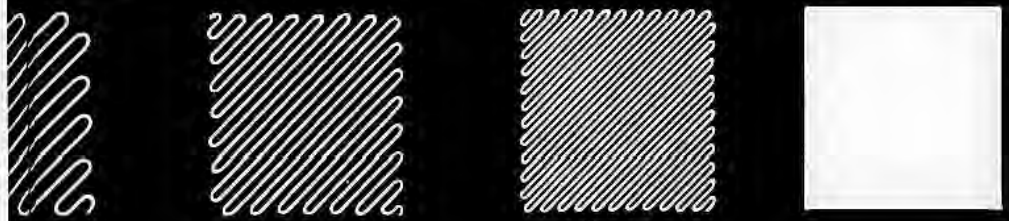
FasMath™

83S87 User's Manual

High Performance CMOS Math Processor



Cyrix™

Advancing the Standards

Cyrix

FasMath™ CX-83S87 User's Manual

January, 1990

Cyrix

P.O. Box 850118
Richardson, TX 75085-0118

IMPORTANT NOTICE

Cyrix Corporation (Cyrix) reserves the right to make changes in the devices or specifications contained herein without notice. Before design-in or order placement customers are advised to verify that the information on which orders or design activities are based is current.

Cyrix warrants its products to conform to current specifications in accordance with Cyrix' standard warranty. Testing is performed to the extent necessary as determined by Cyrix to support this warranty. Unless explicitly specified by customer order requirements not all device characteristics are necessarily tested.

Cyrix assumes no liability unless specifically agreed to in writing for customer's product design or infringement of patents or copyrights of third parties arising from use of Cyrix devices. No license, either express or implied, to Cyrix' patents, copyrights, or other intellectual property rights pertaining to any machine or combination of Cyrix' devices is hereby granted.

Cyrix products are not intended for use in any medical, life saving, or life sustaining systems.

Table of Contents

1. About This Manual

This document defines the electrical and functional characteristics of the Cyrix CX-83S87 device. The device is described in sufficient detail to provide both the system designer and the programmer the information needed to evaluate and use the device. Detailed descriptions of the internal architecture of the device are not contained in this document.

2. References

The following documents are frequently cited throughout the body of this specification. The reader should be familiar with their contents and have them available for reference when using this manual:

1. **"Microprocessors"**, 1990, Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

2. **"376 Embedded Processor Hardware Reference Manual"**, 1988, Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

3. **"80386 Hardware Reference Manual"**, 1987, Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

4. **"80387 Programmer's Reference Manual"**, 1987, Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

5. **"IEEE-754-1985 Standard for Binary Floating Point Arithmetic"**, 1985, Institute for Electrical and Electronic Engineers, Inc., 345 East 47th Street, New York, NY 10017.

3. General Introduction

3.1 Overview

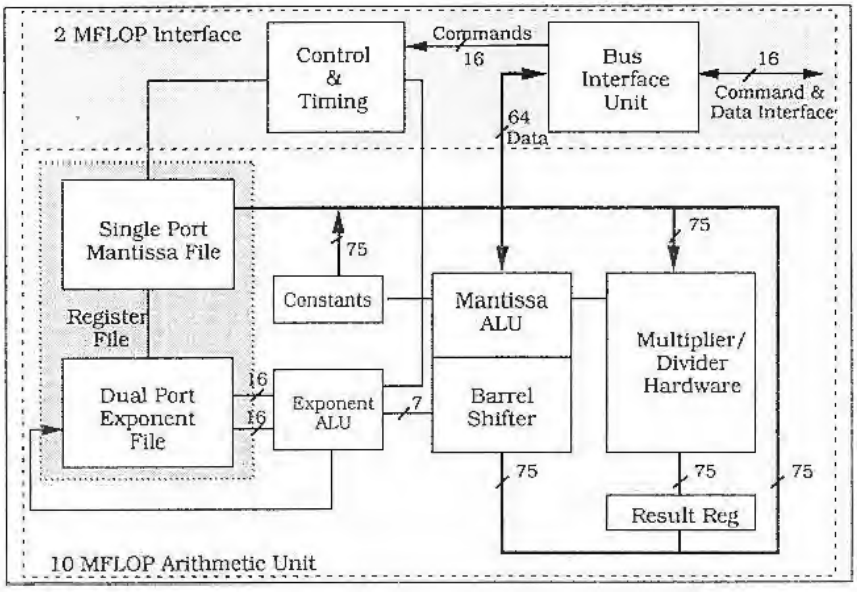
The Cyrix CX-83S87 is a CMOS VLSI integrated circuit which is socket compatible and software compatible with the Intel 387SX math coprocessor yet offers substantially improved performance. The CX-83S87 requires 4 to 10 times fewer clock cycles to execute floating point operations than the Intel 387SX. This was accomplished by implementing its floating point primitive operations in hardware rather than in a microprogrammed sequencer. This approach allows the CX-83S87 to perform simple floating point operations at the same speed as the 386SX processor can perform integer additions; square root, elementary functions. The transcendental functions are evaluated correspondingly faster.

3.2 Circuit

The CX-83S87 device is fabricated using a 1.0 micron, double layer metal CMOS process. This permits the CX-83S87 to operate at clock rates of 16 and 20 Mhz. State of the art ESD protection and latch-up prevention circuits are incorporated into the CX-83S87 design. The CX-83S87 is available in a 68-pin, ceramic j-lead package or PLCC package. Both packages are identical in size, and shape to the Intel 387SX 68 pin PLCC package.

3.3 Standards

The CX-83S87 implements a full extended double precision IEEE-754-1985 architecture using 80-bit internal format for data storage and computation. This format assures maximum accuracy and operand/result compatibility with the original 387SX.



FasMath™ 83S87 Block Diagram

3.4 Architecture

The CX-83S87 architecture is partitioned into three functional blocks: The Interface Unit, the Execution Unit, and the Control Unit. The Interface Unit manages the CX-83S87 interface to the host 386SX device. The Execution Unit performs all floating point primitive operations including operand type conversions, normalizations, additions, multiplications, divisions, and rounding of results. The Control Unit supervises the execution of primitives, sequences primitives to realize complex operations, and controls traffic to/from the Interface Unit.

