2.4		V	I_{OH} = −0.4 mA
I_{CC}	Power Supply Current		156 135	mA mA	16 MHz 12.5 MHz
I_{LI}	Input Leakage Current		±10	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	I/O Leakage Current		±10	μA	$0.45V \le V_{OUT} \le V_{CC} - 0.45V$
C_{IN}	Input Capacitance		10	pF	F_C = 1 MHz
C_O	I/O or Output Capacitance		12	pF	F_C = 1 MHz
C_{CLK}	Clock Capacitance		20	pF	F_C = 1 MHz

AC Characteristics

T_C = 0°C to +85°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V unless otherwise specified

Symbol	Parameter	12.5 MHz		16 MHz		Test Conditions
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	
T_{dvwh} (t6)	Data Setup to $\overline{NPWR}$	43		33		
T_{whdx} (t7)	Data Hold from $\overline{NPWR}$	14		14		
T_{rlrh} (t8)	$\overline{NPRD}$ Active Time	59		54		
T_{wlwh} (t9)	$\overline{NPWR}$ Active Time	59		54		
T_{avwl} (t10)	Command Valid to $\overline{NPWR}$	0		0		
T_{avrl} (t11)	Command Valid to $\overline{NPRD}$	0		0		
T_{mhrl} (t12)	Min Delay from PEREQ Active to $\overline{NPRD}$ Active	40		30		
T_{whax} (t18)	Command Hold from $\overline{NPWR}$	12		8		
T_{rhax} (t19)	Command Hold from $\overline{NPRD}$	12		8		
T_{ivcl} (t20)	$\overline{NPRD}$, $\overline{NPWR}$, RESET to CLK Setup Time	46		38		Note 1
T_{clih} (t21)	$\overline{NPRD}$, $\overline{NPWR}$, RESET from CLK Hold Time	26		18		Note 1
T_{rscl} (t24)	RESET to CLK Setup	21		19		Note 1
T_{clrs} (t25)	RESET from CLK Hold	14		9		Note 1
T_{cmdi} (t26)	Command Inactive Time					
	Write to Write	69		59		
	Read to Read	69		59		
	Read to Write	69		59		
	Write to Read	69		59		

NOTE:
1. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at a specific CLK edge.

Timing Responses

All timings are measured at 1.5V unless otherwise specified

Symbol	Parameter	12.5 MHz		16 MHz		Test Conditions
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	
T_{rhqz} (t27) T_{rlqv} (t28)	$\overline{NPRD}$ Inactive to Data Float* $\overline{NPRD}$ Active to Data Valid		18 50		18 45	Note 2 Note 3
T_{ilbh} (t29)	$\overline{ERROR}$ Active to Busy Inactive	104		104		Note 4
T_{wlbv} (t30)	$\overline{NPWR}$ Active to Busy Active		80		60	Note 4
T_{klml} (t31)	$\overline{NPRD}$ or $\overline{NPWR}$ Active to PEREQ Inactive		80		60	Note 5
T_{rhqh} (t32)	Data Hold from $\overline{NPRD}$ Inactive	2		2		Note 3
T_{rlbh} (t33)	RESET Inactive to BUSY Inactive		80		60	

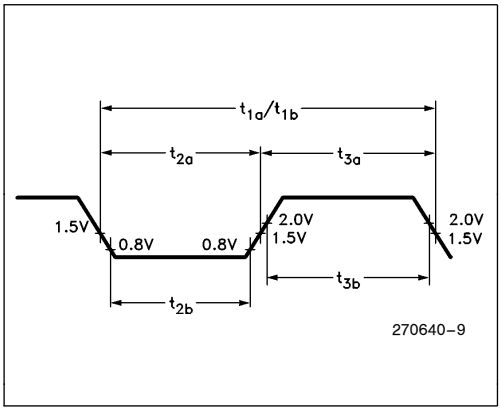
NOTES:
*The data float delay is not tested.
2. The float condition occurs when the measured output current is less than I_{OL} on $D_{15}-D_0$.
3. $D_{15}-D_0$ loading: $C_L = 100$ pF.
4. BUSY loading: $C_L = 100$ pF.
5. On last data transfer of numeric instruction.

