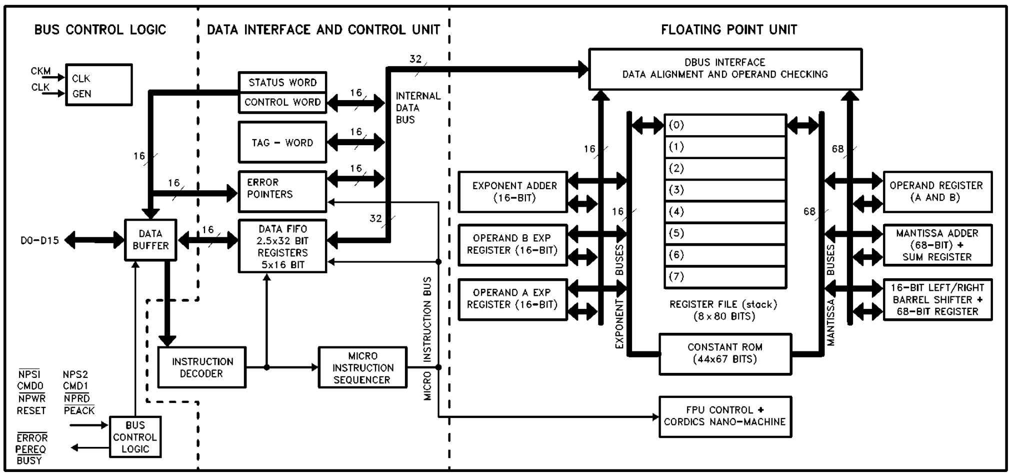
ADVANCE INFORMATION

M80C287
80-BIT CHMOS III NUMERIC PROCESSOR EXTENSION

Military

- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Implements Extended M387 Numerics Coprocessor Instruction Set**
- **Two to Three Times M8087/M80287 Performance at Equivalent Clock Speed**
- **Low Power Consumption**
- **Upward Object-Code Compatible from M8087 and M80287**
- **Interfaces with M80286 and M80C286 CPUs**
- **Expands CPU's Data Types to Include 32-, 64-, 80-Bit Floating Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends CPU'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**
- **Operates in Both Real and Protected Mode Systems**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in 40-pin CERDIP**
  (See Packaging Outlines and Dimensions, order #231369)
- **Military Temperature Range: −55°C to +125°C (T_C)**

The Intel M80C287 is a high-performance numerics processor extension that extends the architecture of the M80C286 CPU with floating point, extended integer, and BCD data types. A computing system that includes the M80C287 fully conforms to the IEEE Floating Point Standard. Using a numerics oriented architecture, the M80C287 adds over seventy mnemonics to the instruction set of the M80C286 CPU, making a complete solution for high-performance numerics processing. The M80C287 is implemented with 1.5 micron, high-speed CHMOS III technology and packaged in a 40-pin CERDIP. The M80C287 is upward object-code compatible from the M80287 and M8087 numerics coprocessors. With proper socket design, either an M80287 or an M80C287 can use the same socket.


BUS CONTROL LOGIC
DATA INTERFACE AND CONTROL UNIT
FLOATING POINT UNIT
CKM
CLK
CLK GEN
STATUS WORD
CONTROL WORD
TAG - WORD
ERROR POINTERS
DATA FIFO 2.5x32 BIT REGISTERS 5x16 BIT
DATA BUFFER
D0-D15
INTERNAL DATA BUS
INSTRUCTION DECODER
MICRO INSTRUCTION SEQUENCER
MICRO INSTRUCTION BUS
NPS1 NPS2 CMD0 CMD1 NPWR NPRD RESET PEACK
ERROR PEREQ BUSY
BUS CONTROL LOGIC
DBUS INTERFACE DATA ALIGNMENT AND OPERAND CHECKING
EXPONENT ADDER (16-BIT)
OPERAND B EXP REGISTER (16-BIT)
OPERAND A EXP REGISTER (16-BIT)
EXPONENT BUSES
(0) (1) (2) (3) (4) (5) (6) (7)
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
16
32
68

271092–1

Figure 1. M80C287 Block Diagram

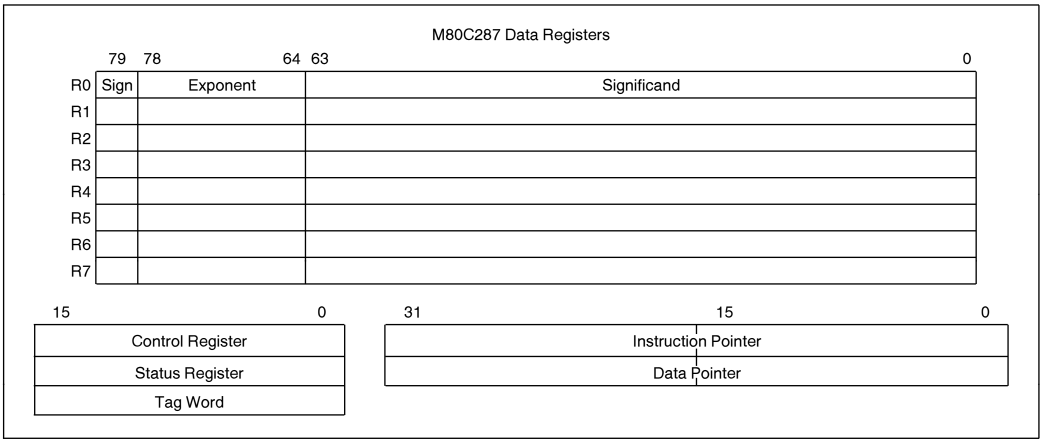


Figure 2. M80C287 Register Set

FUNCTIONAL DESCRIPTION

The M80C287 Numeric Processor Extension (NPX) 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 M80C287 effectively extends the register and instruction set of the 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 M80C287. Essentially, the M80C287 can be treated as an additional resource or an extension to the M80C286 CPU. The M80C286 CPU together with an M80C287 NPX can be used as a single unified system.

The M80C287 has two operating modes. After reset, the M80C287 is in the real-address mode. It can be placed into protected mode by executing the FSETPM instruction. It can be switched back to real-address mode by executing the FRSTPM instruction (note that this feature is useful only with CPU's that can also switch back to real-address mode). These instructions control the format of the administrative instructions FLDENV, FSTENV, FRSTOR, and FSAVE. Regardless of operating mode, all references to memory for numerics data or status information are performed by the M80C286 CPU, and therefore obey the memory-management and protection rules of the M80C286 CPU.

In real-address mode, a system that includes the M80C287 is completely upward compatible with software for the M8086/M8087 and for M80286/M80287 real-address mode.

In protected mode, a system that includes the M80C287 is completely upward compatible with software for M80286/M80287 protected mode systems.

The only differences of operation that may appear when M8086/M8087 programs are ported to a protected-mode M80C287 system are 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.

PROGRAMMING INTERFACE

The M80C287 adds to the CPU additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. To use the M80C287 requires no special programming tools, because all new instructions and data types are directly supported by the assembler and compilers for high-level languages. All 8086/8088 development tools that support the M8087 can also be used to develop software for the M80C286/M80C287 in real-address mode. All M80286 development tools that support the M80287 can also be used to develop software for the M80C286/M80C287. The M80C287 supports all M387 NPX instructions, producing the same binary results.

