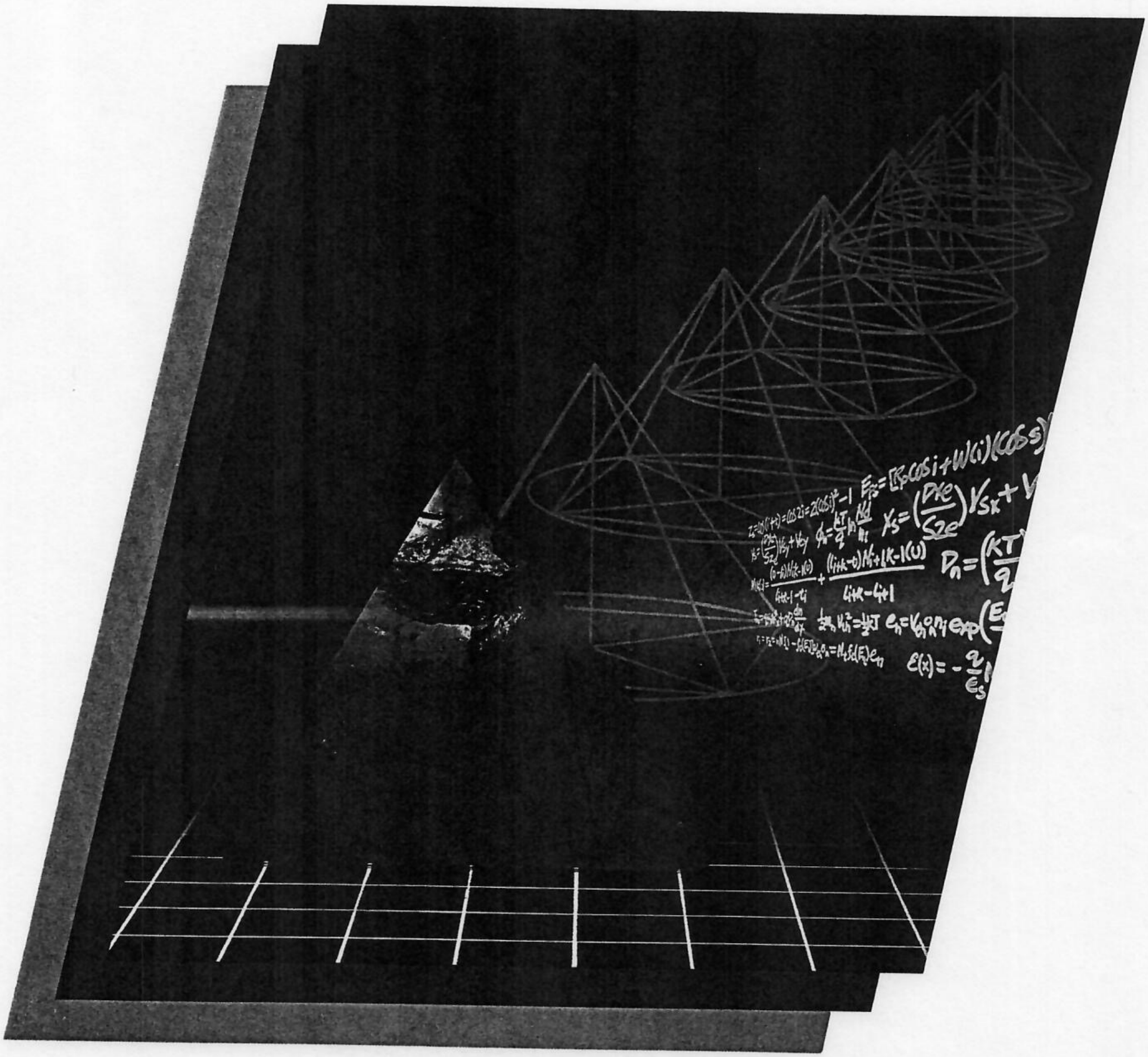
IIT 3C87
Advanced Math CoProcessor
Data Book



INTEGRATED
INFORMATION
TECHNOLOGY • INC

INTEGRATED INFORMATION
TECHNOLOGY • INC.

3C87DX ADVANCED MATH COPROCESSOR

Data Book

October 1991
Order No. 03-01-1002-002-0001

Copyright Notice

Disclaimer

Integrated Information Technology, Inc., makes no representations or warrantees regarding the content of this technical specification. The information is subject to change at any time without notice. Integrated Information Technology, Inc., assumes no responsibility for any errors contained herein.

Trademarks

4x4 and the IIT logo are registered trademarks of Integrated Information Technology, Inc. All other trademarks are registered to their respective companies.

TABLE OF CONTENTS

Figures

Tables

INTRODUCTION

FEATURES

- Pin compatible with the 80387DX
- Runs all 80387 software
- Uses the same programing tools as the Intel 80287 and 80387
- Implements ANSI/IEEE standard for binary floating point arithmetic
- Requires fewer coprocessor clock cycles for instruction execution than the 80387DX
- Encoded 4x4 matrix/vector transform
- Twenty-four registers (three banks of eight stack registers) available to the user
- High-performance 80-Bit architecture
- Low power CMOS technology provides 25% less power dissipation than the 80387DX
- Full-range transcendental operations for sine, cosine, tangent, arctangent, exponent, and logarithm
- Available at 16, 20, 25, 33 and 40 Mhz

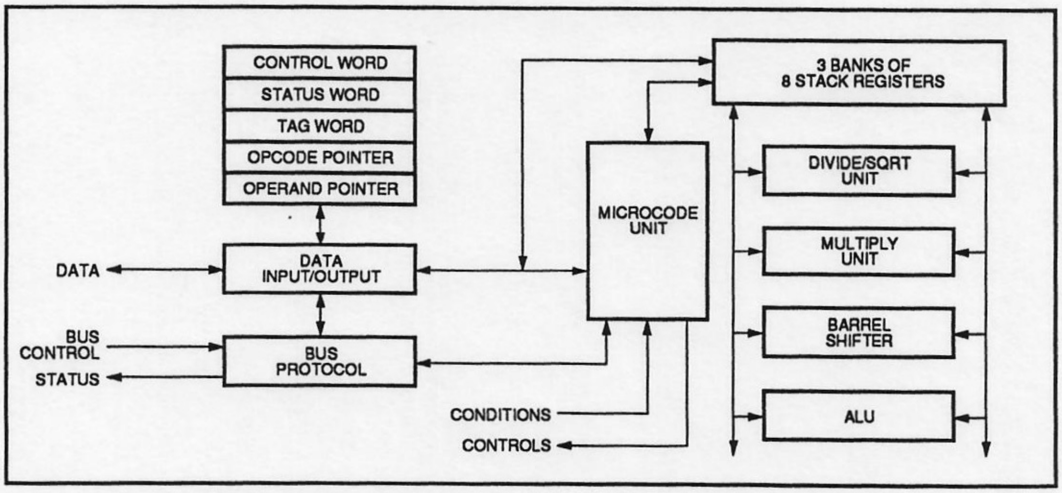


Figure 1. 3C87DX Block Diagram

OVERVIEW

IIT's 3C87DX Advanced Math CoProcessor is a high-performance floating point coprocessor that is pin compatible with the Intel 80387DX.

The device's unique architecture requires fewer clock cycles for most numeric operations than the equivalent operation in the 80387DX.

Table 1 compares the number of clock cycles required to perform typical floating point instructions.

The 3C87DX includes embedded microcode capable of performing a 4x4 matrix transformation as a single instruction.

IIT's unique 4x4™ feature can speed up 3D graphics modeling by as much as three times. The 3C87DX can execute transcendental functions such as sine, cosine, tangent, exponent and log functions. These functions, along with the added data types, registers, instructions and interrupts, facilitate high-speed numerical processing.

Table 1. Comparison of Clock Cycle Requirements

Instruction	Number of Clock Cycles Required		Speed Improvement
	IIT 3C87DX	Intel 80387DX	
ADD	12	23	1.9x
MPY	17	28	1.6x
DIV	46	88	1.9x
SQRT	48	119	2.5x
REM	38	75	2.0x
TAN	297	384	1.3x

GENERAL INFORMATION

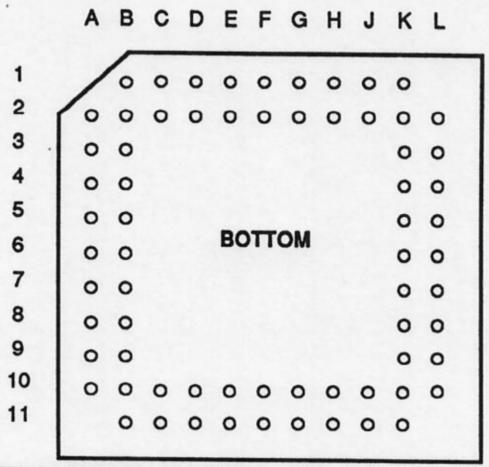
PIN DESCRIPTION

Figure 2 shows the location of all pins in the PGA. Table 2 on p.8 identifies all pins by name, and briefly describes their functions and characteristics. Outputs are initially tri-state and leave the floating condition when status enable (STEN) is asserted.

Data pins D_{31}-D_0 are bi-directional and tri-state. They leave the floating state when the 3C87DX is selected by asserting STEN, NPS1#, and NPS2 during read cycles.