Clock Timings

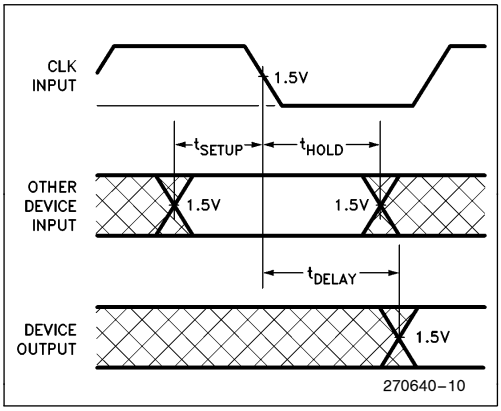
Symbol	Parameter		12.5 MHz		16 MHz*		Test Conditions
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	
T_{clcl} (t1a) (t1B)	CLK Period	CKM = 1 CKM = 0	80 40	250 125	N/A 31.25	N/A 125	Note 6 Note 6
T_{clch} (t2a) (t2b)	CLK Low Time	CKM = 1 CKM = 0	35 9		N/A 7		Note 6 Note 7
T_{chcl} (t3a) (t3b)	CLK High Time	CKM = 1 CKM = 0	35 13		N/A 9		Note 6 Note 8
T_{ch2ch1}(t4)				10		8	Note 9
T_{ch1ch2}(t5)				10		8	Note 10

NOTES:
*16 MHz operation is available only in divide-by-2 mode (CKM strapped LOW).
6. At 1.5V
7. At 0.8V
8. At 2.0V
9. CKM = 1: 3.7V to 0.8V at 16 MHz, 3.5V to 1.0V at 12.5 MHz
10. CKM = 1: 0.8V to 3.7V at 16 MHz, 1.0V to 3.5V at 12.5 MHz

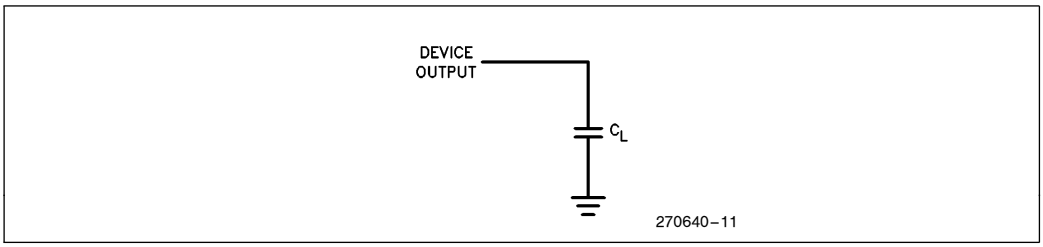
AC DRIVE AND MEASUREMENT POINTS—CLK INPUT