All communication between the M80C286 CPU and the M80C287 is transparent to applications software. The M80C286 CPU automatically controls the M80C287 whenever a numerics instruction is executed. All physical memory and virtual memory of the M80C286 CPU are available for storage of the instructions and operands of programs that use the M80C287. All memory addressing modes are available for addressing numerics operands.

The instructions that the M80C287 adds to the instruction set are listed at the end of this data sheet.

Data Types

Table 1 lists the seven data types that the M80C287 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 that correspond to the word size of the CPU; operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

Internally, the M80C287 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.

Register Set

Figure 2 shows the M80C287 register set. When an M80C287 is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

DATA REGISTERS

M80C287 computations use the M80C287's data registers. These eight 80-bit registers provide the equivalent capacity of 20 32-bit registers. Each of the eight data registers in the M80C287 is 80 bits wide and is divided into "fields" corresponding to the NPX's extended-precision real data type.

The M80C287 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 M80C287 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 register 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 M80C287. It may be read and inspected by programs.

Bit 15, the B-bit (busy bit) is included for M8087 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 $\overline{\text{BUSY}}$ output of M80C287.

Bits 13–11 (TOP) point to the M80C287 register that is the current top-of-stack.

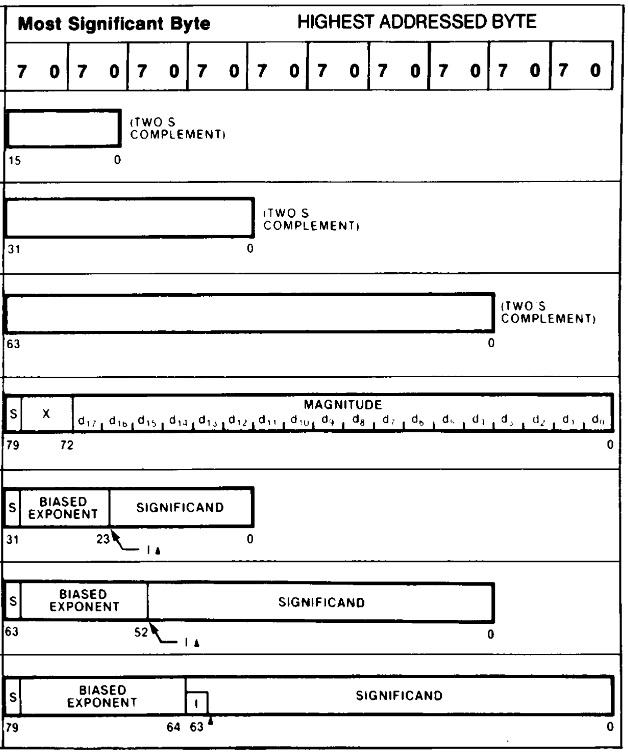
The four numeric condition code bits (C_3–C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2 through 5.

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 ($C_1 = 0$).

Table 1. M80C287 Data Type Representation in Memory

Data Formats	Range	Precision
Word Integer	$\pm 10^4$	16 Bits
Short Integer	$\pm 10^9$	32 Bits
Long Integer	$\pm 10^{18}$	64 Bits
Packed BCD	$\pm 10^{18}$	18 Digits
Single Precision	$\pm 10^{\pm 38}$	24 Bits
Double Precision	$\pm 10^{\pm 308}$	53 Bits
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits



271092–2

NOTES:
1. S = Sign bit (0 = positive, 1 = negative)
2. d_n = Decimal digit (two per byte)
3. X = Bits have no significance: M80C287 ignores when loading, zeroes 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: $(-1)^S (2^{E-BIAS}) (F_0 F_1 \ldots)$

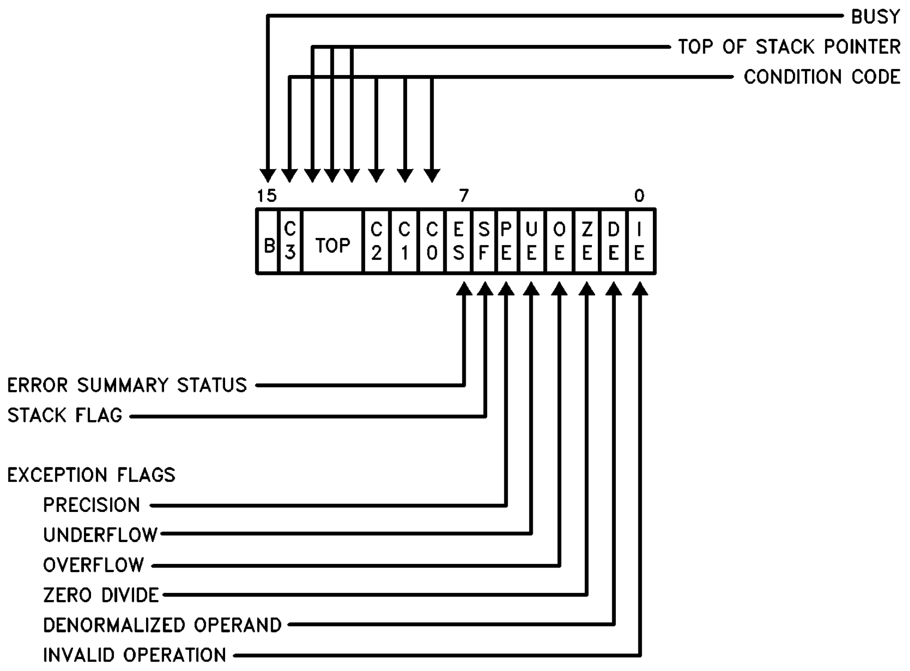
15							0
TAG (7)	TAG (6)	TAG (5)	TAG (4)	TAG (3)	TAG (2)	TAG (1)	TAG (0)

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 3. M80C287 Tag Word

Figure 4 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the M80C287 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 M80C287 is activated immediately.



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 4. Status 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/$\overline{\text{U}}$	Reduction 0 = Complete 1 = Incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of Comparison (See Table 2.4)		Zero or O/$\overline{\text{U}}$	Operand is Not Comparable (Table 2.4)
FXAM	Operand Class (See Table 2.5)		Sign or O/$\overline{\text{U}}$	Operand Class (Table 2.5)
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant Loads, FXTRACT, FLD, FILD, FBLD, FSTP (Ext Real)	UNDEFINED		Zero or O/$\overline{\text{U}}$	UNDEFINED
FIST, FBSTP, FRNDINT, FST FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/$\overline{\text{U}}$	UNDEFINED
FPTAN, FSIN, FCOS, FSINCOS	UNDEFINED		Roundup or O/$\overline{\text{U}}$ 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/$\overline{\text{U}}$	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.

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.

C2	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	0	−Denormal

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.

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 M80C287 recognizes.

The high-order byte of the control word configures the M80C287 operating mode, including precision, rounding, and infinity control.