In reset, the following pins are set up as shown below:

- High: READYO#, BUSY#
- Low: PEREQ, ERROR#
- Tri-state Off: D_{31}-D_0



A2	D_9	B1	D_8	C1	D_7	E1	V_{CC}	G1	D_3	J1	V_{SS}	K1	PEREQ	L2	ERROR#
A3	D_{11}	B2	V_{SS}	C2	D_6	E2	V_{SS}	G2	D_2	J2	V_{CC}	K2	BUSY#	L3	READYO#
A4	D_{12}	B3	D_{10}	C10	D_{23}	E10	D_{26}	G10	D_{28}	J10	V_{SS}	K3	Tie High	L4	STEN
A5	D_{14}	B4	V_{CC}	C11	V_{SS}	E11	D_{27}	G11	D_{29}	J11	CKM	K4	W/R#	L5	V_{SS}
A6	V_{CC}	B5	D_{13}									K5	V_{CC}	L6	NPS1#
A7	D_{16}	B6	D_{15}	D1	D_5	F1	V_{CC}	H1	D_1			K6	NPS2	L7	V_{CC}
A8	D_{18}	B7	V_{SS}	D2	D_4	F2	V_{SS}	H2	D_0			K7	ADS#	L8	CMD0#
A9	V_{CC}	B8	D_{17}	D10	D_{24}	F10	V_{CC}	H10	D_{30}			K8	READY#	L9	Tie High
A10	D_{21}	B9	D_{19}	D11	D_{25}	F11	V_{SS}	H11	D_{31}			K9	N.C.	L10	RESETIN
		B10	D_{20}									K10	CPUCLK2		
		B11	D_{22}									K11	NUMCLK2		

Figure 2. Pin-Out Information

Table 2. Pin Description

Pin Name*	Pin #	Input/ Output	Description
RESETIN	L10	I	The raising of System Reset (RESETIN)will cause the 3C87DX to stop processing and go dormant. Once raised, RESETIN needs to stay high for at least 40 CPUCLK2 time periods. When RESETIN goes low, the transition must be clocked with CPUCLK2. After going low, 50 CPUCLK2 time periods must pass before the 3C87DX can accept an instruction. RESETIN should be tied to CPU RESET.
CPUCLK2	K10	I	CPU Clock 2 (CPUCLK2) is used to clock the data interface, control unit, and the floating point unit of the 3C87DX. The rising edge of this signal is also used to reference other signals in the 3C87DX.
NUMCLK2	K11	NC*	Internally not connected to chip.
CKM	J11	NC*	Internally not connected to chip.
PEREQ	K1	O	Processor Extension Request (PEREQ) is used to indicate to the CPU that the 3C87DX is ready for a data transfer. It is lowered when NPU is not yet ready for data transfer, or when there is no more data to be transferred. It will go inactive before BUSY# goes inactive, and should be tied to the CPU's PEREQ input. Please refer to Figure 16 on p.37 for PEREQ's timing relationships to the BUSY# and READY# signals.
BUSY#	K2	O	Busy Status (BUSY#), if asserted, signals that the 3C87DX is executing an instruction. It is clocked with CPUCLK2 and should be tied to the CPU BUSY# pin. Please see Figure 16 on p.37 for its timing relationships to the PEREQ and READY# signals.
ERROR#	L2	O	The Error Status (ERROR#) signal is clocked to CPUCLK2. ERROR# indicates that an unmasked exception has occurred during the execution of the current instruction being processed. After reset, it is used to signal the CPU that a math coprocessor is in the system. ERROR# can only be deactivated by FINIT, FCLEX, FSTENV, and FSAVE.

*Note: # indicates an active low signal; NC means "not connected".

Table 2. Pin Description (Continued)

Pin Name	Pin #	Input/ Output	Description
READYO#	L3	O	The Ready Output (READYO#) signal indicates the termination of a coprocessor BUS cycle. In systems where no extra wait states are needed, the READYO# signal can be tied to the READY# input of the CPU. It is active only when the bus cycle selects the 3C87DX, and it is clocked to CPUCLK2.
D_{31}-D_0	A2-A5, A7, A8, A10, B1, B3, B5, B6, B8-B11, C1, C2, C10, D1, D2, D10, D11, E10, E11, G1, G2, G10, G11, H1, H2, H10, H11	I/O	The Data I/O (D_{31}-D_0) bi-directional pins transfer opcodes and data to/from the 3C87DX. These pins are active high and are clocked to CPUCLK2.
W/R#	K4	I	The Write/Read Bus Cycle (W/R#) signal is used to indicate whether the current bus cycle is a write cycle (signal high) or a read cycle (signal low). This pin should be connected to the corresponding signal on the CPU. It is ignored if any of signals STEN, NPS1#, or NPS2 are de-asserted, and is clocked to CPUCLK2.
ADS#	K7	I	Address Strobe (ADS#) and READY# indicate when the bus control logic may sample W/R # and the chip select signals (NPS1#, NPS2, CMD0#). ADS# is clocked to CPUCLK2 and should be tied to CPU ADS# for proper operation.
READY#	K8	I	Bus Ready Input (READY#) is used to trace bus activity by the bus control logic. Until terminated by READY#, bus cycles can be indefinitely extended. The READY# pins of the CPU and the 3C87DX should be tied together and clocked to CPUCLK2 for proper functionality.
STEN	L4	I	Status Enable (STEN) is the signal that selects the 3C87DX chip. When low, the signals BUSY#, PEREQ, ERROR#, and READYO# all go into a floating state. D_{31}-D_0 require STEN and other conditions to be present before these signals will leave the floating state. The STEN pin is usually connected to Vcc through a resistor. If required, the pin can be pulled down during testing. If STEN is used to select the chip, its state change should occur during the same clock time as NPS1#, NPS2, and CMD0#.

Table 2. Pin Description (Continued)

Pin Name	Pin #	Input/ Output	Description
NPS1#	L6	I	The Chip Select 1 (NPS1#) signal is clocked to CPUCLK2. When asserted with STEN and NPS2 during the first period of a CPU bus cycle, it indicates that the bus cycle is being used to communicate with the 3C87DX. This pin should be strapped to the CPU M/IO# signal so that the 3C87DX is selected only during CPU I/O cycles.
NPS2	K6	I	During the first period of a bus cycle, the Chip Select 2 (NPS2) signal indicates that the cycle's purpose is to communicate with the 3C87DX. This pin should be connected to the CPU's A_{31} pin for proper functionality. This signal is clocked to CPUCLK2.
CMD0#	L8	I	During a write cycle, the Command (CMD0#) signal indicates if an opcode (CMD0# asserted) or data (CMD0# de-asserted) is being sent to the 3C87DX. When asserted in a read cycle, it indicates that the status or control register is being read. When de-asserted, it indicates that the data register is being read. CMD0# is clocked to CPUCLK2 and should be tied to the CPU's A_2 signal for correct operation.
V_{CC}	A6, A9, B4, E1, F1 F10, J2, K5, L7	I	5V supply
V_{SS}	B2, B7, C11, E2 F2, F11, J1, J10, L5	I	Ground

SYSTEM ARCHITECTURE

3C87DX COMPONENTS

The 3C87DX CoProcessor can be divided into three sections:

- bus control logic (BCL)
- control and data interface units
- floating point unit (FPU)

The 3C87DX is a peripheral to the CPU and is accessed through the use of certain reserved I/O addresses. The BCL unit handles the bus cycle tracking and CPU interface. The BCL unit always operates synchronously with the CPU. Since the BCL communicates only with the CPU, all data to and from the coprocessor must pass through the CPU.

The control and data interface units manage the flow of data to and from the FPU. These units synchronize the coprocessor with the CPU. Incoming data is latched and either sent to the internal data FIFOs or to the instruction decoder. The instruction decoder generates sequences of microinstructions that control the 3C87DX.

The FPU is responsible for executing all numeric instructions. It performs all arithmetic, transcendental, and data transfer instructions using the register stack.

3C87DX ADDRESSING AND SYNCHRONIZATION

The 3C87DX is addressed using inputs NPS1#, NPS2, and STEN. If those inputs are all asserted, the bus cycle is intended for the coprocessor. In order for the coprocessor to respond to an I/O cycle, bit A_{31} must be set. Programs running in the CPU cannot directly access the 3C87DX without using an escape instruction.

Synchronization is performed through signals BUSY#, PEREQ, and ERROR#. BUSY# is used to synchronize instruction transfer. After receiving an escape instruction, the 3C87DX raises BUSY#. (See Figure 3, p.12.) The CPU will wait for BUSY# to go low before it sends a new opcode.

The 3C87DX uses PEREQ to signal the CPU that it is ready to send or receive data. Since the 3C87DX is a memory mapped device, it operates under the CPU's addressing modes. For instructions involving data operands, the CPU uses its PEREQ input to identify when the 3C87DX needs an operand.

After all operands have been transferred, the CPU may resume execution of the integer program operation while the 3C87DX simultaneously executes the numeric operation. Here the FWAIT instruction may be used to postpone "dependent" integer execution until after the coprocessor has returned the needed value.

When the 3C87DX detects an exception, it asserts the ERROR# signal, which causes a CPU interrupt.

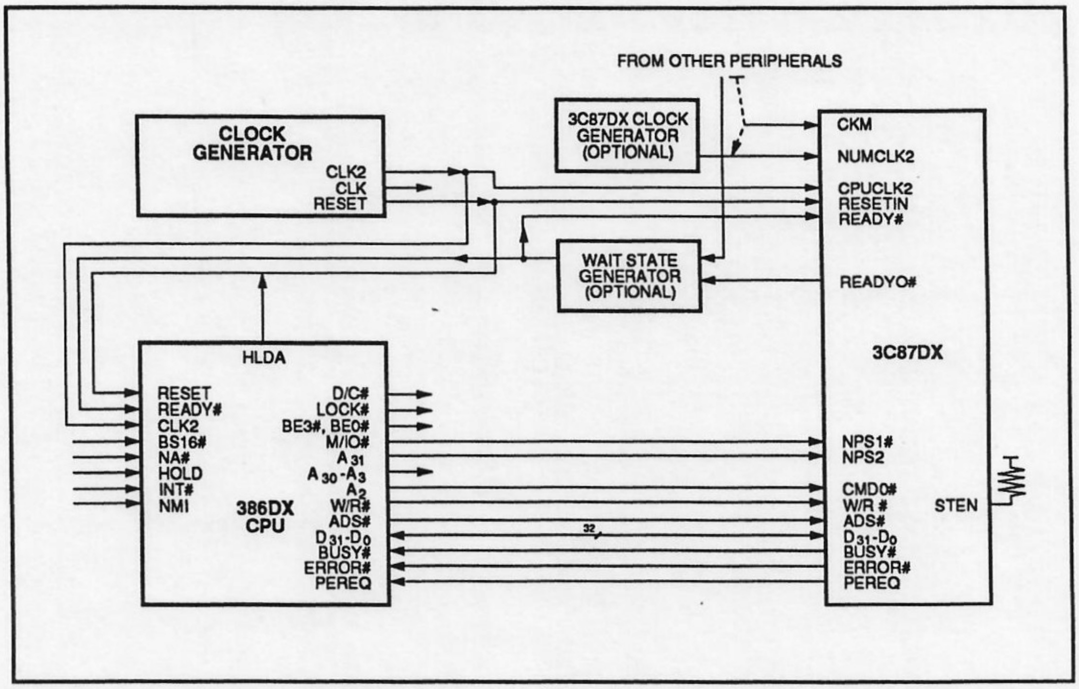


Figure 3. System Configuration

BUS OPERATION

The 3C87DX supports pipelined and non-pipelined bus cycles. A non-pipelined cycle occurs when the CPU asserts ADS# and no other coprocessor bus cycles are in progress. A pipelined bus cycle occurs when the CPU simultaneously asserts ADS# and provides valid next address and control signals during the READY# asserted phase of the current bus cycle.