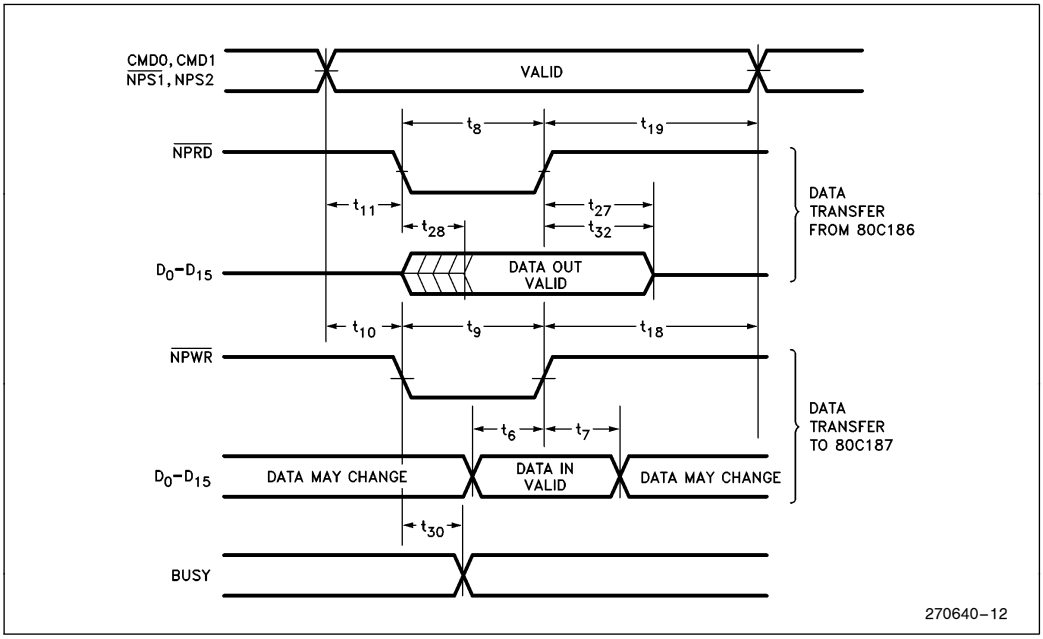
AC SETUP, HOLD, AND DELAY TIME MEASUREMENTS—GENERAL



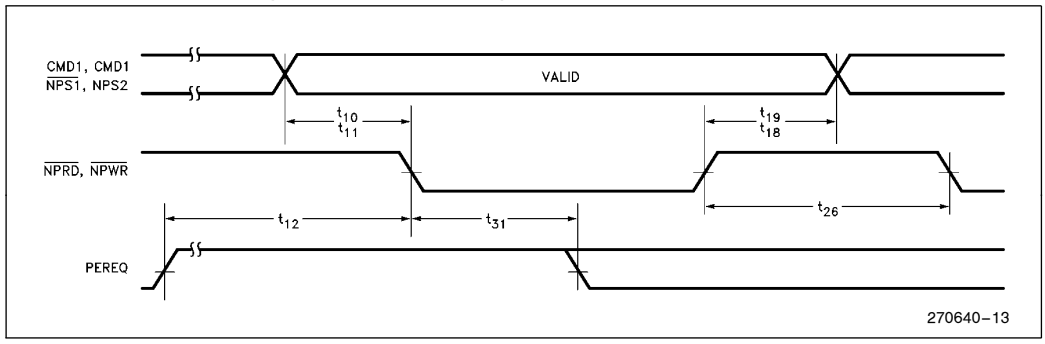
AC TEST LOADING ON OUTPUTS



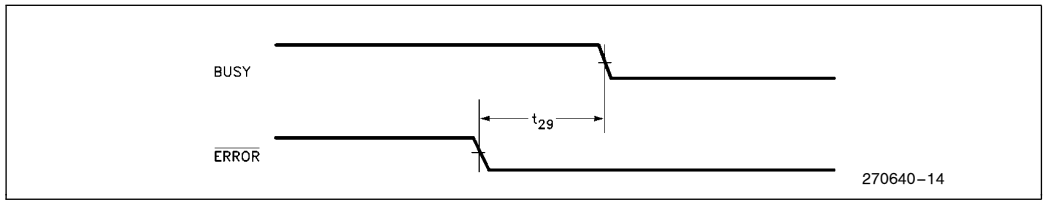
DATA TRANSFER TIMING (INITIATED BY CPU)



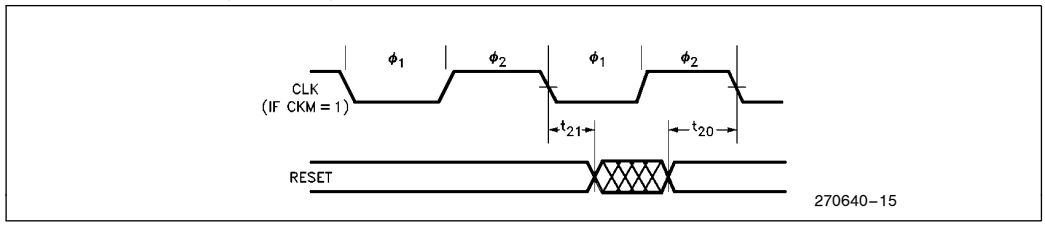
DATA CHANNEL TIMING (INITIATED BY 80C187)