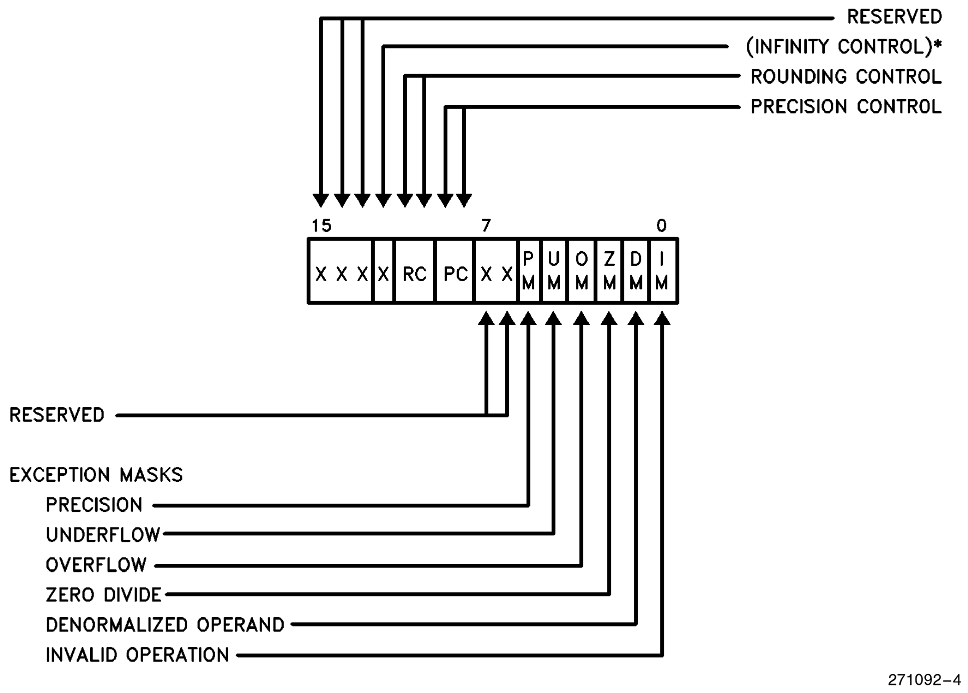
- The "infinity control bit" (bit 12) is not meaningful to the M80C287, and programs must ignore its value. To maintain compatibility with the M8087 and M80287, this bit can be programmed; however, regardless of its value, the M80C287 always treats infinity in the affine sense ($-\infty < +\infty$). 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 M80C287 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.

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 numeric instruction, the M80C287 contains registers that aid in diagnosis. These registers supply the opcode of the failing numeric instruction, the address of the instruction, and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written exception handlers. Whenever the M80C287 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. CPUs with 32-bit internal architectures contain 32-bit versions of these registers and do not use the contents of the NPX registers. This difference is not apparent to programmers, however.

The instruction and data pointers appear in one of four formats depending on the operating mode of the system (protected mode or real-address mode) and (for CPUs with 32-bit internal architectures) depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). (See Figures 6 and 7) The ESC instructions 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.



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 M80C287. To maintain compatibility with the M80287, this bit can be programmed; however, regardless of its value, the M80C287 treats infinity in the affine sense ($-\infty < +\infty$).

Figure 5. Control Word

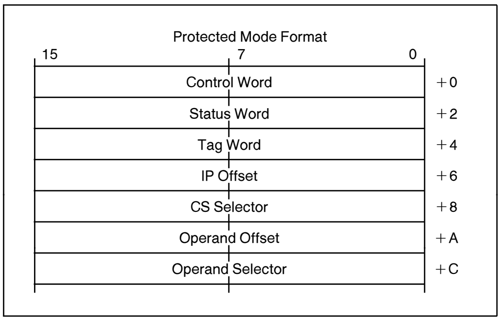


Figure 6. Protected Mode Instruction and Data Pointer Image in Memory

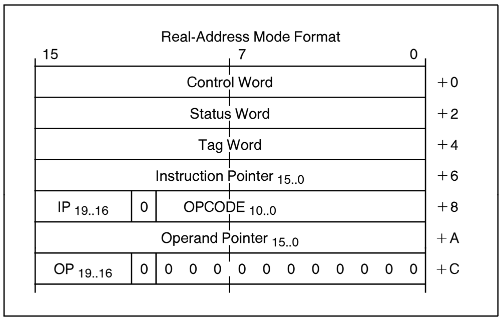


Figure 7. Real Mode Instruction and Data Pointer Image in Memory

Table 6. CPU Interrupt Vectors Reserved for NPX

Interrupt Number	Cause of Interrupt
7	In a system with a CPU that has control registers, an ESC instruction was encountered when EM or TS of CPU 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	In a protected-mode system, an operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for expand-up segments, zero for expand-down segments) and spanned inaccessible addresses (See Note). 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. The segment overrun exception should be handled by executing an FNINIT instruction (i.e., an FINIT without a preceding WAIT). The exception can be avoided by never allowing numerics operands to cross the end of a segment.
13	In a protected-mode system, the first word 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 M80C287 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 CPU 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.

NOTE:
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 FFFCH will span addresses FFFC–FFFFH and 0000–0003H; however addresses FFFEH and FFFFH are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible segments but intermediate bytes of the operand fall in a not-present segment or page or in a segment or page to which the procedure does not have access rights.

Interrupt Description

CPU interrupts are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 6 shows these interrupts and their functions.

Exception Handling

The M80C287 detects six different exception conditions that can occur during instruction execution. Table 7 lists the exception conditions in order of precedence, showing for each the cause and the

Table 7. Exceptions

Exception	Cause	Default Action (If Exception is Masked)
Invalid Operation	Operation on a signalling 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.	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. ⅓); the result is rounded according to the rounding mode.	Normal processing continues.

default action taken by the M80C287 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 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 M80C287 saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

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 80287 after RESET. For compatibility with the M8087 and M80287, 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).

M80C286/M80C287 initialization software should execute an FNINIT instruction (i.e an FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for either M80287 or M80C287 software, but Intel recommends its use to help ensure upward compatibility with other processors.

M8087 and M80287 Compatibility

This section summarizes the differences between the M80C287 and the M80287. Any migration from the M8087 directly to the M80C287 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 M80C287 to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

GENERAL DIFFERENCES

The M80C287 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 M80C287 resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the M80C287 does not support the special data formats pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

The denormal exception has a different purpose on the M80C287. A system that uses the denormal-exception handler solely to normalize the denormal operands, would better mask the denormal exception on the M80C287. The M80C287 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 M80C287:

1. When the overflow or underflow exception is masked, the M80C287 differs from the M80287 in rounding when overflow or underflow occurs. The M80C287 produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the M80C287 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 M80C287 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 signalling NaN, FLD *single/double precision* signals an invalid-operand exception.
12. The M80C287 only generates quiet NaNs (as on the M80287); however, the M80C287 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 M8087/M80287 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, ∞) generates the invalid operation exception.
    - FSCALE(finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE(finite, $+\infty$) generates -in 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 M80C287 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.

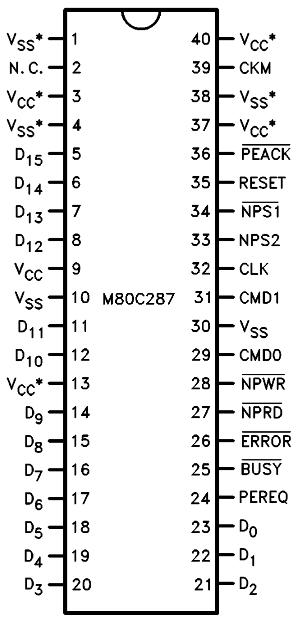
HARDWARE INTERFACE

Signal Description

In the following signal descriptions, the M80C287 pins are grouped by function as follows:

1. Execution control—CLK, CKM, RESET
2. NPX handshake—PEREQ, $\overline{\text{PEACK}}$, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$
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 8 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. Figure 8 shows the locations of pins on the CERDIP package. Table 9 helps to locate pin identifiers in Figure 8.