The difference between pipelined and non-pipelined cycles is the way in which the 3C87DX changes from one state to another. During a pipelined bus cycle, the 3C87DX may be sent into a different state: T_p.

This occurs when both ADS# and READY are asserted in a bus cycle's T_{rs} state. The 3C87DX remains in the T_p state for only one cycle before returning to the T_{rs} state. (See Figure 17, p.38.)

Bus activity during consecutive non-pipelined bus cycles is shown in Figure 18, p.38.

CPU bus cycles are tracked with the ADS# and the READY# signals. At the beginning of a coprocessor bus cycle, the ADS# is asserted along with the 3C87DX select signals. The end of any bus cycle is signaled by a READY# assertion.

BUS CYCLE FUNCTION SELECT

The two types of coprocessor bus cycles are:

- Writing data or opcodes
- Reading control/status registers or data

The bus cycle type is identified by the CMD0# and W/R# input signals.

During the second CLK of a write cycle, the 3C87DX goes into the T_{rs} or READY# sensitive state. The 3C87DX samples READY# and stays in T_{rs} as long as READY# is de-asserted. The 3C87DX asserts READYO# for one CLK period beginning with the second CLK. (See cycle 1 in Figure 17, p.38.) When READY# is asserted, the 3C87DX will return to the idle state where ADS# may again be asserted by the CPU to begin a new cycle.

A read cycle begins the same way a write cycle does. However, at the rising edge of CLK during the first T_{rs} state, the 3C87DX drives the data outputs D_{31}-D_0 for both T_{rs} times. (See cycle 2 in Figure 17, p.38.) The 3C87DX automatically introduces a wait state during NPU reads. This allows extra time for the 3C87DX's data outputs to settle and results in greater system reliability. During the second T_{rs} cycle, the coprocessor asserts the READYO# signal to indicate the end of the read cycle. This forces the system back to the idle state and tri-states pins D_{31}-D_0. Now the CPU can proceed to a new bus cycle.

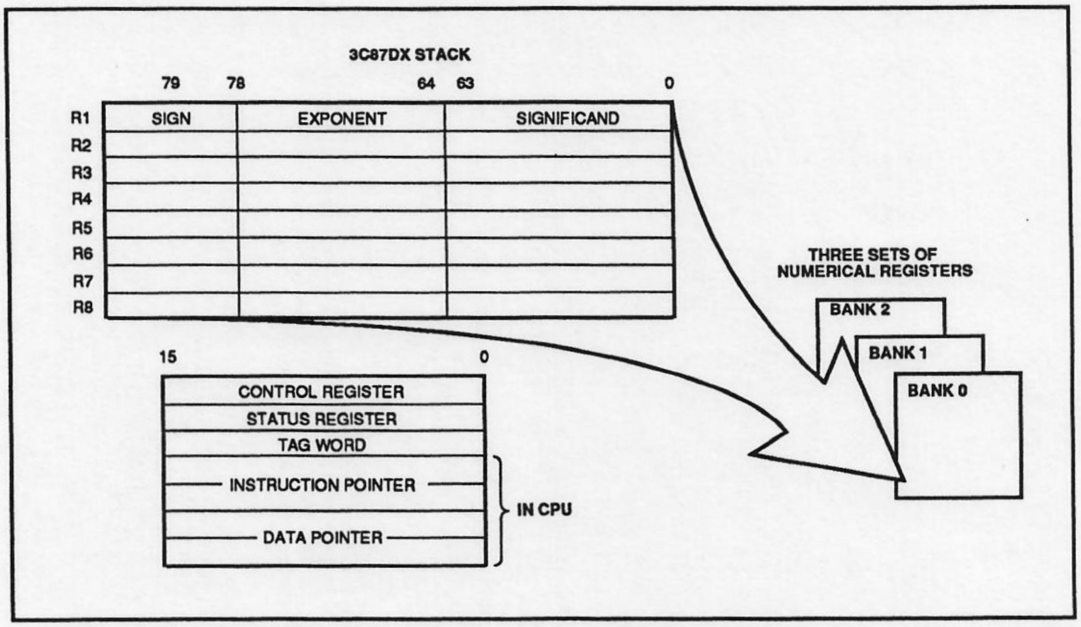


Figure 4. Register Set

REGISTER INFORMATION

REGISTER SET

The 3C87DX register set is shown in Figure 4 on p.14. These registers may be used by application programmers in addition to the registers in the CPU.

DATA REGISTERS

Application programs written for the Intel 80387DX use the first bank of eight 80-bit registers. The first bank of registers can be used as eight individual registers that are explicitly addressed, or as an eight register stack where operations are usually performed on the first item of the stack (top-of-stack).

The top-of-stack is identified by the TOP field of the status word. When floating point data is pushed onto the top-of-stack, TOP is decremented by one and the data is loaded into the newly identified register. When data is popped from the stack, the value identified by TOP is stored and TOP is incremented.

The 3C87DX stack is built downwards toward lower addressed registers. The data registers are addressed by instructions either implicitly or explicitly. The instructions that operate on the top-of-stack register are implicitly addressed by TOP. Instructions that explicitly address a register do so relative to the register address defined by TOP.

The programmer is also provided with an additional sixteen 80-bit registers arranged as two banks of eight stack registers that are not found on the 80387DX. These extra banks may be used for the 4x4 matrix transform or other data intensive math operations.

Table 3. Data Types

DATA FORMATS	RANGE	PRE-CISION	MOST SIGNIFICANT BYTE … HIGHEST ADDRESSED BYTE 7 0 | 7 0 | 7 0 | 7 0 | 7 0 | 7 0 | 7 0 | 7 0 | 7 0 | 7 0
WORD INTEGER	$\pm 10^4$	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	$\pm 10^{\pm 18}$	18 DIGITS	S | X | MAGNITUDE d_{17} d_{16} d_{15} d_{14} d_{13} d_{12} d_{11} d_{10} d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 79 72 … 0
SHORT REAL	$\pm 10^{\pm 38}$	24 BITS	S | BIASED EXPONENT | SIGNIFICAND 31 23 ← 1▲ 0
LONG REAL	$\pm 10^{\pm 308}$	53 BIT	S | BIASED EXPONENT | SIGNIFICAND 63 52 ← 1▲ 0
TEMPORARY REAL	$\pm 10^{\pm 4932}$	64 BITS	S | BIASED EXPONENT | 1 | SIGNIFICAND 79 64 63▲ 0

DATA TYPES

The 3C87DX supports seven data types, as shown in Table 3 on p.16. The least significant digits of the operands are stored in memory with the least significant bit in the lowest memory address.

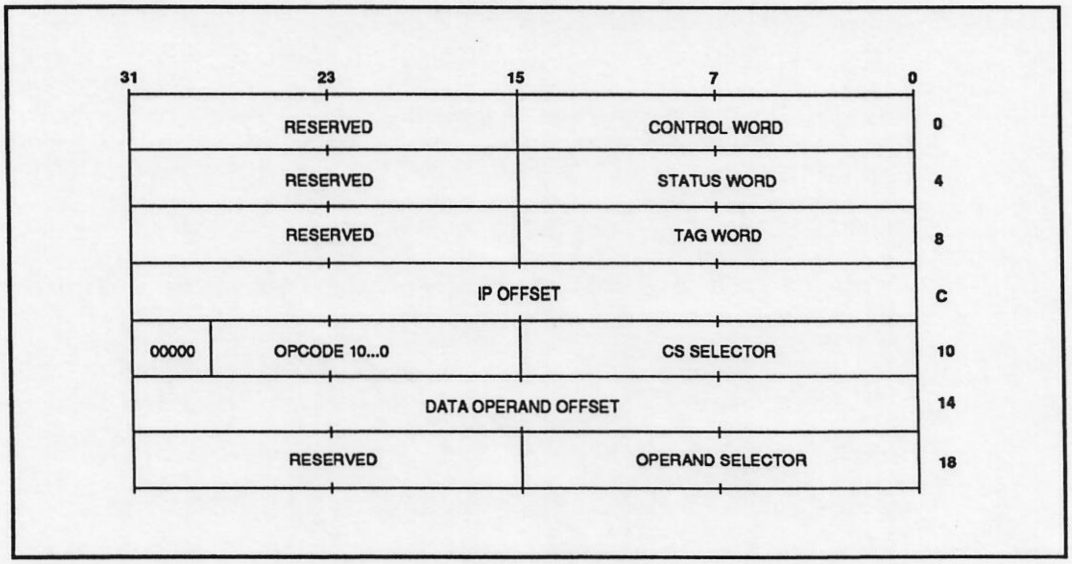
Internally the 3C87DX represents all numbers in extended-precision real format. All other formats are converted to extended-precision real format when they are accessed from memory. When operands are stored back into memory, an inverse type conversion is performed.

POINTER REGISTERS

During operation, certain combinations of opcode and data may "fail" and cause a numeric exception. In order to identify these failing numeric instructions, the 3C87DX accesses two pointer registers. The instruction-pointer register and data-pointer register are located in the 386DX CPU. These supply the addresses of any failed numeric instruction and its memory operand (if in memory). The data in the pointers are accessed through the use of the instructions FLDENV, FSTENV, FSAVE, and FRSTOR, and are provided for user-written error handling routines.