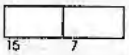
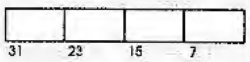
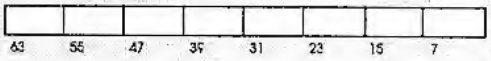
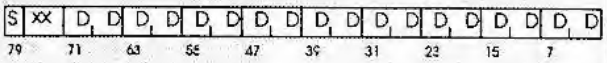
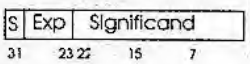
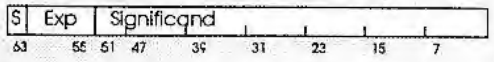
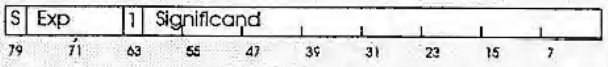
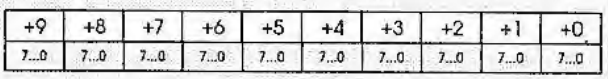
3.5 Programming

The CX-83S87 processor provides the user 8 data registers accessed in a stack-like manner, a control register, and a status register. The CX-83S87 also provides a data register tag word which improves context switching and stack performance by maintaining empty/non-empty status for each of the eight data registers. In addition, registers in the 386SX contain pointers to (a) the memory location containing the current CX-83S87 instruction word and (b) the memory location containing the operand associated with that instruction (if any).

The instructions used to program the CX-83S87 are binary and function compatible with those defined for the Intel 387SX Numeric Processor Extension (cf 2.1). Instructions are provided which load/store data and constants, perform arithmetic, elementary, and transcendental functions, manipulate fraction and exponent fields of operands, and control the status and operating mode of the CX-83S87.

3.6 Data Types

The Cyrix CX-83S87 supports IEEE-754-1985 32-bit single precision, 64-bit double precision, and 80-bit extended double precision real data formats. The CX-83S87 also supports 18 digit BCD integer data format and 16-bit, 32-bit, and 64-bit binary integer data formats. The CX-83S87 data formats are schematically depicted in the following figure.

Format	Size	Most Significant ... Least Significant
Word Integer	16 bits	
"Short" Integer	32 bits	
Long Integer	64 bits	
BCD Integer	80 bits	
Single Real	32 bits	
Double Real	64 bits	
Extended Real	80 bits	
Byte Displacement From Base Address		

Memory Data Formats

Internally, all data are converted to 80-bit extended precision format for storage and manipulation. The range and precision provided by 80-bit extended precision format allows the *exact* representation of 16, 32, & 64 bit binary integers, 18 digit BCD integers, and 32 & 64 bit real numbers. Note that the reverse is not necessarily true: the results of 80-bit calculations may require rounding to be represented in the other formats. The following figure shows the range and precision characteristics of each CX-83S87 data type.

Format	Size	Dynamic Range	Resolution
Word Integer	16 bits	-32768 +32767	1 in 2^{16}
"Short" Integer	32 bits	-2,147,483,648 +2,147,483,647	1 in 2^{32}
Long Integer	64 bits	-9,223,382,027,854,875,808 +9,223,382,027,854,875,807	1 in 2^{64}
BCD Integer	80 bits	-999,999,999,999,999,999 +999,999,999,999,999,999	1 in 10^{18}
Single Real	32 bits	$5.8775 \times 10^{-39} \leq \lvert N \rvert \leq 3.4028 \times 10^{+38}$	1 in 2^{24}
Double Real	64 bits	$1.1125 \times 10^{-308} \leq \lvert N \rvert \leq 1.7977 \times 10^{+308}$	1 in 2^{53}
Extended Real	80 bits	$1.68105 \times 10^{-4932} \leq \lvert N \rvert \leq 1.1897 \times 10^{+4932}$	1 in 2^{64}

CX-83S87 Data Type Numerical Properties

3.7 Computational Accuracy

The representation of real numbers in any number system of finite precision is inherently approximate. A simple fraction such as 1/3 cannot be precisely represented in a finite number of digits. This simple observation raises the possibility that different computing systems may choose different methods of approximation and produce different results given the same inputs and algorithms.

The IEEE-754-1985 Standard for Binary Floating Point Arithmetic attempts to solve this problem by specifying the error bounds for the calculation of binary floating point values. These error bounds are controlled in two ways: (1) by directly specifying the error allowable in a primitive calculation and (2) by specifying the exact rounding algorithms to be used. The result of this standardization is that given the same set of input values and rounding instructions, all conformal machines will produce the same result to within a defined margin of error.

Computations internal to the CX-83S87 are performed using 80-bit extended precision operands. The 15-bit exponent and the 64-bit significand data are operated on using independent 15-bit & 75-bit integer arithmetic units. The significand ALU includes provisions for handling extra Guard, Round, & Indicator (sticky) bits to assist in maintaining precision when rounding.

These extra bits are used in IEEE specified rounding modes and help maintain accuracy when the precision of a result exceeds that available in the significand. The programmer can select any of the four IEEE specified rounding modes for use in computation: round to nearest or even, round down, round up, and truncate.

In addition, for the sake of compatibility with earlier generations of floating point processors, a "precision control" function is provided to the programmer. This is used to specify that internal results be maintained in single, double, or extended precision range and resolution.

4. Instruction Set

4.1 Instruction Set Summary

The following table summarizes the operation and allowed forms of every member of the CX-83S87 instruction set:

Mnemonic	Result	Operation
F2XM1	TOS ←	$2^{TOS}-1$
FABS	TOS ←	|TOS|
FADD	ST(i) ←	ST(i)+TOS
FADD	TOS ←	TOS+ST(i)
FADD	TOS ←	TOS+M.DR
FADD	TOS ←	TOS+M.SR
FADDP	ST(i-1) ←	ST(i)+TOS; SP←SP+1
FIADD	TOS ←	TOS+M.SI
FIADD	TOS ←	TOS+M.WI
FCHS	TOS ←	-TOS
FCLEX	---	Clear Exceptions
FCOM	CC ←	TOS-ST(i)
FCOM	CC ←	TOS-M.DR
FCOM	CC ←	TOS-M.SR
FCOMP	CC ←	TOS-ST(i); SP←SP+1
FCOMP	CC ←	TOS-M.DR; SP←SP+1
FCOMP	CC ←	TOS-M.SR; SP←SP+1
FCOMPP	CC ←	TOS-ST(1); SP←SP+2
FICOM	CC ←	TOS-M.SI
FICOM	CC ←	TOS-M.WI
FICOMP	CC ←	TOS-M.SI; SP←SP+1
FICOMP	CC ←	TOS-M.WI; SP←SP+1
FCOS	TOS ←	COS(TOS)
FDECSTP	SP ←	SP-1
FDIV	ST(i) ←	ST(i)/TOS
FDIV	TOS ←	TOS/ST(i)
FDIV	TOS ←	TOS/M.DR
FDIV	TOS ←	TOS/M.SR
FDIVP	ST(i-1) ←	ST(i)/TOS; SP←SP+1
FDIVR	TOS ←	ST(i)/TOS
FDIVR	ST(i) ←	TOS/ST(i)
FDIVR	TOS ←	M.DR/TOS
FDIVR	TOS ←	M.SR/TOS
FDIVRP	ST(i-1) ←	TOS/ST(i); SP←SP+1
FIDIV	TOS ←	TOS/M.SI
FIDIV	TOS ←	TOS/M.WI
FIDIVR	TOS ←	M.SI/TOS
FIDIVR	TOS ←	M.WI/TOS
FFREE	TAG(i) ←	Empty
FINCSTP	SP ←	SP+1
FINIT	---	Initialize