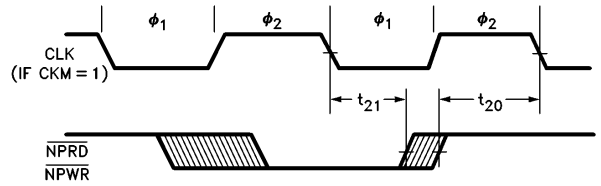
ERROR OUTPUT TIMING



CLK, RESET TIMING (CKM = 1)

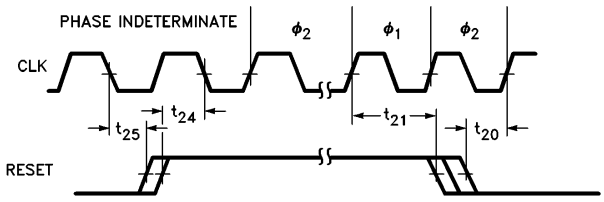


CLK, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ TIMING (CKM = 1)



270640–16

CLK, RESET TIMING (CKM = 0)

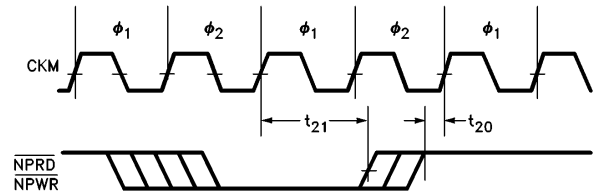


270640–17

RESET must meet timing shown to guarantee known phase of internal divide by 2 circuits.

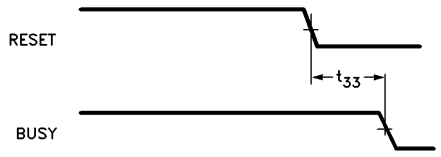
NOTE:
RESET, $\overline{\text{NPWR}}$, $\overline{\text{NPRD}}$ inputs are asynchronous to CLK. Timing requirements are given for testing purposes only, to assure recognition at a specific CLK edge.

CLK, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ TIMING (CKM = 0)



270640–18

RESET, BUSY TIMING



270640–19

80C187 EXTENSIONS TO THE CPU's INSTRUCTION SET

Instructions for the 80C187 assume one of the five forms shown in Table 12. 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 CPU's addressing modes.

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of CPU instructions (refer to Programmer's Reference Manual for the CPU). The DISP (displacement) is optionally present in instructions that have MOD and R/M fields. Its presence depends on the values of MOD and R/M, as for instructions of the CPU.

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 requests delaying processor access to the bus; and that no exceptions are detected during instruction execution. Timings are given in internal 80C187 clocks and include the time for opcode and data transfer between the CPU and the NPX. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Table 12. Instruction Formats

	Instruction									Optional Field
	First Byte				Second Byte					
1	11011	OPA		1	MOD		1	OPB	R/M	DISP
2	11011	MF		OPA	MOD		OPB *		R/M	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	

NOTES:

OP = Instruction opcode, possibly 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

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

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

*In FSUB and FDIV, the low-order bit of OPB is the R (reversed) bit

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

ESC = 11011

ST(i) = Register Stack Element *i*
- 000 = Stack Top
- 001 = Second Stack Element
- •
- •
- •
- 111 = Eighth Stack Element

80C187 Extensions to the 80C186 Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–3	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	DISP	40	65–72	59	67–71
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	DISP		90–101		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	DISP		74		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	DISP		296–305		
ST(i) to ST(0)	ESC 001	11000 ST(i)			16		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	DISP	58	93–107	73	80–93
ST(0) to ST(i)	ESC 101	11010 ST(i)			13		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	DISP	58	93–107	73	80–93
ST(0) to long integer memory	ESC 111	MOD 111 R/M	DISP		116–133		
ST(0) to extended real	ESC 011	MOD 111 R/M	DISP		83		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	DISP		542–564		
ST(0) to ST(i)	ESC 101	11001 ST (i)			14		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			20		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	DISP	48	78–85	67	77–81
ST(i) to ST(0)	ESC 000	11010 ST(i)			26		
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	DISP	48	78–85	67	77–81
ST(i) to ST(0)	ESC 000	11011 ST(i)			28		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			28		
FTST = Test ST(0)	ESC 001	1110 0100			30		
FUCOM = Unordered compare	ESC 101	11100 ST(i)			26		
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)			28		
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001			28		
FXAM = Examine ST(0)	ESC 001	11100101			32-40		
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110			22		
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000			26		
FLDPI = Load pi into ST(0)	ESC 001	1110 1011			42		
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001			42		

Shaded areas indicate instructions not available in 8087.