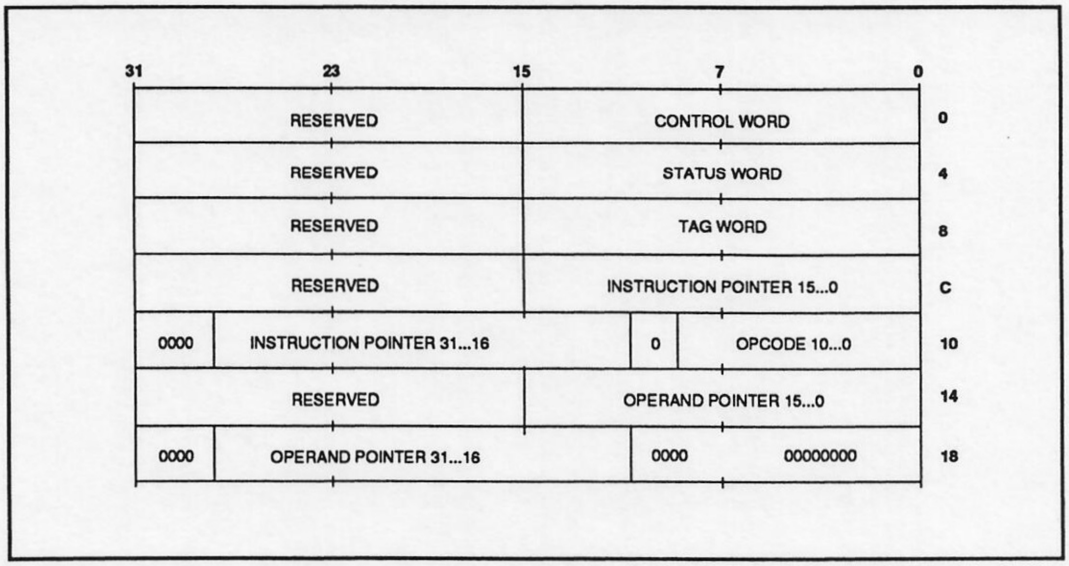
When the CPU decodes a new ESC instruction, its address, the opcode, and the operand's address are saved.

Depending upon the CPU's mode of operation (protected or real address) and the bit length of the operand, the instruction and data pointers are stored in 16- or 32-bit formats as shown in Figures 5 through 8 on pages 18 and 19.



RESERVED		CONTROL WORD		0
RESERVED		STATUS WORD		4
IP OFFSET				C
RESERVED		TAG WORD		8
00000	OPCODE 10...0	CS SELECTOR		10
DATA OPERAND OFFSET				14
RESERVED		OPERAND SELECTOR		18

Figure 5. 32-bit Protected Mode Format



RESERVED		CONTROL WORD		0
RESERVED		STATUS WORD		4
RESERVED		TAG WORD		8
RESERVED		INSTRUCTION POINTER 15...0		C
0000	INSTRUCTION POINTER 31...16	0	OPCODE 10...0	10
RESERVED		OPERAND POINTER 15...0		14
0000	OPERAND POINTER 31...16	0000	00000000	18

Figure 6. 32-bit Real Address Mode Format

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

Figure 7. 16-bit Protected Mode Format

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

Figure 8. 16-bit Real-Address Mode and Virtual-8086 Mode Format

SOFTWARE INSTRUCTIONS

DESCRIPTION

The 3C87DX Advanced Math CoProcessor supports the following enhanced mathematics instruction set. Most instructions are issued as a word.

Table 4. Instruction Set

Mnemonics [1]	Instructions	Lo Byte	Hi Byte [2, 3]	Typ. Clock Count
FADD ST, ST(i)	ST(i) <– ST(i) + ST(i)	D8	1 1 0 0 0 ST(i)	12
FADD ST(i), ST	ST(i) <– ST(i) + ST(0)	DC	1 1 0 0 0 ST(i)	12
FADDP ST(i), ST	ST(i) <– ST(i) + ST(0), Pop	DE	1 1 0 0 0 ST(i)	12
FIADD mem 2i	ST(0) <– ST(0) + mem 2i	DE	MOD 0 0 0 R/M | SIB/DISP	25
FIADD mem 4i	ST(0) <– ST(0) + mem 4i	DA	MOD 0 0 0 R/M | SIB/DISP	26
FADD mem 4r	ST(0) <– ST(0) + mem 4r	D8	MOD 0 0 0 R/M | SIB/DISP	21
FADD mem 8r	ST(0) <– ST(0) + mem 8r	DC	MOD 0 0 0 R/M | SIB/DISP	30
FSUB ST, ST(i)	ST(0) <– ST(0) - ST(i)	D8	1 1 1 0 0 ST(i)	13
FSUB ST(i), ST	ST(i) <– ST(i) - ST(i)	DC	1 1 1 0 1 ST(i)	13
FSUBR ST, ST(i)	ST(0) <– ST(i) - ST(0)	D8	1 1 1 0 1 ST(i)	13
FSUBR ST(i), ST	ST(i) <– ST(i) - ST(0)	DC	1 1 1 0 0 ST(i)	13
FSUBP ST(i), ST	ST(i) <– ST(0) - ST(i), Pop	DE	1 1 1 0 1 ST(i)	13
FSUBRP ST(i), ST	ST(i) <– ST(i) - ST(0), Pop	DE	1 1 1 0 0 ST(i)	13
FSUB mem 4r	ST(0) <– ST(0) - mem 4r	D8	MOD 1 0 0 R/M | SIB/DISP	25
FSUB mem 8r	ST(0) <– ST(0) - mem 8r	DC	MOD 1 0 0 R/M | SIB/DISP	34
FSUBR mem 4r	ST(0) <– mem 4r - ST(0)	D8	MOD 1 0 1 R/M | SIB/DISP	25
FSUBR mem 8r	ST(0) <– mem 8r - ST(0)	DC	MOD 1 0 1 R/M | SIB/DISP	34
FISUB mem 2i	ST(0) <– ST(0) - mem 2i	DE	MOD 1 0 0 R/M | SIB/DISP	25
FISUB mem 4i	ST(0) <– ST(0) - mem 4i	DA	MOD 1 0 0 R/M | SIB/DISP	24
FISUBR mem 2i	ST(0) <– mem 2i - ST(0)	DE	MOD 1 0 1 R/M | SIB/DISP	25
FISUBR mem 4i	ST(0) <– mem 4i - ST(0)	DA	MOD 1 0 1 R/M | SIB/DISP	24

Notes:	1)	ST, ST(0)	-	TOP of Stack
		mem 2i	-	16 bit integer memory
		mem 4i	-	32 bit integer memory
		mem 8i	-	64 bit integer memory
		mem 4r	-	32 bit real memory
		mem 8r	-	64 bit real memory
		mem 10r	-	80 bit real memory
		mem 10d	-	80 bit BCD memory
	2)	MOD R/M	-	Address mode specifier, as defined in 80x86
		SIB	-	Scale-Index-Base specifier, as defined in 80x86
		DISP	-	Displacement specifier, as defined in 80x86
	3)	ST(i)	-	3-bit pointer to stack index
		000	-	TOP of Stack
		111	-	Last Stack Element

Table 4. Instruction Set (continued)

Mnemonics	Instructions	Lo Byte	Hi Byte	Typ. Clock Count
FMUL ST, ST(i)	ST(0) <– ST(0) x ST(i)	D8	1 1 0 0 1 ST(i)	17
FMUL ST(i), ST	ST(i) <– ST(i) x ST(0)	DC	1 1 0 0 1 ST(i)	17
FMULP ST(i), ST	ST(i) <– ST(i) x ST(0), Pop	DE	1 1 0 0 1 ST(i)	17
FIMUL mem 2i	ST(0) <– ST(0) x mem 2i	DE	MOD 0 0 1 R/M | SIB/DISP	28
FIMUL mem 4i	ST(0) <– ST(0) x mem 4i	DA	MOD 0 0 1 R/M | SIB/DISP	33
FMUL mem 4r	ST(i) <– ST(0) x mem 4r	D8	MOD 0 0 1 R/M | SIB/DISP	26
FMUL mem 8r	ST(i) <– ST(0) x mem 8r	DC	MOD 0 0 1 R/M | SIB/DISP	35
FDIV ST, ST(i)	ST(0) <– ST(0) + ST(i)	D8	1 1 1 1 0 ST(i)	46
FDIV ST(i), ST	ST(i) <– ST(0) + ST(i)	DC	1 1 1 1 1 ST(i)	46
FDIVR ST, ST(i)	ST(0) <– ST(i) + ST(0)	D8	1 1 1 1 1 ST(i)	46
FDIVR ST(i), ST	ST(i) <– ST(i) + ST(0)	DC	1 1 1 1 0 ST(i)	46
FDIVP ST(i), ST	ST(i) <– ST(0) + ST(i), Pop	DE	1 1 1 1 1 ST(i)	46
FDIVRP ST(i), ST	ST(i) <– ST(i) + ST(0), Pop	DE	1 1 1 1 0 ST(i)	46
FDIV mem 4r	ST(0)<–ST(0) + mem 4r	D8	MOD 1 1 0 R/M | SIB/DISP	55
FDIV mem 8r	ST(0)<–ST(0) + mem 8r	DC	MOD 1 1 0 R/M | SIB/DISP	64
FDIVR mem 4r	ST(0)<–mem 4r + ST(0)	D8	MOD 1 1 1 R/M | SIB/DISP	55
FDIVR mem 8r	ST(0)<–mem 8r + ST(0)	DC	MOD 1 1 1 R/M | SIB/DISP	64
FIDIV mem 2i	ST(0)<–ST(0)+mem 2i	DE	MOD 1 1 0 R/M | SIB/DISP	60
FIDIV mem 4i	ST(0)<–ST(0)+mem 4i	DA	MOD 1 1 0 R/M | SIB/DISP	61
FIDIVR mem 2i	ST(0)<–mem 2i+ST(0)	DE	MOD 1 1 1 R/M | SIB/DISP	60
FIDIVR mem 4i	ST(0)<–mem 4i+ST(0)	DA	MOD 1 1 1 R/M | SIB/DISP	61
FSQRT	ST(0) <– square root [ST(0)]	D9	FA	48
FSCALE	ST(0) <– ST(0)*$2^{ST(1)}$	D9	FD	40
FPREM	ST(0) <– Remainder [ST(0)/ST(1)]	D9	F8	16-60
FPREM1	ST(0) <– IEEE Remainder [ST(0)/ST(1)]	D9	F5	16-60
FRNDINT	ST(0) <– Integer [ST(0)]	D9	FC	14