FLD	TOS ← ST(i); SP←SP-1
FLD	TOS ← M.DR; SP←SP-1
FLD	TOS ← M.LI; SP←SP-1
FLD	TOS ← M.SR; SP←SP-1
FLD	TOS ← M.XR; SP←SP-1
FBLD	TOS ← M.BCD; SP←SP-1
FILD	TOS ← M.SI; SP←SP-1
FILD	TOS ← M.WI; SP←SP-1
FLD1	TOS ← 1.0; SP←SP-1
FLDCW	Ctl Word ← Memory
FLDENV	Env Regs ← Memory
FLDL2E	TOS ← $Log_2(e)$; SP←SP-1
FLDL2T	TOS ← $Log_2(10)$; SP←SP-1
FLDLG2	TOS ← $Log_{10}(2)$; SP←SP-1
FLDLN2	TOS ← $Log_e(2)$; SP←SP-1
FLDPI	TOS ← π; SP←SP-1
FLDZ	TOS ← 0.0; SP←SP-1
FMUL	ST(i) ← ST(i)*TOS
FMUL	TOS ← TOS*ST(i)
FMULP	ST(i-1) ← ST(i)*TOS; SP←SP+1
FMUL	TOS ← TOS*M.DR
FMUL	TOS ← TOS*M.SR
FIMUL	TOS ← TOS*M.SI
FIMUL	TOS ← TOS*M.WI
FNOP	--- No Operation
FPATAN	TOS ← ATAN($\frac{ST(1)}{TOS}$); SP←SP+1
FPREM	TOS ← Rem($\frac{TOS}{ST(1)}$)
FPREM1	TOS ← Rem($\frac{TOS}{ST(1)}$)
FPTAN	TOS;ST(1) ← 1; TAN(TOS); SP←SP-1
FRNDINT	TOS ← Round(TOS)
FRSTOR	--- Restore state.
FSAVE	--- Save state.
FSCALE	TOS ← $TOS*2^{(ST(1))}$
FSIN	TOS ← SIN(TOS)
FSINCOS	TOS;ST(1) ← COS;SIN(TOS); SP←SP+1
FSQRT	TOS ← $\sqrt{TOS}$
FST	ST(i) ← TOS
FST	M.DR ← TOS
FST	M.SR ← TOS
FSTP	ST(i-1) ← TOS; SP←SP+1
FSTP	M.DR ← TOS; SP←SP+1
FSTP	M.LI ← TOS; SP←SP+1
FSTP	M.SR ← TOS; SP←SP+1
FSTP	M.XR ← TOS; SP←SP+1
FBSTP	M.BCD ← TOS; SP←SP+1

FIST	M.SI ← TOS
FIST	M.WI ← TOS
FISTP	M.SI ← TOS; SP←SP+1
FISTP	M.WI ← TOS; SP←SP+1
FSTCW	Memory ← Control word.
FSTENV	Memory ← Ctl,Status,IP,DP.
FSTSW	Memory ← Status
FSTSWAX	AX ← Status
FSUB	ST(i) ← ST(i)-TOS
FSUB	TOS ← TOS-ST(i)
FSUBP	ST(i-1) ← ST(i)-TOS; SP←SP+1
FSUB	TOS ← TOS-M.DR
FSUB	TOS ← TOS-M.SR
FISUB	TOS ← TOS-M.WI
FISUB	TOS ← TOS-M.SI
FSUBR	TOS ← ST(i)-TOS
FSUBR	ST(i) ← TOS-ST(i)
FSUBRP	ST(i-1) ← TOS-ST(i); SP←SP+1
FSUBR	TOS ← M.DR-TOS
FSUBR	TOS ← M.SR-TOS
FISUBR	TOS ← M.WI-TOS
FISUBR	TOS ← M.SI-TOS
FTST	CC ← TOS-0.0
FUCOM	CC ← TOS-ST(i)
FUCOMP	CC ← TOS-ST(i); SP←SP+1
FUCOMPP	CC ← TOS-ST(1); SP←SP+2
FXAM	CC ← Class of TOS.
FXCH	TOS ↔ ST(i) Exchange
FXTRACT	TOS;ST(1) ← Signif; Exp; SP←SP-1
FYL2X	TOS ← ST(1)*Log_2(TOS); SP←SP+1
FYL2XP1	TOS ← ST(1)*Log_2(1+TOS); SP←SP+1

The abbreviations and conventions used in the CX-83S87 instruction summary are:

1. TOS == Top of stack register pointed to by SSS in Status Register. (cf. Sec 4.3).

2. ST(1)== Next to Top of stack register (pointed to by SSS+1).

3. Memory operands are referred to as "M.XX" where "XX"=:
   "WI" --> 16-bit integer;
   "SI" --> 32-bit integer;
   "LI" --> 64-bit integer;
   "SR" --> 32-bit real;
   "DR" --> 64-bit real;
   "XR" --> 80-bit real;
   "BCD" --> 18-digit BCD integer.

4. The "Operation" column refers to stack layout *before* instruction execution.

5. The "Result" column refers to stack layout *after* instruction execution.

6. Operands separated by a semicolon indicate that the leftmost destination receives the leftmost source, e.g., TOS<--COS(TOS) in FSINCOS; ST(1)<--SIN(TOS).

7. SP== CX-83S87 Stack Pointer (Contents of SSS).

8. IP== CX-83S87 Instruction Pointer.

9. DP== CX-83S87 Data Pointer.

10. ST(i)== Randomly addressed CX-83S87 data register i.

11. CC== Condition codes in CX-83S87 Status Register. (cf. Sec 4.3)

12. Env Regs==Status, Mode Control, and Tag Registers, IP & DP.

4.2 Data Registers

The Cyrix CX-83S87 provides a set of eight 80-bit data registers for use in computations. These registers are accessed in a stack-like fashion for programming purposes. An explicit data load instruction (FLD) must be used to store a datum into the CX-83S87 "Top of Stack" prior to performing arithmetic operations on it. The load instruction causes a "push" operation by decrementing the "stack pointer" (SSS field of the Status Register) by one modulo 8 and loading the datum into the physical register newly addressed by SSS.

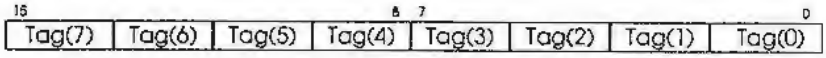
The CX-83S87 provides a complete set of dual operand instructions to ease programming of its pseudo-stack register set. The instructions FADD, FCOM, FDIV, FMUL, and FSUB perform arithmetic operations on the Top of Stack and either memory or explicitly addressed register operands. The result is placed in the Top of Stack. Note that conversion of the memory operand to 80-bit internal format is performed automatically.