271092–5

*The corresponding pins of the M80287 differ.

Figure 8. CERDIP Pin Configuration

Table 8. 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
$\overline{\text{PEACK}}$	Processor Extension ACKnowledge	Low	I
$\overline{\text{BUSY}}$	Busy status	Low	O
$\overline{\text{ERROR}}$	Error status	Low	O
D15–D0	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

Table 9. CERDIP Pin Cross-Reference

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

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.

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.

SYSTEM RESET (RESET)

A LOW to HIGH transition on this pin causes the M80C287 to terminate its present activity and to enter a dormant state. RESET must remain active (HIGH) for at least four CLK periods (i.e., the RESET signal presented to the M80C287 must be at least four M80C287 clocks long, regardless of the frequency of the CPU). Note that the M80C287 is active internally for 25 clock cycles after the termination of the RESET signal (the HIGH to LOW transition of RESET); therefore, the first instruction should not be written to the M80C287 until 25 clocks after the falling edge of RESET. Table 10 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive states.

Table 10. Output Pin Status during Reset

Output Pin Name	Value During Reset
$\overline{\text{BUSY}}$	HIGH
$\overline{\text{ERROR}}$	HIGH
PEREQ	LOW
$D_{15}-D_0$	Tristate OFF

PROCESSOR EXTENSION REQUEST (PEREQ)

When active, this pin signals to the CPU that the M80C287 is ready for data transfer to/from its data FIFO. With M80286 or M80C286 CPUs, PEREQ can be deactivated after assertion of $\overline{\text{PEACK}}$. These CPUs rely on the NPX to deassert PEREQ when all operands have been transfered. When there are more than five data transfers, PEREQ is deactiviated after the first three transfers and subsequently after every four transfers. This signal always goes inactive before $\overline{\text{BUSY}}$ goes inactive.

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

When active, this pin signals to the CPU that the M80C287 is currently executing an instruction. It should be connected to the CPU's $\overline{\text{BUSY}}$ pin. During the RESET sequence this pin is HIGH.

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

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 $\overline{\text{ERROR}}$ pin of the CPU. $\overline{\text{ERROR}}$ can change state only when $\overline{\text{BUSY}}$ is active.

PROCESSOR EXTENSION ACKNOWLEDGE ($\overline{\text{PEACK}}$)

During execution of escape instructions, an M80286 or M80C286 CPU asserts $\overline{\text{PEACK}}$ to acknowledge that the request signal (PEREQ) has been recognized and that data transfer is in progress. The M80286/M80C286 also drives this signal HIGH during RESET.

This input may be asynchronous with respect to the M80C287 clock except during a RESET sequence, when it must satisfy setup and hold requirements relative to RESET.

DATA PINS (D_{15}–D_0)

These bidirectional pins are used to transfer data and opcodes between the CPU and M80C287. 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 ($\overline{\text{NPWR}}$)

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

NUMERIC PROCESSOR READ ($\overline{\text{NPRD}}$)

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

NUMERIC PROCESSOR SELECTS ($\overline{\text{NPS1}}$ and NPS2)

Concurrent assertion of these signals indicates that the CPU is performing an escape instruction and enables the M80C287 to execute that instruction. No data transfer involving the M80C287 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 M80C287.

SYSTEM POWER (V_{CC})

System power provides the +5V±5% 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 on the front page, the M80C287 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 M80C287 and transferring outputs from the M80C287 to memory. A dedicated communication protocol makes possible high-speed transfer of opcodes and operands between the M80C286 CPU and M80C287.

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 de-

Table 11. Bus Cycles Definition

$\overline{NPS1}$	NPS2	CMD0	CMD1	$\overline{NPRD}$	$\overline{NPWR}$	Bus Cycle Type
x	0	x	x	x	x	M80C287 not selected
1	x	x	x	x	x	M80C287 not selected
0	1	0	0	1	0	Opcode write to M80C287
0	1	0	0	0	1	CW or SW read from M80C287
0	1	1	0	0	1	Read data from M80C287
0	1	1	0	1	0	Write data to M80C287
0	1	0	1	1	0	Write exception pointers
0	1	0	1	0	1	Reserved
0	1	1	1	0	1	Reserved
0	1	1	1	1	0	Reserved

coder 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 $\overline{BUSY}$, PEREQ, and $\overline{ERROR}$ signals that synchronize M80C287 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 11 defines the types of M80C287 bus cycles.

M80C287 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. The correspondence between I/O addresses and control signals is defined by Table 12. To guarantee correct operation of the NPX, programs must not perform any I/O operations to these reserved port addresses.

Table 12. I/O Address Decoding

I/O Address (Hexadecimal)	M80C287 Select and Command Inputs			
	NPS2	$\overline{NPS1}$	CMD1	CMD0
00F8	1	0	0	0
00FA	1	0	0	1
00FC	1	0	1	0

CPU/NPX SYNCHRONIZATION

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

$\overline{BUSY}$ is used to synchronize instruction transfer from the M80C286 CPU to the M80C287. When the M80C287 recognizes an ESC instruction, it asserts $\overline{BUSY}$. For most ESC instructions, the M80C286 CPU waits for the M80C287 to deassert $\overline{BUSY}$ 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. Thus, memory access on behalf of the NPX always obeys the rules applicable to the mode of the CPU, whether the CPU be in real-address mode or protected mode.

Once the M80C286 CPU initiates an M80C287 instruction that has operands, the M80C286 CPU waits for PEREQ signals that indicate when the M80C287 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 M80C287 executes the ESC instruction.

In M8086/M8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In M80C287 systems, however, WAIT instructions are required only for operand synchronization; namely, after NPX stores to memory (except FSTSW and FSTCW) or load from memory. (In M80C286/M80C287 systems, WAIT is required before FLDENV and FRSTOR; with other CPU's, WAIT is not required in these cases.) 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 CPU, the M80C287 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.

Bus Operation

With respect to bus interface, the M80C287 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 M80C287 also examines the $\overline{\text{NPRD}}$ and $\overline{\text{NPWR}}$ 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 M80C287 activates its $\overline{\text{BUSY}}$ output some time after the leading edge of the $\overline{\text{NPRD}}$ or $\overline{\text{NPWR}}$ signal. Input and output data are referenced to the trailing edges of the $\overline{\text{NPRD}}$ and $\overline{\text{NPWR}}$ signals.

The M80C287 activates the PEREQ signal when it is ready for data transfer. In M80286/80C286 systems, the CPU activates $\overline{\text{PEACK}}$ when no more data transfers are required, which causes the M80C287 to deactivate PEREQ, halting the data transfer.