Table 4. Instruction Set (continued)

Mnemonics	Instructions	Lo Byte	Hi Byte	Typ. Clock Count
FXTRACT	push, ST(0) <– Fraction (old ST) ST(1)<–Exponent (old ST)	D9	F4	39
FABS	ST(0) <– |ST(0)|	D9	E 1	10
FCHS	ST(0) <– -ST(0)	D9	E 0	10
FXCH ST(i)	ST(i) <—> ST(0) Exchange	D9	1 1 0 0 1 ST(i)	16
FLD ST(i)	push, ST(0) <– ST(i)	D9	1 1 0 0 0 ST(i)	15
FILD mem 2i	push, ST(0) <– mem 2i	DF	MOD 0 0 0 R/M | SIB/DISP	21
FILD mem 4i	push, ST(0) <– mem 4i	DB	MOD 0 0 0 R/M | SIB/DISP	22
FILD mem 8i	push, ST(0) <– mem 8i	DF	MOD 1 0 1 R/M | SIB/DISP	30
FLD mem 4r	push, ST(0) <– mem 4r	D9	MOD 0 0 0 R/M | SIB/DISP	20
FLD mem 8r	push, ST(0) <– mem 8r	DD	MOD 0 0 0 R/M | SIB/DISP	29
FLD mem 10r	push, ST(0) <– mem 10r	DB	MOD 1 0 1 R/M | SIB/DISP	38
FLD mem 10d	push, ST(0) <– mem 10d	DF	MOD 1 0 0 R/M | SIB/DISP	187
FST ST(i)	ST(i) <– ST(0)	DD	1 1 0 1 0 ST(i)	12
FIST mem 2i	mem 2i <– ST(0)	DF	MOD 0 1 0 R/M | SIB/DISP	34
FIST mem 4i	mem 4i <– ST(0)	DB	MOD 0 1 0 R/M | SIB/DISP	36
FST mem 4r	mem 4r <– ST(0)	D9	MOD 0 1 0 R/M | SIB/DISP	32
FST mem 8r	mem 8r <– ST(0)	DD	MOD 0 1 0 R/M | SIB/DISP	39
FSTP ST(i)	ST(i) <– ST(0), Pop	DD	1 1 0 1 1 ST(i)	12
FISTP mem 2i	mem 2i <– ST(0), Pop	DF	MOD 0 1 1 R/M | SIB/DISP	34
FISTP mem 4i	mem 4i <– ST(0), Pop	DB	MOD 0 1 1 R/M | SIB/DISP	37
FISTP mem 8i	mem 8i <– ST(0), Pop	DF	MOD 1 1 1 R/M | SIB/DISP	45
FSTP mem 4r	mem 4r <– ST(0), Pop	D9	MOD 0 1 1 R/M | SIB/DISP	32
FSTP mem 8r	mem 8r <– ST(0), Pop	DD	MOD 0 1 1 R/M | SIB/DISP	39
FSTP mem 10r	mem 10r <– ST(0), Pop	DB	MOD 1 0 1 R/M | SIB/DISP	45
FBSTP mem 10d	mem 10d <– ST(0), Pop	DF	MOD 1 1 0 R/M | SIB/DISP	220

Table 4. Instruction Set (continued)

Mnemonics	Instructions	Lo Byte	Hi Byte	Typ. Clock Count
FCOM ST (i)	Compare ST(0), ST(i)	D8	1 1 0 1 0 ST(i)	16
FICOM mem 2i	Compare ST(0), mem 2i	DE	MOD 0 1 0 R/M | SIB/DISP	26
FICOM mem 4i	Compare ST(0), mem 4i	DA	MOD 0 1 0 R/M | SIB/DISP	27
FCOM mem 4r	Compare ST(0), mem 4r	D8	MOD 0 1 0 R/M | SIB/DISP	25
FCOM mem 8r	Compare ST(0), mem 8r	DC	MOD 0 1 0 R/M | SIB/DISP	31
FCOMP ST (i)	Compare & Pop ST(0), ST(i)	D8	1 1 0 1 1 ST(i)	17
FICOMP mem 2i	Compare & Pop ST(0), mem 2i	DE	MOD 0 1 1 R/M | SIB/DISP	26
FICOMP mem 4i	Compare & Pop ST(0), mem 4i	DA	MOD 0 1 1 R/M | SIB/DISP	27
FCOMP mem 4r	Compare & Pop ST(0), mem 4r	D8	MOD 0 1 1 R/M | SIB/DISP	25
FCOMP mem 8r	Compare & Pop ST(0), mem 8r	DC	MOD 0 1 1 R/M | SIB/DISP	31
FCOMPP	Compare, Pop twice ST(0), ST(1)	DE	D9	19
FUCOM ST(i)	IEEE Compare	DD	1 1 1 0 0 ST(i)	16
FUCOMP ST (i)	IEEE Compare, Pop	DD	1 1 1 0 1 ST(i)	16
FUCOMPP	IEEE Compare, Pop Twice	DA	E9	19
FTST	Compare, ST(0), 0	D9	E4	10
FXAM	Examine ST(0)	D9	E5	10
FLDZ	push, ST(0) <-- 0	D9	EE	15
FLD1	push, ST(0) <-- 1	D9	E8	15
FLDPi	push, ST(0) <-- π	D9	EB	15
FLDL2T	push, ST(0) <-- $\log_2$ (10)	D9	E9	15
FLDL2E	push, ST(0) <-- $\log_2$ (e)	D9	EA	15
FLDLG2	push, ST(0) <-- $\log_{10}$ (2)	D9	EC	15
FLDLN2	push, ST(0) <-- $\log_e$ (2)	D9	ED	15
FCOS	ST(0) <-- Cosine [ST(0)]	D9	FF	306

Table 4. Instruction Set (continued)

Mnemonics	Instructions	Lo Byte	Hi Byte	Typ. Clock Count
FSIN	ST(0) <– Sine [ST(0)]	D9	FE	306
FSINCOS	push, ST(0) <– Cosine (old ST) ST(i) <– Sine (old ST)	D9	FB	343
FPTAN	push, ST(0) <– 1 ST(i) <– tan (old ST)	D9	F2	297
FPATAN	ST(0) <– $\tan^{-1}$ ($\frac{ST\ (1)}{ST(0)}$), Pop	D9	F3	337
F2XM1	ST(0) <– $2^{ST\ (0)}$-1	D9	F0	245
FYL2X	ST(1) = ST(1) * $\log_2$ ST; and Pop ST	D9	F1	288
FYL2XP1	ST(1) = ST(1) * $\log_2$ [ST (0) + 1]; and Pop ST	D9	F9	267
FINIT	Initialize	DB	E3	10
FSTSW AX	store status word to AX	DF	E0	5
FSTSW	store status word	DD	MOD 1 1 1 R/M | SIB/DISP	5
FSTCW	store control word	D9	MOD 1 1 1 R/M | SIB/DISP	5
FLDCW	load control word	D9	MOD 1 0 1 R/M | SIB/DISP	12
FCLEX	clear exceptions	DB	E2	8
FSTENV	store environment	D9	MOD 1 1 0 R/M | SIB/DISP	45

Table 4. Instruction Set (continued)

Mnemonics	Instructions	Lo Byte	Hi Byte	Typ. Clock Count
FLDENV	load environment	D9	MOD 1 0 0 R/M | SIB/DISP	52
FSAVE	save 3C87 state	DD	MOD 1 1 0 R/M | SIB/DISP	360
FRSTOR	restore 3C87 state	DD	MOD 1 0 0 R/M | SIB/DISP	340
FINCSTP	increment stack pointer	D9	F7	7
FDECSTP	decrement stack pointer	D9	F6	7
FFREE ST(i)	empty ST(i)	DD	1 1 0 0 0 ST(i)	7
FNOP	no operation	D9	D0	7
FSBP0	Setbank pointer to bank 0 (During powerup, the bank pointer automatically points to bank 0.)	DB	E8	6
FSBP1	Setbank pointer to bank 1	DB	EB	6
FSBP2	Setbank pointer to bank 2	DB	EA	6
F4X4	4x4 Instruction	DB	F1	242

THE 4X4 INSTRUCTION

The 4x4 instruction is designed to solve the following computationally intensive problems in a single instruction.

$$X_n = X_0{*}A_{00} + Y_0{*}A_{01} + Z_0{*}A_{02} + W_0{*}A_{03}$$
$$Y_n = X_0{*}A_{10} + Y_0{*}A_{11} + Z_0{*}A_{12} + W_0{*}A_{13}$$
$$Z_n = X_0{*}A_{20} + Y_0{*}A_{21} + Z_0{*}A_{22} + W_0{*}A_{23}$$
$$W_n = X_0{*}A_{30} + Y_0{*}A_{31} + Z_0{*}A_{32} + W_0{*}A_{33}$$

Where:

A_{33}, A_{23}, A_{13}, A_{03}, A_{32}, A_{22}, A_{12}, and A_{02} are the coefficients in bank 1.

A_{31}, A_{21}, A_{11}, A_{01}, A_{30}, A_{20}, A_{10} and A_{00} are the coefficients in bank 2.

X_0, Y_0, Z_0, and W_0 are the input data in register bank 0 prior to execution of the 4x4 instruction.

X_n, Y_n, Z_n, and W_n are the output data in register bank 0 after execution of the 4x4 instruction.