All other CX-83S87 arithmetic instructions operate only on data contained in CX-83S87 registers.

The explicit register addressing feature operates by specifying the displacement (i) of the target register from the current Top of Stack. This displacement is added to the SSS field modulo 8 (prior to any increment or decrement operations specified by the instruction) to calculate the physical register number. Syntactically, ST(i) is used to specify displacement (i) from the Top of Stack. Thus, ST(0) is Top of Stack, ST(1) is next to Top of Stack, and so on. Logical register references of the form ST(i) are always relative to the current Top of Stack.

The limited number of physical registers and the use of "wrapping" modulo 8 arithmetic to address them leads to the possibility of a data register overwrite error. To prevent this and to simplify error detection, the CX-83S87 maintains a register Tag Word comprised of two bits for each physical data register. Tag Word fields assume one of four values depending on the contents of their associated data registers: Valid("00"), Zero("01"), Special("10"), and Empty("11"). Note: Denormal, Infinity, QNaN, SNaN and unsupported formats will be tagged as "Special". Tag values are maintained transparently by the CX-83S87 and are only available to the programmer indirectly through the FSTENV and

FSAVE instructions. The Tag Word with tag fields for each associated physical register, tag(R), is schematically depicted below:



TAG VALUES:

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

The CX-83S87 detects two kinds of register operation errors: Source Register Empty and Destination Register Full . The Source Register Empty error occurs when an instruction attempts to read a source operand from a data register which is Empty, i.e. Tag(R)="11". The Data Register Full error occurs when an attempt is made to store data into a register which is not Empty, i.e. Tag(R)≠"11". In the context of load ("push") and store ("pop") operations these errors are interpreted as stack wraparound/overflow (Dest. Full) and "stack underflow" (Source Empty) conditions. Note that in other contexts, such as "FST ST(4),ST(0)" with ST(4) not empty, or "FLD ST(0),ST(6)" with ST(6) empty, these errors also occur. Register errors generate the Invalid Exception and are reported through the SF bit of the Status Register.

The results of CX-83S87 operations are converted to the desired data format and stored in memory using the register store instruction, FST. The store instruction always uses the Top of Stack as its source operand. All forms of the FST instruction allow a "pop" of the Top of Stack upon completion.

"Pop" operations are effected by marking the physical register addressed by the SSS field of the Status Register as Empty and incrementing the SSS field by 1 modulo 8. "Pop" operations are provided as options for the dual operand instructions FADD, FDIV, FMUL, FSUB, and FUCOM *when using a data register destination operand.* In addition, FCOM provides a "pop" option when used with either memory or a data register as a source operand to facilitate conditional testing. Finally, both FCOM and FUCOM provide a double "pop" form which compares the Top of Stack to the next to Top of Stack and "pops" both upon completion.

For special programming situations, the FFREE instruction can be used to mark an explicitly addressed register as Empty. FINCSTP & FDECSTP can also be used to increment or decrement the SSS field of the Status Register.

4.3 Control & Status Registers

The Cyrix CX-83S87 communicates information about its status and the results of operations to the host 386SX via the Status Register. The CX-83S87 Status Register is comprised of bit fields that reflect exception status, operation execution status, register status, operand class, and comparison results. This register is continuously accessible to the 386SX CPU regardless of the state of the Control or Execution Units.

The CX-83S87 Mode Control Register (MCR) is used by the 386SX µP to specify the operating mode of the CX-83S87 processor. The MCR contains bit fields which specify the rounding mode to be used, the precision with which to calculate results, and the exception conditions which should be reported to the host via traps. The user controls precision, rounding, and exception reporting by setting or clearing appropriate bits in the MCR.

The following figure details the CX-83S87 Status Register and Mode Control Register as they appear to the programmer:

Register	15 12	11 8	7 4	3 0
Status	B C3 S S	S C2 C1 C0	ES SF P U	O Z D I
Mode Ctl	* * * *	RC RC PC PC	* * P U	O Z D I
				Exceptions/Masks

Table 3: CX-83S87 Status Register & Mode Control Register

The bit fields used in Table 3 are defined as follows:

B -- Copy of ES bit.

C3,C2,C1,C0 -- Condition code bits.

SSS -- Top of Stack Register number.

ES -- Error Indicator. Set to 1 if an unmasked exception is detected.

SF -- Stack Fault or invalid register operation bit.

RC -- Rounding Control bits:
00 - Round to nearest or even.
01 - Round toward minus infinity.
10 - Round toward plus infinity.
11 - Truncate.

PC -- Precision Control bits:
00 - 24-bit mantissa.
01 - Reserved.
10 - 53-bit mantissa.
11 - 64-bit mantissa.

Exception Status & Mask Bits -- "1" in Status Register indicates exception:
"1" in MCR masks exception trap.
P - Precision error.
U - Underflow error.
O - Overflow error.
Z - Divide by zero error.
D - Denormalized operand error.
I - Invalid operation.

4.4 Exception Handling

The CX-83S87 performs extensive data checking during the data input and type conversion process and during the process of performing calculations. The detection of an inconsistency or an error is considered an exception condition and is reported in the CX-83S87 status word. Optionally, exceptions can initiate an exception trap sequence in the host processor.

There are seven types of exceptions that the CX-83S87 con detect and report: Precision Exception, Underflow Exceptian, Overflow Exception, Divide By Zero Exception, Denormal Operand Exception, Involid Operation Exceptian, and Register Error. These seven exceptions map into six separately enabled exception traps (invalid operation and register errors are reported on the same trap). Exception traps are unmasked (enabled) by writing a "0" into the appropriate bit of the CX-83S87 Mode Control Register.

When the CX-83S87 arithmetic unit detects an exception, it sets the corresponding exception status bit in the CX-83S87 Status Word in all cases except Underflow when Underflow is masked. In this case, the Underflaw bit is only set if Precision Exception is also set. The CX-83S87 Interface Unit compares the unmasked exceptions in the CX-83S87 Control Word against the exception status bits maintained in the Status Word. When an unmasked exception is detected, a trap sequence is initiated by the Interface Unit by asserting the CX-83S87 ERROR# output signal. Note that unmasking an exception will couse an immediate trap if the exception status bit is set.

The detectian of an exception during the execution of an instruction results in one of two possible outcomes. If the exception is masked, the CX-83S87 generates the IEEE-754-1985 specified result, stores the result in the destination location, and proceeds with program execution. If exceptions are unmasked and an exception is Precision, Underflow, or Overflow, the IEEE-754-1985 specified result is stored in the destination and an Error trap sequence is initiated. If exceptians are unmasked and the exceptian is Zero Divide, Denormal, ar Invalid, the offending instruction is abarted, no result is written, and the Error trap sequence is begun.