M80287/M80C287 Socket Compatibility and CPU Interfacing

In general, the M80C287 can fit in existing M80287 sockets, provided that the necessary connections to V_{CC} and V_{SS} are made and that the clock requirements are met. The pinouts for the M80C287 are identical to those of the M80287 except for the pins marked by asterisk (*) in Figure 8. The pins marked by asterisk are status lines for monitoring ESCAPE instructions and bus cycles. These lines are not critical for proper operation of an M80287. Note that when the clock is fed in directly (CKM = 1) the M80C287 requires a 50% duty cycle clock signal, whereas the M80287 requires a 33% duty cycle. Also note that with CKM = 0, the M80C287 divides the clock input by two, not by three as on the M80287.

The interface between the M80C287 and the M80286/M80C286 CPU (illustrated in Figure 9) has these characteristics:

- The M80C287 resides on the local data bus of the CPU.
- The CPU and M80C287 share the same RESET signals. They may also share the same clock input; however, for greatest performance, an external oscillator may be needed.
- The corresponding $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$, PEREQ, and $\overline{\text{PEACK}}$ pins are connected together.
- NPS2 is tied HIGH permanently, while $\overline{\text{NPS1}}$, CMD1, and CMD0 come from the latched address pins. The M80286 generates I/O addresses 00F8H, 00FAH, and 00FCH during NPX bus cycles. Address 00FEH is reserved.
- The M80C287 $\overline{\text{NPRD}}$ and $\overline{\text{NPWR}}$ inputs are connected to I/O read and write signals from local bus control logic.

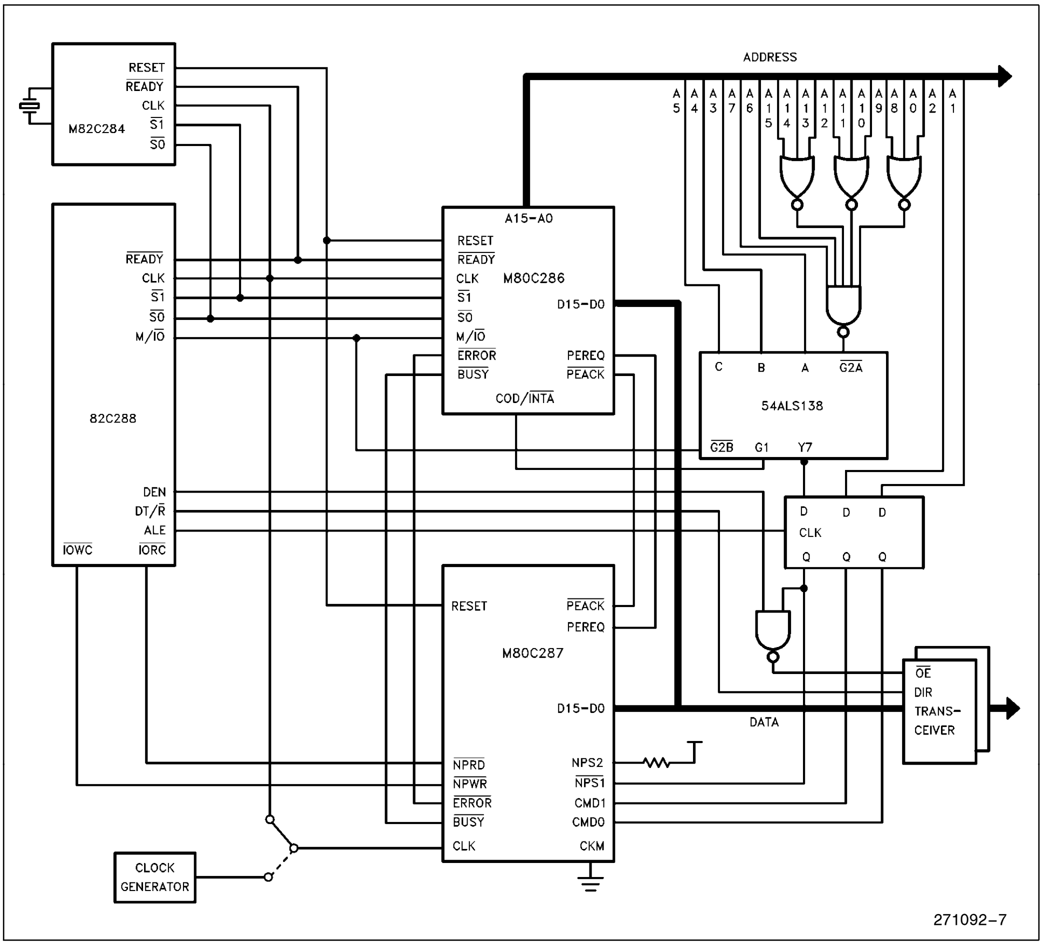


Figure 9. M80C286/M80C287 System Configuration

ELECTRICAL DATA

ABSOLUTE MAXIMUM RATINGS*

Case temperature (T_C) 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	1.5 Watt

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

**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:

$$I_{CCtyp} = 55 + 5*F \text{ mA}, \text{ where F is in MHz.}$$

When the frequency is reduced below the minumum operating frequency specified in the AC Characteristics table, the internal states of the M80C287 may become indeterminate. The M80C287 clock cannot be stopped; otherwise, I_{CC} would increase significantly beyond what the equation above indicates.

Operating Conditions

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

DC CHARACTERISTICS (Over Specified Operating Conditions)

Symbol	Parameter	Min	Max	Units	Comments
V_{IL}	Input LOW Voltage	-0.5	$+0.8$	V	
V_{IH}	Input HIGH Voltage	2.2	$V_{CC}+0.5$	V	
V_{ICL}	Clock Input LOW Voltage	-0.5	$+0.8$	V	
V_{ICH}	Clock Input HIGH Voltage	2.2	$V_{CC}+0.5$	V	
V_{OL}	Output LOW Voltage		0.45	V	$I_{OL} = 3$ mA
V_{OH}	Output HIGH Voltage	2.4		V	$I_{OH} = -800\ \mu A$
I_{CC}	Power Supply Current		115	mA	CLK = 10 MHz
I_{LI}	Input Leakage Current		± 10	μA	$0V \leq V_{IN} \leq V_{CC}$
I_{LO}	I/O Leakage Current		± 10	μA	$0.45V \leq V_{OUT} \leq V_{CC} - 0.45$
C_{IN}	Input Capacitance		10	pF	$F_C = 1$ MHz
C_O	I/O or Output Capacitance		20	pF	$F_C = 1$ MHz
C_{CLK}	Clock Capacitance		12	pF	$F_C = 1$ MHz

AC CHARACTERISTICS (Over Specified Operating Conditions)

Symbol	Parameter	10 MHz		Comments
		Min (ns)	Max (ns)	
t6	Data setup to $\overline{\text{NPWR}}$	50		
t7	Data hold from $\overline{\text{NPWR}}$	18		
t8	$\overline{\text{NPWR}}$ active time	91.5		
t9	$\overline{\text{NPRD}}$ active time	91.5		
t10	Command valid to $\overline{\text{NPRD}}$	0		
t11	Command valid to $\overline{\text{NPWR}}$	0		
t12	Min delay from PEREQ active to $\overline{\text{NPRD}}$ active	50		
t33	$\overline{\text{PEACK}}$ active time	61.5		
t34	$\overline{\text{PEACK}}$ inactive time	76.5		
t35	$\overline{\text{PEACK}}$ inactive to $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ inactive	40		
t36	$\overline{\text{PEACK}}$ active setup to $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ active	40		
t37	$\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ inactive to $\overline{\text{PEACK}}$ active	−30		
t38	$\overline{\text{PEACK}}$ Setup to RESET Falling Edge	80		
t39	$\overline{\text{PEACK}}$ Hold from RESET Falling Edge	80		
t18	Command hold from $\overline{\text{NPWR}}$	20		
t19	Command hold from $\overline{\text{NPRD}}$	20		
t20	$\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, RESET to CLK setup time	54		Note 1
t21	$\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, RESET from CLK hold time	38		Note 1
t24	RESET to CLK setup	22		Note 1
t25	RESET from CLK hold	20		Note 1
t26	Command inactive time			
	Write to write	76.5		
	Read to read	76.5		
	Read to write	76.5		
	Write to read	76.5		

NOTE:
1. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at a specific CLK edge (not tested).