SOFTWARE IDENTIFICATION OF A 3C87DX MATH COPROCESSOR

Software developers can use the assembly language shown in Figure 9 to identify the presence of a 3C87DX.

```
/* ASSEMBLY Listing to identify IIT Coprocessor*/
/****************************************************/
.MODEL SMALL
      fsb0    equu    <db 0DBh,  0E8h>
      fsb1    equ     <db 0DBh,  0EBh>
      fsb2    equ     <db 0DBh,  0EAh>
      f4x4    equ     <db 0DBh,  0F1h>

.code
mem10r        dt      3ffcffffffffff000000h
var1          dt      3fff8465245441234567h
var2          dt      44448044414c4c415300h
data1         dt      ?
res           dw      ?                       ; F4X4 result
```

Figure 9. Assembly Listing to Identify a 3C87DX

```
        PUBLIC  _IITCoPro
_IITCoPro       PROC
                finit                           ; First step checking
                fld1
                fld     tbyte ptr mem10r
                fyl2xp1
                fstp    tbyte ptr data1
                mov     ax, word ptr data1
                and     ax, 0ffh
                cmp     ax, 08fh
                je                              ; F4X4TEST
                cmp     ax, 090h
                mov     ax, 2                   ; Other Coprocessor
                jne     exit
                fninit
                fld     tbyte ptr var1          ; 3fff 8465 2454 4123 4567
                fld     tbyte ptr var2          ; 4444 8044 414c 4c41 5300
                fprem
                fwait
                fstp    tbyte ptr data1
                mov     ax, word ptr data1
                cmp     ax, 0f75ch
                mov     ax, 2                   ; Values returned in AX
                jne     exit                    ; 0 = Intel
                mov     ax, 0                   ; 1 = IIT
                jmp     exit                    ; 2 = Other Coprocessor
F4X4TEST:
                fnclex                          ; Double check IIT Coprocessor
                fninit                          ; Clear possible exceptions
                fwait
                fsb1                            ; Set bank pointer to bank1
                fld1                            ; Fill in the matrix
                fld1
                fld1
                fld1
                fld1
                fld1
                fld1
                fld1
                fwait
                fninit
                fwait
```

Figure 9. Assembly Listing to Identify a 3C87DX (Continued)

```
                fsb2                        ; Set bank pointer to bank2
                fld1
                fld1
                fld1
                fld1
                fld1
                fld1
                fld1
                fld1
                fwait
                fninit
                fwait
                fsb0
                fld1
                fld1
                fld1
                fld1

                fwait
                f4x4
                fwait
                fist    word ptr res
                fwait
                mov     ax, word ptr res
                cmp     ax, 04h
                mov     ax, 0               ; Other Coprocessor
                jne     exit
                mov     ax, 1               ; IIT Coprocessor
exit:
                fnclex
                fninit
                ret
_IITCoPro       ENDP

        PUBLIC _F4X4
_F4X4           PROC
$lsf    STRUC
        base    dw      ?
        regIP   dw      ?
        fptr1   dw      ?
        fptr2   dw      ?
        fptr3   dw      ?
```

Figure 9. Assembly Listing to Identify a 3C87DX (Continued)

```
$lsf   ends
lsf    equ    [bp-base]
       push   bp
       mov    bp,sp
       push   ds
       push   si
       mov    si,lsf.fptr2
       finit
       fwait
       fsb1
       fld    qword ptr [si+10h]
       fld    qword ptr [si+30h]
       fld    qword ptr [si+50h]
       fld    qword ptr [si+70h]
       fld    qword ptr [si+18h]
       fld    qword ptr [si+38h]
       fld    qword ptr [si+58h]
       fld    qword ptr [si+78h]
       fwait
       finit
       fwait
       fsb2
       fld    qword ptr [si+00h]
       fld    qword ptr [si+20h]
       fld    qword ptr [si+40h]
       fld    qword ptr [si+60h]
       fld    qword ptr [si+08h]
       fld    qword ptr [si+28h]
       fld    qword ptr [si+48h]
       fld    qword ptr [si+68h]
       fwait
       finit
       fwait
       fsb0
       mov    si,lsf.fptr1
       fld    qword prt [si+18h]
       fld    qword ptr [si+10h]
       fld    qword ptr [si+08h]
       fld    qword ptr [si+00h]
       fwait
```

Figure 9. Assembly Listing to Identify a 3C87DX (Continued)

```
        f4x4
        fwait
        mov     si,lsf.fptr1
        fstp    qword ptr [si+00h]
        fstp    qword ptr [si+08h]
        fstp    qword ptr [si+10h]
        fstp    qword ptr [si+18h]
        pop     si
        pop     ds
        pop     bp
        ret
_F4X4           ENDP

        /*********************************************************/
```

Figure 9. Assembly Listing to Identify a 3C87DX (Continued)

ELECTRICAL CHARACTERISTICS

Table 5. DC Electrical Characteristics

Parameter		Values* Min	Max	Notes
V_{IH}	High Level Input Voltage	2.0 V	V_{cc}+0.3 V	all inputs except CPUCLK2
V_{IL}	Low Level Input Voltage	-0.3 V	0.8 V	all inputs except CPUCLK2
V_{CH}	CPUCLK2 High Level Input	3.7 V	V_{cc}+0.3 V	•
V_{CL}	CPUCLK2 Low Level Input	-0.3 V	0.8 V	•
V_{OH}	High Level Output Voltage	2.4 V	•	I_{OH}=1 mA
V_{OL}	Low Level Output Voltage	•	0.45 V	I_{OL}=4 mA
I_{LI}	Input Leakage Current	•	±15 µA	measured at operating voltage range on signal pins
I_{LO}	Output Leakage Current	•	±15 µA	
I_{CC}-16	Supply Current at 16 MHz	•	180 mA	typical = 120 mA
I_{CC}-20	Supply Current at 20 MHz	•	220 mA	typical = 150 mA
I_{CC}-25	Supply Current at 25 MHz	•	250 mA	typical = 170 mA
I_{CC}-33	Supply Current at 33 MHz	•	240 mA	typical = 160 mA
I_{CC}-40	Supply Current at 40 MHz	•	270 mA	typical = 190 mA
C_{IN}	Input Capacitance	•	10 pF	f_c = 1 MHz
C_O	Input/Output Capacitance	•	12 pF	f_c = 1 MHz
C_{CLK}	CLK Capacitance	•	20 pF	f_c = 1 MHz

*Note: All voltages are with respect to V_{SS} (ground).

Table 6. Operating Specifications

Parameter	Values Min	Max
Operating Temperature	0°C	70°C
Case Temperature Range	-65°C	+110°C
Storage Temperature Range	-65°C	+150°C
Supply Voltage V_{CC}	4.5 V	5.5 V
Voltage Range on any Pin	-0.5 V	V_{CC} +0.5 V
Power Dissipation		1.4 Watt

Table 7. AC Electrical Characteristics

Parameter	16 MHz		20 MHz		25 MHz		33 MHz		40 MHz		Voltage Thresh.
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
	(nanosec.)		(nanosec.)		(nanosec.)		(nanosec.)		(nanosec.)		
t_1 CPUCLK2 Period	31.25	125	25	125	20	125	15	125	12.5	125	2.0V
t_2 CPUCLK2 High Time	9		8		7		6.25		5.5		2.0V
t_3 CPUCLK2 High Time	5		5		4		4.5		4		3.0V
t_4 CPUCLK2 Low Time	9		8		7		6.25		5.5		2.0V
t_5 CPUCLK2 Low Time	7		6		5		4.5		4		0.8V
t_6 CPUCLK2 Rise Time		8		8		7		4		3	0.8V to 3.7V
t_7 CPUCLK2 Fall Time		8		8		7		4		3	3.7V to 0.8V
For output pins BUSY#, READY#, PEREQ, ERROR#:											C_L=75pF
t_8 Output Delay	3	35	2	32.	2	24		22		20	
t_9 Tri-state Delay	1	60	1	50	1	40	1	30	1	20	
For output pins ADS#, W/R#, STEN, CMD0#, READY#, NPS1#, NPS2:											
t_{10} Setup Time	25		20		14		12		11		
t_{11} Hold Time	12		12		11		11		10		
D_{31}-D_0 Data Pins:											
t_{12} Setup Time	13		13		13		13		11		
t_{13} Hold Time	11		11		10		10		9		C_L=120pF
t_{14} Output Delay	0	54	0	54	0	38	0	35	0	33	
t_{15} Tri-state Delay	5	33	5	27	5	24	4	20	4	18	
t_{16} RESETIN Setup Time	13		12		10		7		6		
t_{17} RESETIN Hold Time	4		4		3		3		2		