4.5 Precision Exception

This exception is caused by the productian of a result which is not exactly representable in the destination locatian af the instruction. For instance, the number 1/10 is not exactly representable as a binary floating point number. Similarly, all transcendental and elementary functions return approximations to the infinitely precise, irrational value. Both situatians will cause the Precision Exceptian to be set.

4.6 Underflow Exception

This exception is caused by the production of a result which is tiny in magnitude and requires either a denarmal or zero result to represent the value. The significance of the Underflow Exception depends on the state of the Underflow Mask bit. When Underflow is masked the exception status is set only when the result is both denormal/zero and precision has been last .

When Underflow is unmasked, the Underflow Exception occurs whenever the result would be denormal/zero. In this case, if the destination is a stack register, the true result is

multiplied by $2^{24,576}$, rounded, and stored in the destination register. This behavior occurs for all underflow cases except the extreme underflow produced by the FSCALE instruction. FSCALE extreme underflow will produce +/- 0 just as if Underflow was masked. If the destination is memory, the instruction is aborted and no result is written.

4.7 Overflow Exception

This exception is caused by the production of a result which is larger in magnitude than the destination format can accommodate. When Overflow is masked the exception status is set and the result is rounded in accordance with the RC mode bits.

When Overflow is unmasked, the Overflow Exception causes the results to be scaled to fit in the destination format. In this case, if the destination is a stack register, the true result is divided by $2^{24,576}$, rounded, and stored in the destination register. This behavior occurs for all overflow cases except the extreme overflow produced by the FSCALE instruction. FSCALE extreme overflow will produce +/- ∞ just as if Overflow was masked. If the destination is memory, the instruction is aborted and no result is written.

4.8 Denormal Exception

This exception is caused by the use of a denormal operand as input to an instruction. When Denormal is masked the instruction proceeds to completion using the denormal operand.

When Denormal is unmasked the exception status is set and an exception trap sequence is initiated without writing any result to the destination of the instruction.

4.9 Divide By Zero Exception

This exception is caused by an attempt to divide a finite non-zero operand by zero. In addition to the divide instruction, the FXTRACT and FYL2X instructions may generate this exception. The masked response to this exception is to return ∞ with its sign set to the exclusive-or of the operands' signs. For FXTRACT, ST(1) is set to -∞ and TOS is set to zero with its sign set to the sign of the initial operand.

When Divide By Zero is unmasked the exception status is set and an exception trap sequence is initiated without writing any result to the destination of the instruction.

4.10 Stack Error Exception

Register errors (cf. sec 2.4.2) produce the stack error exception. When masked, the QNaN indefinite overwrites the destination. When unmasked, an exception trap sequence is initiated and the destination is left undisturbed. This exception is indicated by simultaneously setting the Invalid and Stack Fault bits in the Status Word.

When a Stack Error is signaled, the C1 status bit indicates whether the error was Destination Register Full (C1="1") or Source Register Empty (C1="0").

4.11 Invalid Operation Exception

The IEEE-754-1985 Standard provides for the detection and reporting of attempts to perform orithmetic on operands that cannot provide meaningful results. Such attempts

are called Invalid Operations. When Invalid Exception is unmasked, the occurrence of such an event causes the instruction to be aborted, a trap sequence to be initiated, and the destination to be left undisturbed. When Invalid is masked, the appropriate IEEE specified response is generated and written into the destination.

The following table defines invalid operations and their default (masked) results:

Operand(s)	Invalid Operation	Default Result
Unsupported	Any Arithmetic	QNaN Indefinite.
*NaN or (NaN,Valid)	Any Arithmetic	Quietized Nan.
QNaN & SNaN	Any Arithmetic	The QNaN.
SNaN & SNaN	Any Arithmetic	Larger SNaN Quietized.
*QNaN & QNaN	Any Arithmetic	Larger QNaN.
NaN	FCOM(P)(P) & FTST	CC= Unordered.
(+∞,-∞) or (-∞,+∞)	Addition	QNaN Indefinite.
(+∞,+∞) or (-∞,-∞)	Subtraction	QNaN Indefinite.
0,∞	Multiplication	QNaN Indefinite.
(∞,∞) or (0,0)	Division	QNaN Indefinite.
Zero Divisor	Partial Remainder	QNaN Indefinite.
∞ Dividend	Partial Remainder	QNaN Indefinite.
∞	Forward Trigonometric	QNaN Indefinite.
Negative Number	Square Root, Log_2	QNaN Indefinite.
Empty Reg, NaN, ∞	Integer & BCD Store	Integer/BCD Indefinite.
Exceeds Integer Range	Integer & BCD Store	Integer/BCD Indefinite.
Empty Register	Exchange Registers	Set Empty to QNaN Indefinite & exchange.

***Note:** Generally the use of a QNaN as an operand in these contexts does not generate the Invalid Exception. All other Invalid Operations in the table produce the Invalid Exception.

Certain other combinations of valid numbers, zeroes, and infinities may give rise to the Invalid Exception for FSCALE, FYL2X, & FYL2XP1. Please consult section 4.12, Detailed Instruction Descriptions, for details.

4.12 Detailed Instruction Descriptions

The detailed instruction descriptions provide a complete reference on the use of each CX-83S87 instruction. The instruction mnemonic, its syntactical variants, allowed operands, encoding, operation, effect on status, possible exceptions, and special operands/results are presented for each instruction type.

Please note that integer variants which cause insertion of the letter "I" and BCD variants which cause insertion of the letter "B" into their mnemonics are included in the listings under their basic type, i.e. FIADD is described under FADD. With this single exception, all the CX-83S87 instructions are catalogued alphabetically by mnemonic in this section.

The integer variants presented under their basic type are: FIADD, FICOM, FIDIV, FILD, FIMUL, FIST, & FISUB. Similarly, the BCD integer forms are grouped with FLD (FBLD) and FST (FBSTP).

Detailed instruction descriptions are presented in the following standard format:

Syntax:	A BNF-like description of the assembler syntax of the instruction.
Forms:	A complete list of the permitted operand source & destination combinations for the instruction.
Operands:	A detailed table showing all permitted operand types, the instruction encoding for each type, and the execution time in clock cycles.
Operation:	A textual description of the action caused by the execution of the instruction.
Status:	A table which summarizes the effect on CX-83S87 condition codes of each instruction.
Exceptions:	A table which summarizes the possible exceptions an instruction can produce and the resulting exception status bits in the Status Register.
Zero/Inf:	A summary of the results produced when executed with extraordinary valued operands.
Notes:	A brief statement of points of special interest not covered elsewhere.