Timing Responses

Symbol	Parameter	10 MHz Min (ns)	10 MHz Max (ns)	Comments
t27	$\overline{\text{NPRD}}$ inactive to data float		25	Note 1
t28	$\overline{\text{NPRD}}$ active to data valid		60	Note 2
t29	$\overline{\text{ERROR}}$ active to $\overline{\text{BUSY}}$ inactive	100		Note 3
t30	$\overline{\text{NPWR}}$ active to $\overline{\text{BUSY}}$ active		100	Note 3
t31	$\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ or $\overline{\text{PEACK}}$ active to PEREQ inactive		100	Note 4
t32	Data hold from $\overline{\text{NPRD}}$ inactive	3		Note 2

NOTES:
1. The float condition occurs when the measured output current is less than I_{OL} on D_{15}–D_0.
2. D_{15}–D_0 loading: $C_L = 100$pf.
3. BUSY# loading: $C_L = 100$pf.
4. On last data transfer of numeric instruction.

Clock Timings

Symbol	Parameter		10 MHz Min (ns)	10 MHz Max (ns)	Comments
t1a	CLK period	CKM = 1	100	250	
t1b		CKM = 0	50	125	
t2a	CLK low time	CKM = 1	35		
t2b		CKM = 0	11		Note 5, 9
t3a	CLK high time	CKM = 1	35		
t3b		CKM = 0	18		Note 6, 9
t4	CLK fall time			10	Note 7
t5	CLK rise time			10	Note 8

NOTES:
5. At 0.8V.
6. At 2.0V.
7. CKM = 1: 3.5V to 1.0V
8. CKM = 1: 1.0V to 3.5V
9. Proper operation can also be achieved by meeting the CPU specification

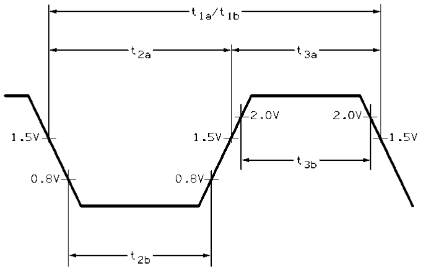


Figure 10. AC Drive and Measurement Points—CLK Input

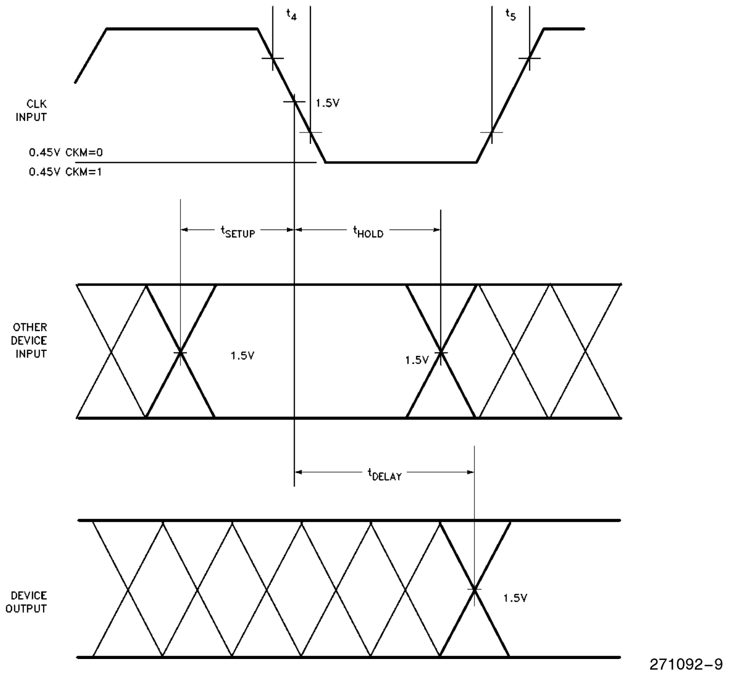


Figure 11. AC Setup, Hold, and Delay Time Measurements—General

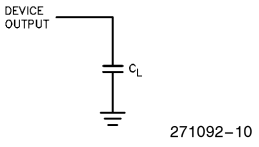


Figure 12. AC Test Loading on Outputs

RESET, $\overline{\text{NPWR}}$, $\overline{\text{NPRD}}$ inputs are asynchronous to CLK. Timing requirements in Figures 16 through 19 are given for testing purposes only, to assure recognition at a specific CLK edge.

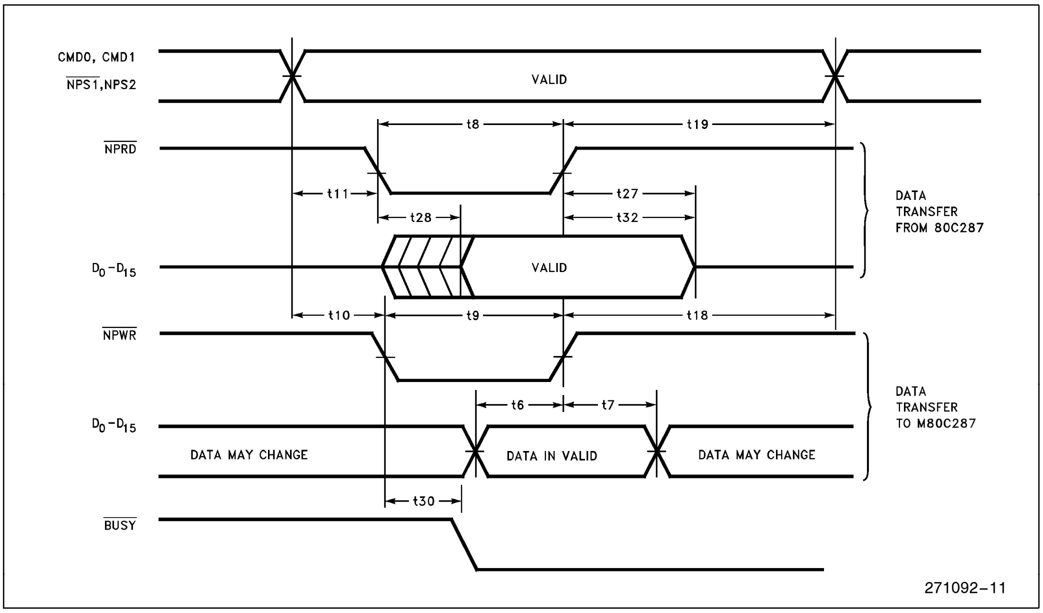


Figure 13. Data Transfer Timing (Initiated by CPU)