WAVEFORM DIAGRAMS

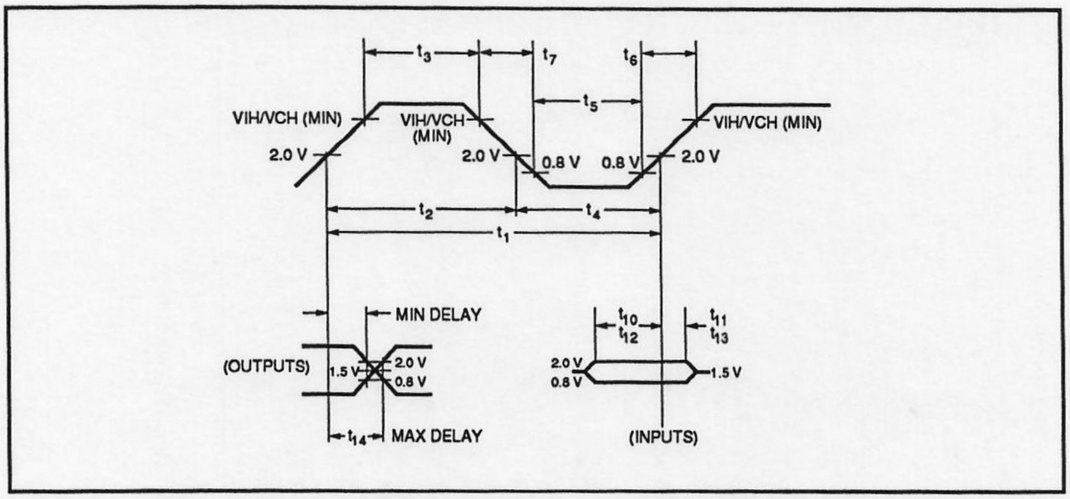


Figure 10. CPUCLK2/NUMCLK2 Waveform and Measurement Points for Input/Output

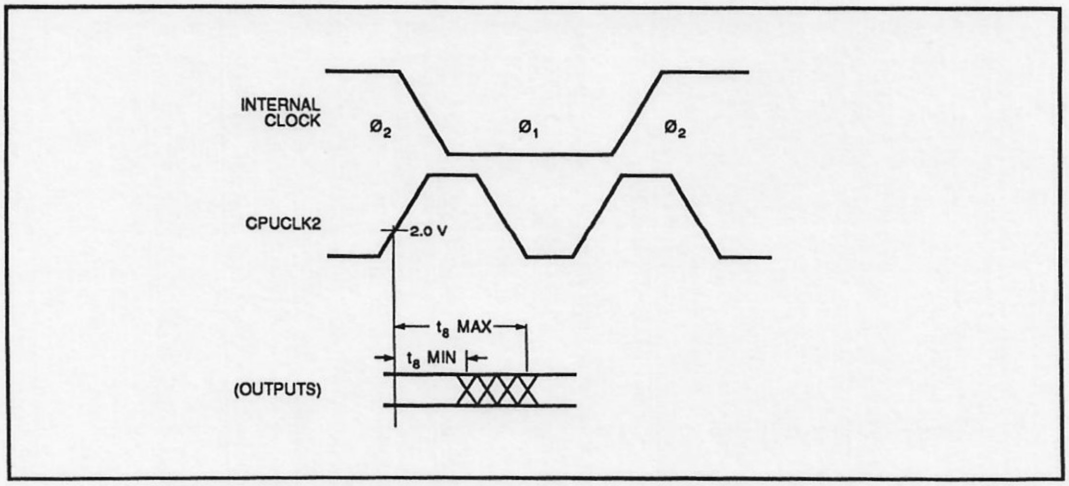


Figure 11. Output Signals

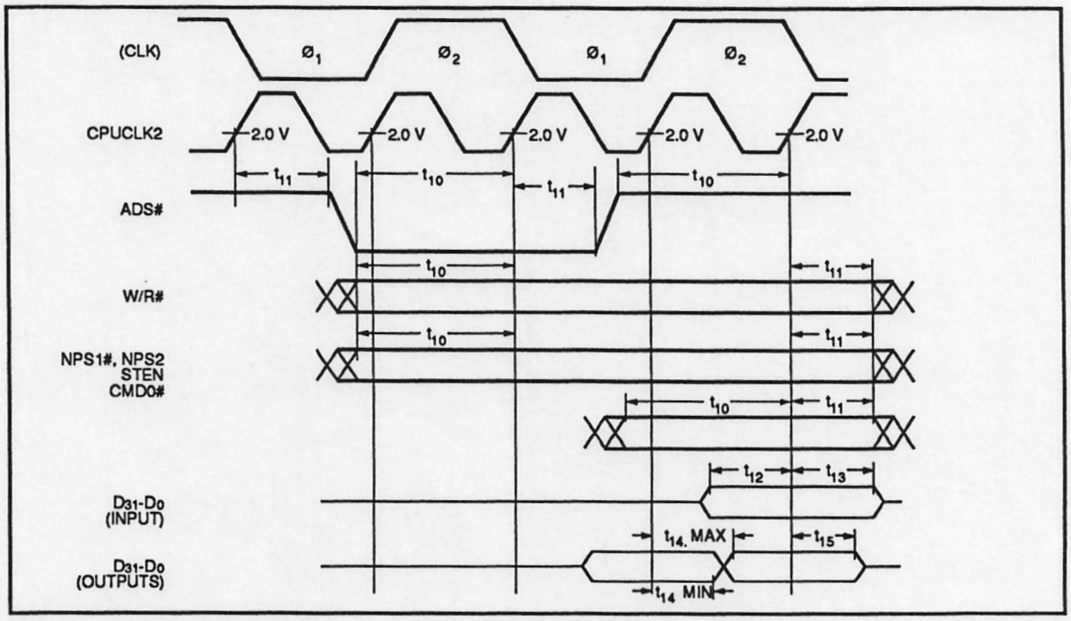
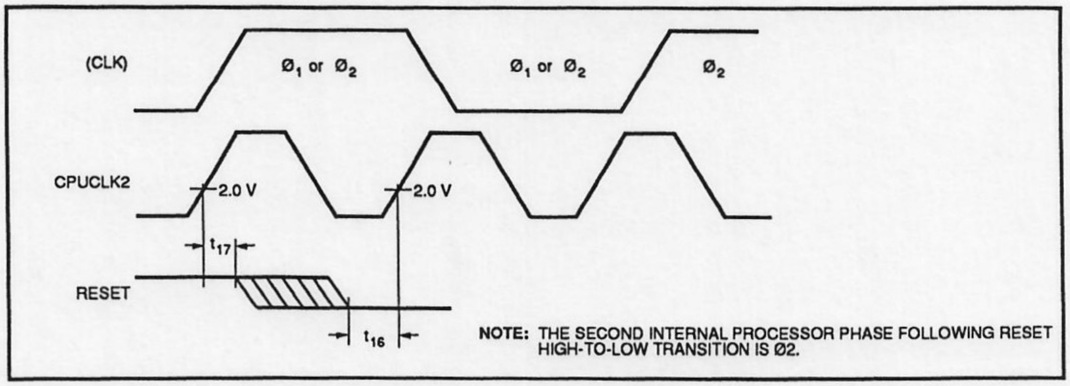


Figure 12. I/O Signals



NOTE: THE SECOND INTERNAL PROCESSOR PHASE FOLLOWING RESET HIGH-TO-LOW TRANSITION IS Ø2.

Figure 13. Reset Signal

Table 8. Signal Timing Relationships

Symbol	Parameter	Min	Max	Clocked To
		(cycles*)		
t_{18}	RESETIN High Duration	40		CPUCLK2
t_{19}	RESETIN Low Duration before 3C87DX instruction	50		CPUCLK2
t_{20}	BUSY # Active Duration	6		CPUCLK2
t_{21}	ERROR # active to BUSY # inactive	6		CPUCLK2
t_{22}	PEREQ inactive to ERROR # active	6		CPUCLK2
t_{23}	Consecutive opcode writes Consecutive opcode/operand writes	6		CPUCLK2
t_{24}	Consecutive operand writes	8		CPUCLK2
t_{25}	BUSY # active from READY # active	4	4	CPUCLK2

*Note: measured in CPUCLK2 cycles

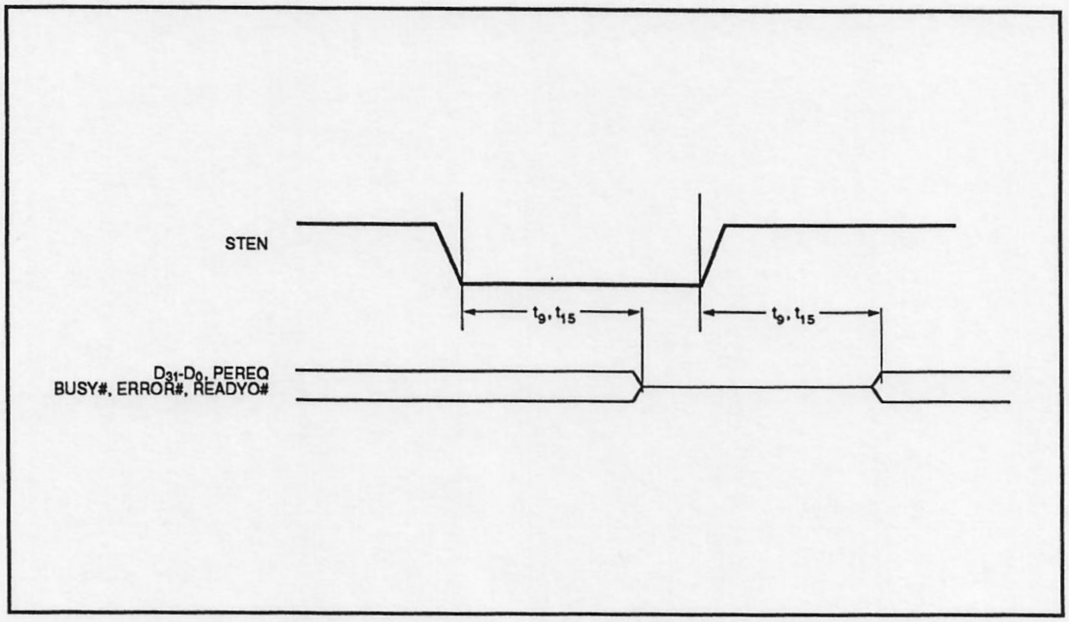
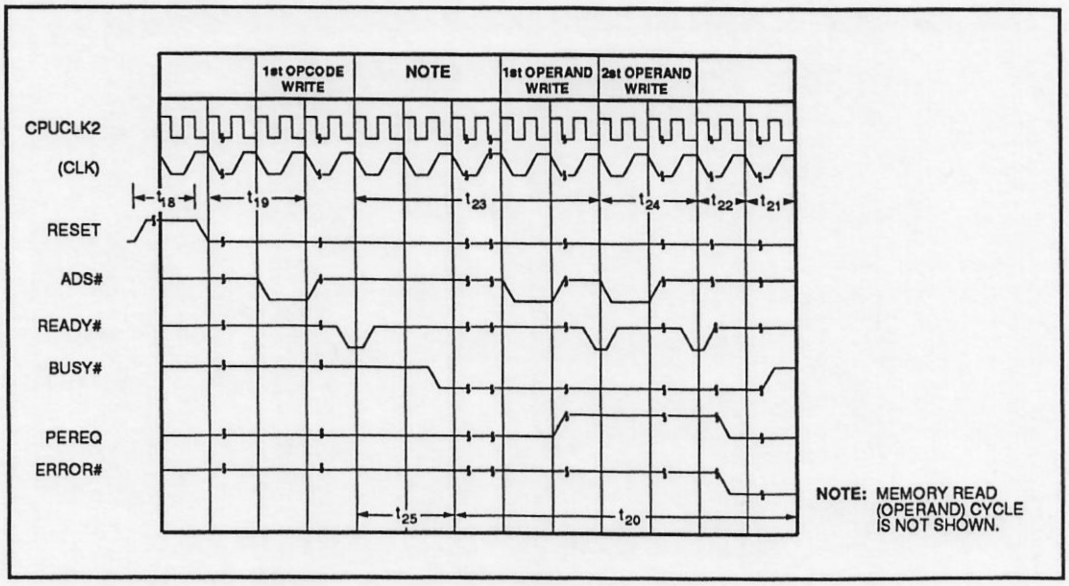
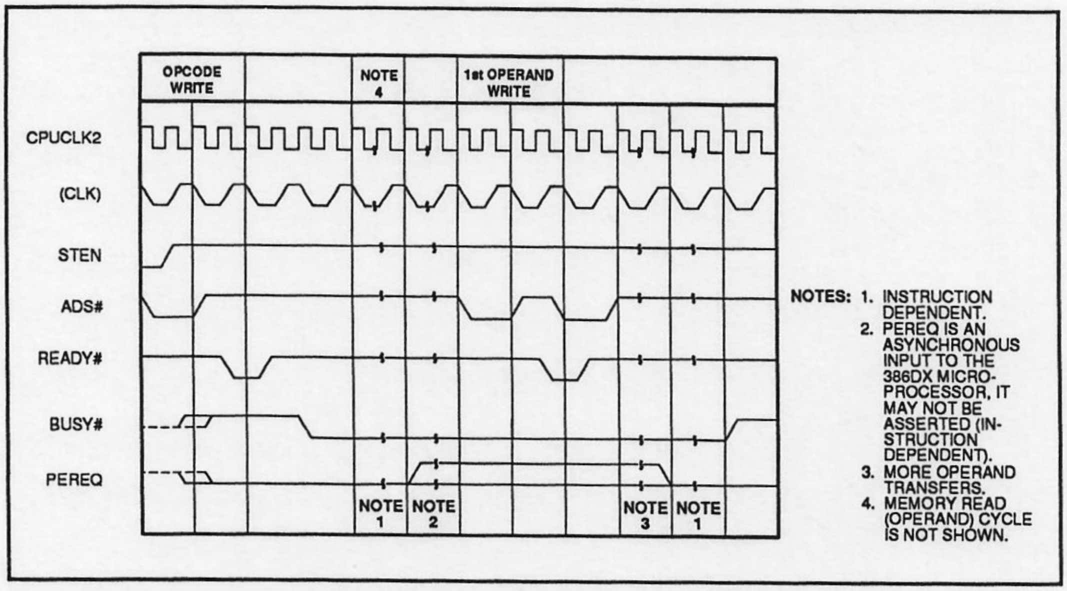