The following symbols and abbreviations are used throughout the detailed instruction descriptions:

Syntax:	Square brackets ("[]") indicate an optional argument. Nesting is permitted. Angle brackets ("<>") indicate an argument of type "<*type*>" must be supplied. Allowed types are: DST Destination operand. SRC Source operand. Literals are: "P" (pop), "R" (reverse), and ",".
Forms:	<TOS> refers to the Top of Stack register, denoted by "ST(0)" to the Intel assembler. <Reg> refers to an explicitly specified CX-83S87 data register denoted by "ST(i)" where 0≤i≤7. <Memory> refers to a legal 386SX memory operand address, e.g. "DWORD PTR 0100".

Operands: Source Operand specifications are given for each form of the instruction. The key to the specifications is:

16-bit Integer	Memory location "WORD PTR"
32-bit Integer	Memory location "DWORD PTR"
64-bit Integer	Memory location "QWORD PTR"
32-bit Real	Memory Location "DWORD PTR"
64-bit Real	Memory Location "QWORD PTR"
80-bit Real	Memory Location "TBYTE PTR"
18 dig BCD Int.	Memory location "TBYTE PTR"
80-bit Register	Explicit register "ST(i)"
Top of Stack	Register "ST(0)"

Encoding fields may have the following values:

$XX	Hexadecimal value of 1st byte.
MD	2-bit 386SX "MOD" field.(of MOD R/M.)
R/M	3-bit 386SX "Reg/Mem" field.(MOD R/M.)
p	1-bit specifying "pop"=1/no "pop"=0.
REG	3-bits specifying CX-83S87 data register.
SIB	386SX Stack Index Base field.
DISP	386SX Displacement field.

Cycle counts are given in 386SX clock cycles and assume no wait states and no DMA overhead in the host system.

Status: Status bits are indicated as:

U	Undefined after instruction execution.
1	Set to "1" as a result of instruction.
0	Set to "0" as a result of instruction.
M	Set to the value loaded from memory.

Exceptions: Exceptions are listed by type and specify the result stored in the destination based on the trap mode: masked or unmasked. Exceptions which are not possible for a given instruction are shown as "N/A". Exception status bits are indicated as:

-	Unaffected by instruction execution.
1	Set to "1" given specified exception.
0	Set to "0" unconditionally.
M	Set to the value loaded from memory.

Zero/Inf: Special symbols used have the following definitions:

"->"	"Converted to".
sgn()	Arithmetic sign of argument ().
"X"	A positive, finite, nonzero integer.
"Y"	A positive, finite, nonzero integer.
NoN	Not a Number.
R(0)	Zero as produced by current RC mode.

F2XM1

Function Evaluation: 2^X-1.

F2XM1

Syntax: **F2XM1**

Forms: **F2XM1**

Operands:

Inst	Source Operand	Encoding			Cycles
F2XM1	Top of Stack	$D9	11 110 000		14-66

Operation: The Top of Stack contains the source operand (x). The function $y = 2^X-1$ is evaluated and the result is normalized and rounded according to the RC mode in effect. The result (y) replaces (x) in the Top of Stack. The source operand (x) must be in the range $-1 \le x \le 1$. To obtain the value 2^X ($\text{antilog}_2(x)$), add 1 to the result.

Arguments $|x| \ge 1.0$ produce undefined results without signaling any exceptions.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Operand Used	-	-	-	-	-	1	-
	Unmasked	Trap/Unaltered	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Trap/Unaltered	-	-	-	-	-	-	1
Reg Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity:

X	Result
+0	+0
-0	-0
-∞	-1
+∞	+∞

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FABS Floating Absolute Value FABS

Syntax: **FABS**

Forms: **FABS**

Operands:

Inst	Source Operand		Encoding		Cycles
FABS	Top of Stack	$D9	11 100 001		4

Operation: The Top of Stack is always the source operand. The sign of the Top of Stack is set to zero (positive).

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Reg Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity:

OP1	Result
+0	+0
-0	+0
-∞	+∞
+∞	+∞

Notes: None.

FADD — Floating Point Add — FADD

Syntax: **FADD(P) ((<DST>,)<SRC>)**

Forms:

FADD	**<TOS>,<Memory>**
FADD	**<TOS>,<Register>**
FADD(P)	**<Register>,<TOS>**

Operands:

Inst	Source Operand	Encoding			Cycles
FIADD	16-bit Integer	$DE	MD 000 R/M	SIB,DISP	13
FIADD	32-bit Integer	$DA	MD 000 R/M	SIB,DISP	15
FADD	32-bit Real	$D8	MD 000 R/M	SIB,DISP	15
FADD	64-bit Real	$DC	MD 000 R/M	SIB,DISP	19
FADD	80-bit Register	$D8	11 000 REG		6
FADD	Top of Stack	$DC	11 000 REG		6
FADDP	Top of Stack	$DE	11 000 REG		6

Operation: The source and destination operands are fetched. The source is converted to extended precision format if necessary. The operands are added and the result is normalized and rounded according to the RC mode in effect at the precision specified by the PC mode bits. The result is stored in the destination register. When the "pop" form is used, the top of stack is popped.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	Masked	R(∞)	-	-	-	1	-	-	-
	Unmasked	Round & Scale	-	-	-	1	-	-	-
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FADD Floating Point Add FADD

Zero/Infinity:

OP1	OP2	Result	OP1	OP2	Result
+0	+0	+0	+∞	+∞	+∞
-0	-0	-0	-∞	-∞	-∞
+0	-0	R(0)	+∞	-∞	Inv Op
-0	+0	R(0)	-∞	+∞	Inv Op
-X	+X	R(0)	+∞	X	+∞
+X	-X	R(0)	-∞	X	-∞

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FCHS Floating Change Sign FCHS

Syntax: **FCHS**

Forms: **FCHS**

Operands:

Inst	Source Operand		Encoding		Cycles
FCHS	Top of Stack	$D9	11 100 000		4

Operation: The Top of Stack is always the source operand. The sign of the Top of Stack is complemented.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Reg Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity:

OP1	Result	OP1	Result
+0	-0	+∞	-∞
-0	+0	-∞	+∞

Notes: None.

FCLEX Clear Exceptions FCLEX

Syntax: **FCLEX**

Forms: **FCLEX**

Operands:

Inst	Source Operand		Encoding		Cycles
FCLEX	None	$DB	11 100 010		4

Operation: All CX-83S87 exception flags are reset to zero. The busy flag is reset to zero. ERROR# goes inactive.

Status:

Result of instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	All Cleared	0	0	0	0	0	0	0

Zero/Infinity: None.

Notes: None.