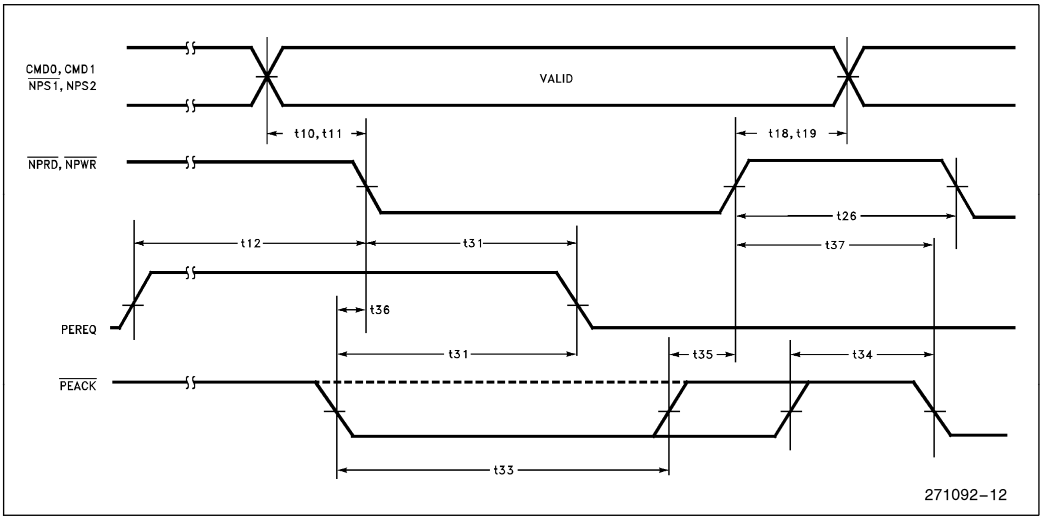


Figure 14. Data Channel Timing (Initiated by M80C287)

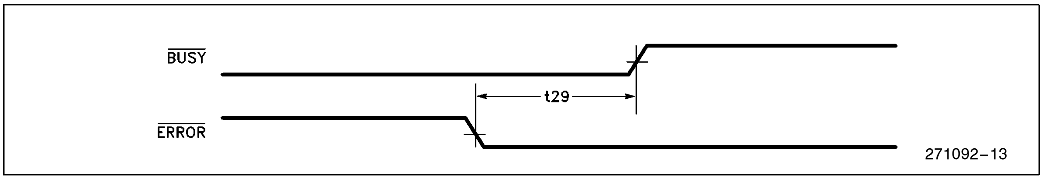


Figure 15. $\overline{\text{ERROR}}$ Output Timing

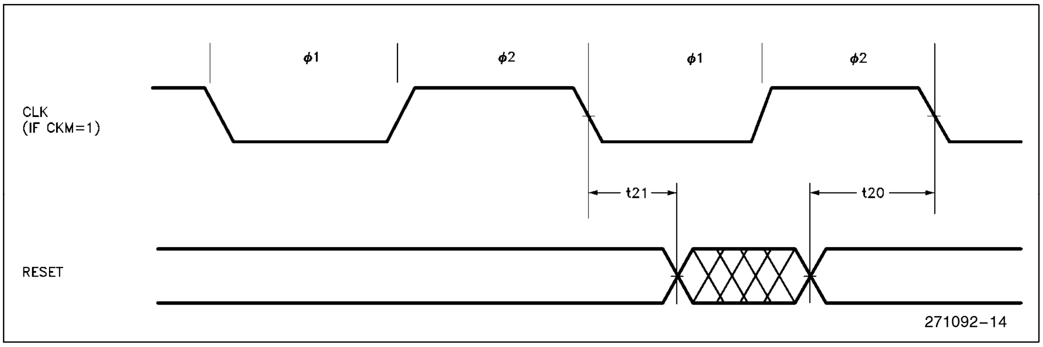


Figure 16. CLK, RESET Timing (CKM = 1)

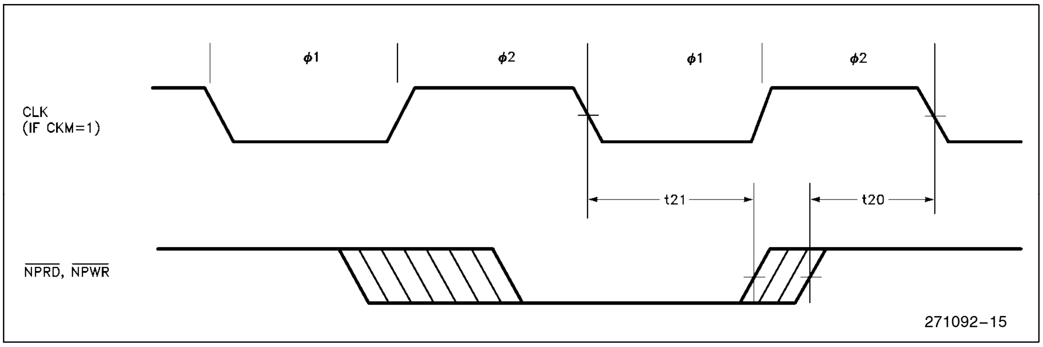
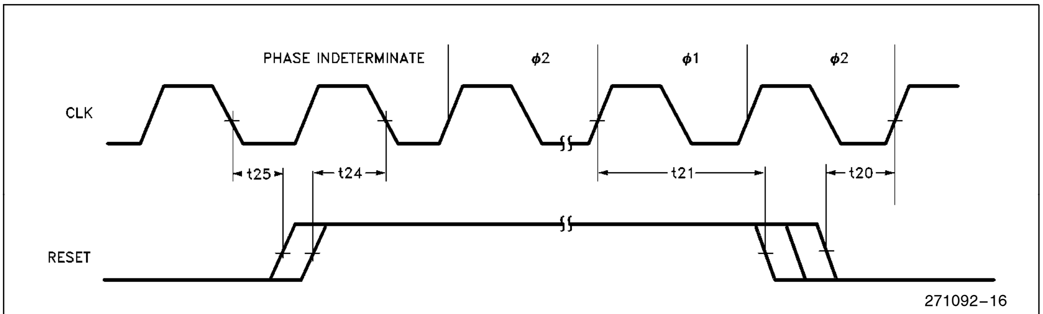


Figure 17. CLK, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ Timing (CKM = 1)



NOTE:
RESET must meet timing shown to guarantee known phase of internal divide by 2 circuit.