Figure 14. Float from STEN



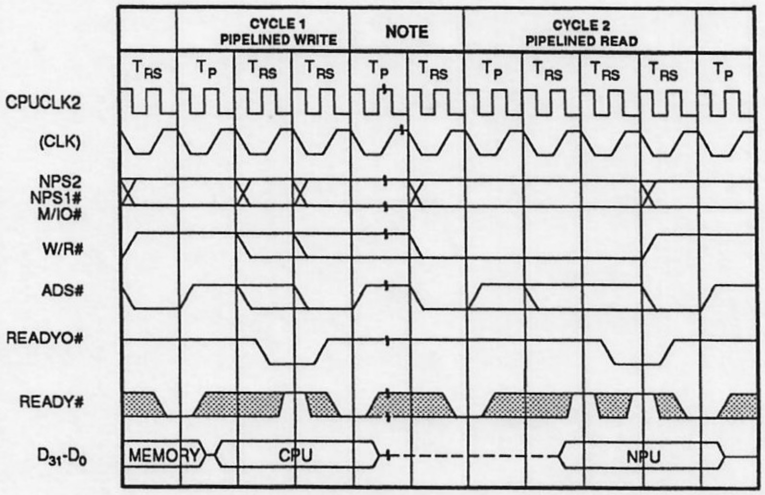
NOTE: MEMORY READ (OPERAND) CYCLE IS NOT SHOWN.

Figure 15. Other Parameters



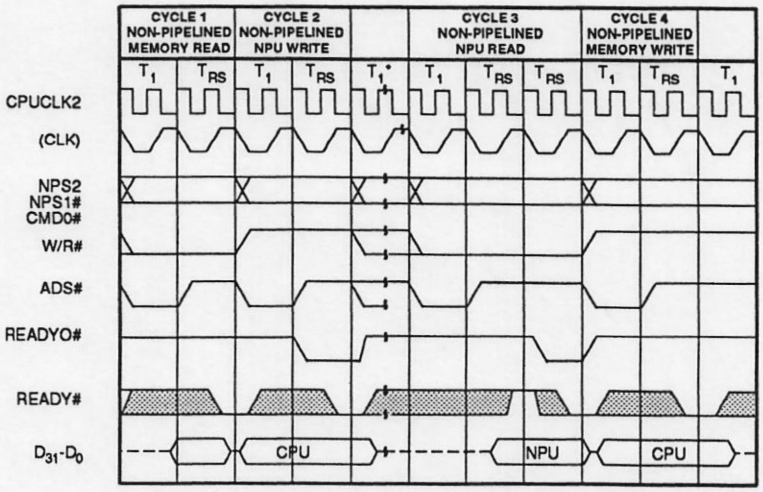
NOTES:
1. INSTRUCTION DEPENDENT.
2. PEREQ IS AN ASYNCHRONOUS INPUT TO THE 386DX MICROPROCESSOR, IT MAY NOT BE ASSERTED (INSTRUCTION DEPENDENT).
3. MORE OPERAND TRANSFERS.
4. MEMORY READ (OPERAND) CYCLE IS NOT SHOWN.

Figure 16. PEREQ and READY# Signal Relationships



NOTE: CYCLES BETWEEN OPERAND WRITE TO THE NPU AND STORING RESULT.

Figure 17. Example of Pipelined Cycles



NOTES: CYCLES 1 & 2 REPRESENT PART OF THE OPERAND TRANSFER CYCLE FOR INSTRUCTIONS INVOLVING EITHER 4-BYTE OR 8-BYTE OPERAND LOADS.

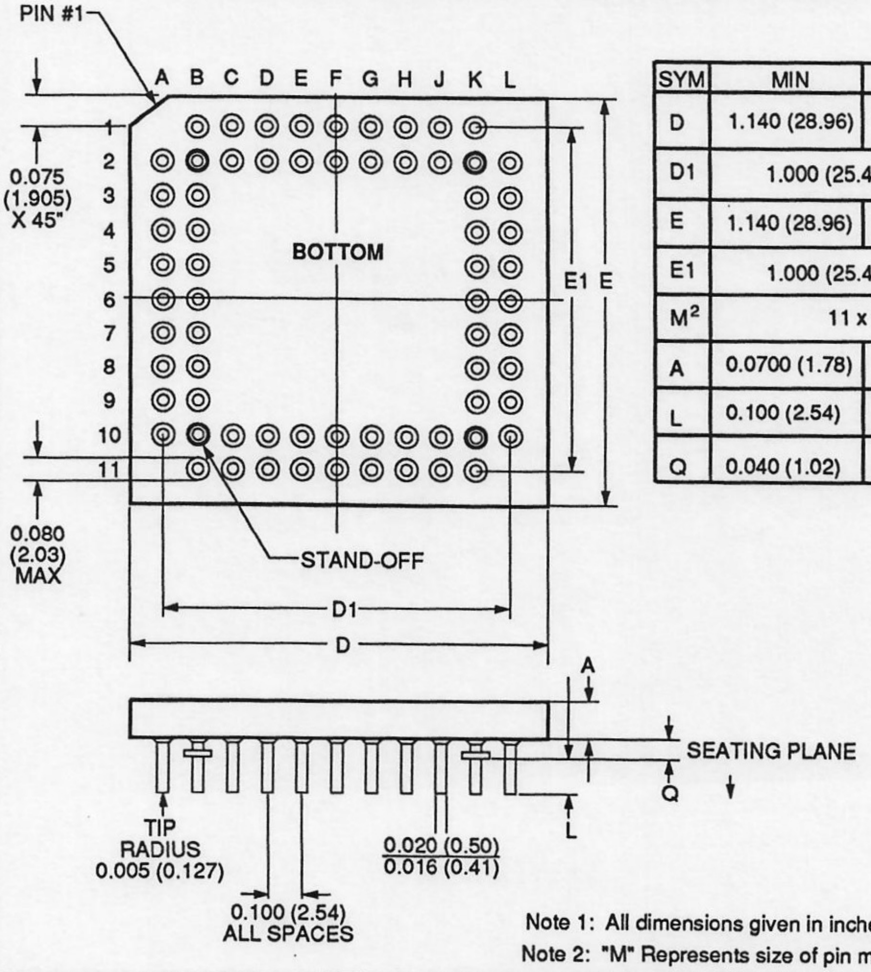
CYCLES 3 & 4 REPRESENT PART OF THE OPERAND TRANSFER CYCLE FOR A STORE OPERATION.

* CYCLES 1 & 2 COULD REPEAT HERE OR T_1 STATES FOR VARIOUS NON-OPERATED TRANSFER CYCLES AND OVERHEAD.

Figure 18. Non-Pipelined Read and Write Cycles

MECHANICAL DESCRIPTION

68 LEAD CPGA (CERAMIC PIN GRID ARRAY) CAVITY-UP PACKAGE [1]



SYM	MIN	MAX
D	1.140 (28.96)	1.180 (29.97)
D1	1.000 (25.40) BSC	
E	1.140 (28.96)	1.180 (29.97)
E1	1.000 (25.40) BSC	
M[2]	11 x 11	
A	0.0700 (1.78)	0.145 (3.68)
L	0.100 (2.54)	0.200 (5.08)
Q	0.040 (1.02)	0.060 (1.52)

Note 1: All dimensions given in inches (millimeters).

Note 2: "M" Represents size of pin matrix.

Figure 19. Packaging Dimensions

Integrated Information Technology Inc. (IIT)
2445 Mission College Blvd. Santa Clara, California 95054 Telephone (408) 727-1885 Fax (408) 980-0432