FCOM

Floating Point Compare

FCOM

Syntax: **FCOM(P) ((<DST>,)<SRC>)**

Forms: **FCOM(P) <TOS>,<Memory>**
FCOM(P)(P) <TOS>,<Reg>

Operands:

Inst	Source Operand		Encoding		Cycles
FICOM(P)	16-bit Integer	$DE	MD 01p R/M	SIB,DISP	11
FICOM(P)	32-bit Integer	$DA	MD 01p R/M	SIB,DISP	13
FCOM(P)	32-bit Real	$D8	MD 01p R/M	SIB,DISP	13
FCOM.D	64-bit Real	$DC	MD 01p R/M	SIB,DISP	17
FCOM(P)	80-bit Reg	$D8	11 01p REG		4
FCOMPP	80-bit Reg	$DE	11 011 001		4

Operation: The source operand is fetched and converted to extended precision format if necessary. The source operand is subtracted from the destination (Top of Stack) and the condition codes are set according to the result. When the "P" form is used, the Top of Stack is popped. The "PP" form causes two "pop" operations to take place.

The result "unordered" is produced when the operands are NaNs, unsupported or when Stack Fault occurs. These operands also cause the Invalid Exception.

Status:

Result of Instruction		C3	C2	C1	C0
Normal Execution:	DST > SRC	0	0	0	0
	DST < SRC	0	0	0	1
	DST = SRC	1	0	0	0
	Unordered	1	1	0	1
Register Error:	Source Reg Empty	1	1	0	1

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	QNaN	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	"Unordered"	-	-	-	-	-	-	1
	Unmasked	Trap/Abort[2]	-	-	-	-	-	-	1
Register Error	Masked	"Unordered"	1	-	-	-	-	-	1
	Unmasked	Trap/Abort[2]	1	-	-	-	-	-	1

FCOM Floating Point Compare FCOM

Zero/Infinity:

DST -	SRC	Result	DST -	SRC	Result
+0	+0	=	+∞	+∞	=
-0	-0	=	-∞	-∞	=
+0	-0	=	+∞	-∞	DST>SRC
-0	+0	=	-∞	+∞	DST<SRC
+0	+X	DST<SRC	+∞	X	DST>SRC
-0	+X	DST<SRC	-∞	X	DST<SRC
+0	-X	DST>SRC	X	-∞	DST>SRC
-0	-X	DST>SRC	X	+∞	DST<SRC
NaN	X	Unordered	X	NaN	Unordered

Notes:

1. QNAN operands produce the invalid exception for this instruction.
2. CC bits are set to unordered and ony "pop" operations are inhibited.

FCOS

Function Evaluation: Cos(x).

FCOS

Syntax: **FCOS**

Forms: **FCOS**

Operands:

Inst	Source Operand	Encoding			Cycles
FCOS	Top of Stack	$D9	11 111 111		5-97

Operation: The Top of Stack contains the source operand (x). The instruction requires (x) in radians and returns (y) such that y= cos(x). The result (y) is normalized and rounded according to the RC mode in effect. The value (y) replaces the contents of the Top of Stack. The source operand (x) must be in the range $|x| \leq 2^{63}$.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Incomplete Reduction of (x):	U	1	0	U
Normal completion:	U	0	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Operand Used	-	-	-	-	-	1	-
	Unmasked	Trap/Unaltered	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Trap/Unaltered	-	-	-	-	-	-	1
Register Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	Result
+0	+1
-0	+1
+∞	Invalid Op.
-∞	Invalid Op.

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FDECSTP Decrement CX-83S87 Stack Pointer FDECSTP

Syntax: **FDECSTP**

Forms: **FDECSTP**

Operands:

Inst	Source Operand		Encoding		Cycles
FDECSTP	None	$D9	1111 0110		5

Operation: The CX-83S87 data register stack pointer is decremented modulo 8. The result of decrementing SP=0 is SP=7.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: This instruction has no effect on any data register's contents or the Tag Word.

FDIV — Floating Point Divide — FDIV

Syntax: **FDIV(R)(P) ((<DST>,)<SRC>)**

Forms:

FDIV(R) <TOS>,<Memory>
FDIV(R) <TOS>,<Reg>
FDIV(R)(P) <Reg>,<TOS>

Operands:

Inst	Source Operand		Encoding		Cycles
FIDIV(R)	16-bit Integer	$DE	MD 11r R/M	SIB,DISP	20-31*
FIDIV(R)	32-bit Integer	$DA	MD 11r R/M	SIB,DISP	22-33*
FDIV(R)	32-bit Real	$D8	MD 11r R/M	SIB,DISP	22-33*
FDIV(R)	64-bit Real	$DC	MD 11r R/M	SIB,DISP	25-36*
FDIV(R)	80-bit Reg	$D8	11 11r REG		13-24*
FDIV	Top of Stack	$DC	11 111 REG		14-25
FDIVR	Top of Stack	$DC	11 110 REG		13-24
FDIVP	Top of Stack	$DE	11 111 REG		14-25
FDIVRP	Top of Stack	$DE	11 110 REG		13-24

*Add one clock cycle to these instruction execution times for reversed operands.

Operation: The source operand is fetched and converted to extended precision format if necessary. The destination is divided by the source and the quotient is normalized and rounded according to the RC mode in effect at the precision specified by the PC mode bits. The quotient replaces the original contents of the destination register. When the "pop" form is used, the top of stack is popped.

The "reverse" form causes the source to be divided by the destination. The quotient is placed in the destination.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

FDIV — Floating Point Divide — FDIV

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	Masked	R(∞)	-	-	-	1	-	-	-
	Unmasked	Round & Scale	-	-	-	1	-	-	-
Div by Zero	Masked	∞ (Note 2)	-	-	-	-	1	-	-
	Unmasked	Trap/Unaltered	-	-	-	-	1	-	-
Denormal	Masked	Operand Used	-	-	-	-	-	1	-
	Unmasked	Trap/Unaltered	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Trap/Unaltered	-	-	-	-	-	-	1
Register Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

OP1	OP2	Result	OP1	OP2	Result
+/-0	+/-0	Invalid	+/-∞	+/-∞	Invalid
+0	+X	+0	+∞	+X	+∞
-0	-X	+0	+∞	-X	-∞
+0	-X	-0	-∞	+X	-∞
-0	+X	-0	-∞	-X	+∞
+X	+Y	$+0^3$	+∞	+0	+∞
-X	-Y	$+0^3$	+∞	-0	-∞
+X	-Y	-0^3	-∞	+0	-∞
-X	+Y	-0^3	-∞	-0	+∞

Notes:

1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

2. Division by zero produces an infinity result with a sign equal to the exclusive-or of the signs of the operands when Divide By Zero Exception is masked.

3. Applies to division of X by Y producing extreme denormalization or underflow when Underflow Exception is masked..

FFREE — Free Floating Point Register — FFREE

Syntax: **FFREE (<DST>)**

Forms: **FFREE <Register>**

Operands:

Inst	Source Operand		Encoding		Cycles
FFREE	80-bit Register	$DD	11 000 REG		5

Operation: The destination register tag is changed to empty. The contents of the register are unaffected.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: None.