Figure 18. CLK, RESET Timing (CKM = 0)

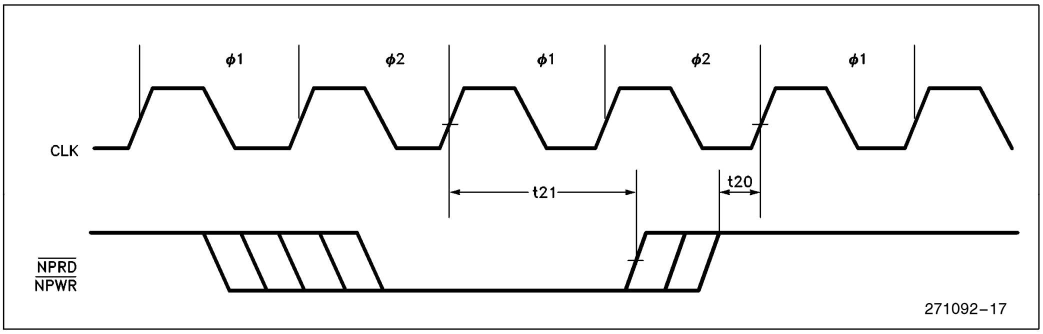


Figure 19. CLK, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ Timing (CKM = 0)

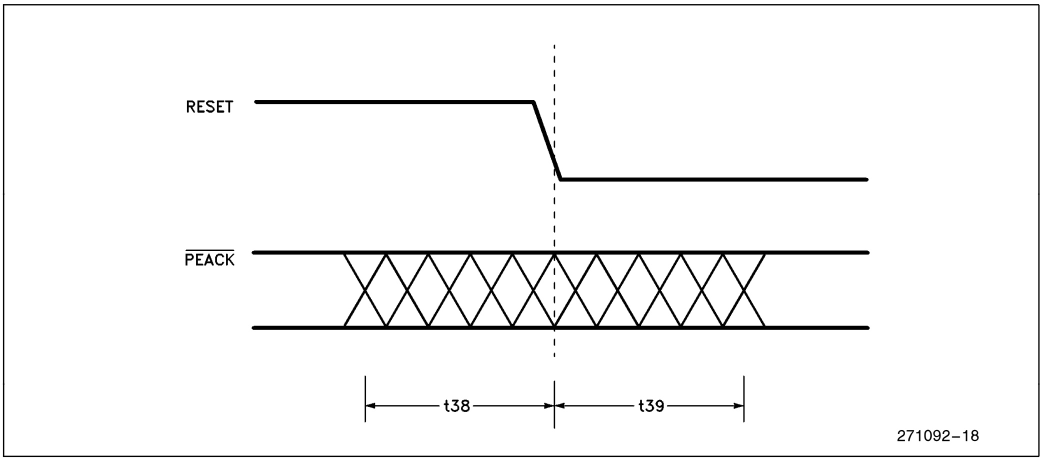


Figure 20. RESET, $\overline{\text{PEACK}}$ Setup and Hold Timing

M80C287 EXTENSIONS TO THE CPU'S INSTRUCTION SET

Instructions for the M80C287 assume one of the five forms shown in Table 13. 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 M80C287 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 13. 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	

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)
- 1–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 the 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

M80C287 Extension to the CPU's 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[1]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	36	61–68	45	61–65
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP	76–87			
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP	48			
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP	270–279			
ST(i) to ST(0)	ESC 001	11000 ST(i)		21			
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	51	86–100	56	88–101
ST(0) to ST(i)	ESC 101	11010 ST(i)		18			
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	51	86–100	56	88–101
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP	91–108			
ST(0) to extended real	ESC 011	MOD 111 R/M	SIB/DISP	61			
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP	520–542			
ST(0) to ST(i)	ESC 101	11001 ST (i)		19			
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)		25			
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	42	72–79	51	71–75
ST(i) to ST(0)	ESC 000	11010 ST(i)		31			
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	SIB/DISP	42	72–79	51	71–77
ST(i) to ST(0)	ESC 000	11011 ST(i)		33			
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001		33			
FTST = Test ST(0)	ESC 001	1110 0100		35			
FUCOM = Unordered compare	ESC 101	11100 ST(i)		31			
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)		33			
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001		33			
FXAM = Examine ST(0)	ESC 001	11100101		37–45			
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110		27			
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000		31			
FLDPI = Load pi into ST(0)	ESC 001	1110 1011		47			
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001		47			

Shaded areas indicate instructions not available in M8087/M80287.

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

M80C287 Extension to the CPU's 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		47			
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100		48			
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101		48			
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	40–48	73–78	49–79	71–85
ST(i) and ST(0)	ESC d P 0	11000 ST(9)		30–38[2]			
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	40–48	73–98	49–77	71–83[3]
ST(i) and ST(0)	ESC d P 0	1110 R R/M		33–41[4]			
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	43–51	77–88	52–77	76–87
ST(i) and ST(0)	ESC d P 0	1100 1 R/M		25–53[5]			
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	105	136–143[6]	114	136–140[7]
ST(i) and ST(0)	ESC d P 0	1111 R R/M		95[8]			
FSQRT[i] = Square root	ESC 001	1111 1010		129–136			
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101		74–93			
FPREM = Partial remainder of ST(0) ÷ ST(1)	ESC 001	1111 1000		81–162			
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101		102–192			
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100		73–87			
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100		75–83			
FABS = Absolute value of ST(0)	ESC 001	1110 0001		29			
FCHS = Change sign of ST(0)	ESC 001	1110 0000		31–37			

Shaded areas indicate instructions not available in M8087/M80287.

NOTES:
2. Add 3 clocks to the range when d = 1.
3. Add 1 clock to **each** range when R = 1.
4. Add 3 clocks to the range when d = 0.
5. Typical = 48 (When d = 0, 42–50, typical = 45).
6. Add 1 clock to the range when R = 1.
7. 135–141 when R = 1.
8. Add 3 clocks to the range when d = 1.
9. $-0 \leq \text{ST}(0) \leq +\infty$.

M80C287 Extension to the CPU's 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		130–779[10]
FPTAN[11] = Partial tangent of ST(0)	ESC 001	1111 0010		198–504j
FPATAN = Partial arctangent	ESC 001	1111 0011		321–494
FSIN = Sine of ST(0)	ESC 001	1111 1110		129–778[10]
FSINCOS = Sine and cosine of ST(0)	ESC 001	1111 1011		201–816[10]
F2XM1[12] = $2^{ST(0)} - 1$	ESC 001	1111 0000		215–483
FYL2X[13] = ST(1) * $\log_2$(ST(0))	ESC 001	1111 0001		127–545
FYL2XP1[14] = ST(1) * $\log_2$(ST(0) + 1.0)	ESC 001	1111 1001		264–554
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		25
FSETPM = Set protected mode	ESC 011	1110 0100		12
FRSTPM = Reset protected mode	ESC 011	1111 0100		12
FSTSW AX = Store status word	ESC 111	1110 0000		18
FLDCW = Load control word	ESC 001	MOD 101 R/M	SIB/DISP	33
FSTCW = Store control word	ESC 101	MOD 111 R/M	SIB/DISP	18
FSTSW = Store status word	ESC 101	MOD 111 R/M	SIB/DISP	18
FCLEX = Clear exceptions	ESC 011	1110 0010		8
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP	192–193
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP	85
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP	521–522
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP	396
FINCSTP = Increment stack pointer	ESC 001	1111 0111		28
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		29
FFREE = Free ST(12)	ESC 101	1100 0 ST(12)		25
FNOP = No operations	ESC 001	1101 0000		19

Shaded areas indicate instructions not available in M8087/M80287.

NOTES:

10. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 78 additional clocks may be needed to reduce the operand.
11. $0 \leq |ST(0)| < 2^{63}$.
12. $-1.0 \leq ST(0) \leq 1.0$.
13. $0 \leq ST(0) < \infty, -\infty < ST(1) < +\infty$.
14. $0 \leq |ST(0)| < (2 - SQRT(2))/2, -\infty < ST(1) < +\infty$.

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 FEN/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.