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

80C187 Extensions to the 80C186 Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–3	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			42		
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100			43		
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101			43		
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	DISP	44–52	77–92	65–73	77–91
ST(i) and ST(0)	ESC d P 0	11000 ST(i)			25–33[b]		
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	DISP	44–52	77–92	65–73	77–91[c]
ST(i) and ST(0)	ESC d P 0	1110 R R/M			28–36[d]		
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	DISP	47–57	81–102	68–93	82–93
ST(i) and ST(0)	ESC d P 0	1100 1 R/M			31–59[e]		
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	DISP	108	140–147[f]	128	142–146[g]
ST(i) and ST(0)	ESC d P 0	1111 R R/M			90[h]		
FSQRT[i] = Square root	ESC 001	1111 1010			124–131		
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101			69–88		
FPREM = Partial remainder of ST(0) ÷ ST(1)	ESC 001	1111 1000			76–157		
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101			97–187		
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100			68–82		
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100			72–78		
FABS = Absolute value of ST(0)	ESC 001	1110 0001			24		
FCHS = Change sign of ST(0)	ESC 001	1110 0000			26–27		

Shaded areas indicate instructions not available in 8087.

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 = 54 (When d = 0, 48–56, typical = 51).
f. Add 1 clock to the range when R = 1.
g. 153–159 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq$ ST(0) $\leq +\infty$.

80C187 Extensions to the 80C186 Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2–3	
TRANSCENDENTAL				
FCOS = Cosine of ST(0)	ESC 001	1111 1111		125–774[j]
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		193–499[j]
FPATAN = Partial arctangent	ESC 001	1111 0011		316–489
FSIN = Sine of ST(0)	ESC 001	1111 1110		124–773[j]
FSINCOS = Sine and cosine of ST(0)	ESC 001	1111 1011		196–811[j]
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		213–478
FYL2X[m] = ST(1) * $\log_2(ST(0))$	ESC 001	1111 0001		122–540
FYL2XP1[n] = ST(1) * $\log_2(ST(0) + 1.0)$	ESC 001	1111 1001		259–549
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		35
FSTSW AX = Store status word	ESC 111	1110 0000		17
FLDCW = Load control word	ESC 001	MOD 101 R/M	DISP	23
FSTCW = Store control word	ESC 001	MOD 111 R/M	DISP	21
FSTSW = Store status word	ESC 101	MOD 111 R/M	DISP	21
FCLEX = Clear exceptions	ESC 011	1110 0010		13
FSTENV = Store environment	ESC 001	MOD 110 R/M	DISP	146
FLDENV = Load environment	ESC 001	MOD 100 R/M	DISP	113
FSAVE = Save state	ESC 101	MOD 110 R/M	DISP	550
FRSTOR = Restore state	ESC 101	MOD 100 R/M	DISP	482
FINCSTP = Increment stack pointer	ESC 001	1111 0111		23
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		24
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		20
FNOP = No operations	ESC 001	1101 0000		14

Shaded areas indicate instructions not available in 8087.

NOTES:
j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 78 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. $0 \leq |ST(0)| < (2 - \sqrt{}(2))/2$, $-\infty < ST(1) < +\infty$.

DATA SHEET REVISION REVIEW

The following list represents the key differences between the -002 and the -001 version of the 80C187 data sheet. Please review this summary carefully.

1. Figure 10, titled ‘‘System Configuration for 8087—Compatible Exception Trapping’’, was replaced with a revised schematic. The previous configuration was faulty. Updated timing diagrams on Data Transfer Timing, Error Output, and RESET/BUSY.