FINCSTP Increment CX-83S87 Stack Pointer FINCSTP

Syntax: **FINCSTP**

Forms: **FINCSTP**

Operands:

Inst	Source Operand	Encoding			Cycles
FINCSTP	None	$D9	1111 0111		5

Operation: The CX-83S87 data register stack pointer is incremented modulo 8. The result of incrementing SP=7 is SP=0.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: This instruction has no effect on any data register's contents or the Tag Word.

FINIT CX-83S87 Initialize FINIT

Syntax: **FINIT**

Forms: **FINIT**

Operands:

Inst	Source Operand		Encoding		Cycles
FINIT	None	$DB	11 100 011		5

Operation: The CX-83S87 is set to its reset condition. The CX-83S87 Mode Control Register is set to $037F_{16}$, the CX-83S87 Status Register is reset to $0000, and all data registers are marked empty (Tag word=$FFFF).

This action sets Rounding control to "round to neorest or even" (RC=00), Precision control to 64-bit extended precision (PC=11), and the Top of Stack register number to zero (SSS=000). All exceptions are cleared (=0), all condition codes are cleared (C_0-C_3=0), and all exceptions are masked (=1).

This instruction differs from a hardware reset by setting MCR bit 0 (Invalid Exception Mask) to 1 and Status Register bits 7 & 0 (Error and Invalid Exception) to 0.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	0	0	0	0

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	0	0	0	0	0	0	0

Zero/Infinity: None.

Notes: None.

FLD — Load CX-83S87 Register — FLD

Syntax: **FLD ((<DST>,)<SRC>)**

Forms: **FLD <TOS>,<Memory>**
FLD <TOS>,<Register>

Operands:

Inst	Source Operand		Encoding		Cycles
FILD	16-bit Integer	$DF	MD 000 R/M	SIB,DISP	7
FILD	32-bit Integer	$DB	MD 000 R/M	SIB,DISP	9
FILD	64-bit Integer	$DF	MD 101 R/M	SIB,DISP	13
FBLD	18 dig BCD Int.	$DF	MD 100 R/M	SIB,DISP	36
FLD	32-bit Real	$D9	MD 000 R/M	SIB,DISP	12
FLD	64-bit Real	$DD	MD 000 R/M	SIB,DISP	16
FLD	80-bit Real	$DB	MD 101 R/M	SIB,DISP	15
FLD	Top of Stack	$D9	11 000 REG		4

Operation: The source operand is fetched and converted to extended precision format if necessary. The operand is stored in the destination register. If TOS is the destination, the SSS field of the Control Register is pre-decremented modulo 8.

Status:

Result of Instruction		C3	C2	C1	C0
Normal Execution:		U	U	0	U
Register Error:	Dest. Reg Full	U	U	1	U
	Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal[1]	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op[1]	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

OP1	Result	OP1	Result
+0	+0	+∞	+∞
-0	-0	-∞	-∞

Notes: 1. Denormal and invalid exceptions cannot occur as a result of FLD 80-bit real or FLD <reg> instructions..

FLD1

Load Floating Constant= 1.0

FLD1

Syntax: **FLD1**

Forms: **FLD1**

Operands:

Inst	Source Operand	Encoding			Cycles
FLD1	80-bit Internal	$D9	11 101 000		4

Operation: The 80-bit extended precision constant one (1.0) is pushed onto the Stack.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FLDCW — Load CX-83S87 Mode Control Register — FLDCW

Syntax: **FLDCW** **<SRC>**

Forms: **FLDCW** **<Memory>**

Operands:

Inst	Source Operand		Encoding		Cycles
FLDCW	2 Bytes	$D9	MD 101 R/M	SIB,DISP	4

Operation: The CX-83S87 Mode Control Register is loaded with the contents of the specified memory location. Note that exceptions indicated in the Status Register due to previous operations may cause an ERROR# trap sequence if the corresponding mask bit in the MCR is set to 0 by this instruction.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: None.

FLDENV Load CX-83S87 Environment FLDENV

Syntax: **FLDENV <SRC>**

Forms: **FLDENV <Memory>**

Operands:

Inst	Source Operand	Encoding			Cycles
FLDENV	14 or 28 Bytes	$D9	MD 100 R/M	SIB,DISP	22

Operation:

The CX-83S87 "Environment" is loaded from the memory location specified. The "Environment" consists of the Mode Control Word, the Status Register, and the Tag Word which are loaded into the CX-83S87. The "Environment" also includes the CX-83S87 Instruction Pointer and the CX-83S87 Data Pointer which are loaded into 80386 CPU registers during execution of this instruction.

The format of the "Environment" data structure is dependent on the operating mode of the 80386 CPU and the operand size in effect.

Loading an "Environment" that contains a Status Register field with an exception indicated and an MCR with that exception enabled causes an ERROR# trap sequence when the next WAIT or exception checking CX-83S87 instruction is executed.

The ERROR# signal is unconditionally de-asserted while the "Environment" data is loaded. If the newly loaded "Environment" calls for an exception trap, ERROR# will be asserted upon completion of the "Environment" data transfer. A subsequent WAIT or exception checking instruction will execute a trap sequence.

32-bit Protected Mode:

31 ... 15	... 0
Reserved	Mode Control Word
Reserved	Status Word
Reserved	Tag Word
Instruction Pointer Offset	
00000 Opcode(10:0)	Code Segment Selector
Data Operand Offset	
Reserved	Operand Seg Selector

32-bit Real Mode:

31 ... 15	... 0
Reserved	Mode Control Word
Reserved	Status Word
Reserved	Tag Word
Reserved	Instruction Ptr(15:0)
0000 Instruction Ptr(31:16)	0 Opcode(10:0)
Reserved	Operand Ptr(15:0)
0000 Operand Ptr(31:16)	0000 0000 0000

FLDENV Load CX-83S87 Environment FLDENV

16-bit Protected Mode:

15	0
Mode Control Word	
Status Word	
Tag Word	
Instruction Ptr Offset	
Code Segment Selector	
Data Operand Offset	
Operand Seg Selector	

16-bit Real Mode:

15			0
Mode Control Word			
Status Word			
Tag Word			
Instruction Ptr(15:0)			
IP(19:16)	0	Opcode(10:0)	
Operand Ptr(15:0)			
DP(19:16)	0000	0000	0000

Status:

Result of Instruction	C3	C2	C1	C0
Loaded from Memory:	M	M	M	M

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Loaded	N/A	N/A	M	M	M	M	M	M	M

Zero/Infinity: None.

Notes: None.

FLDL2E Load Floating Constant= $Log_2(e)$ FLDL2E

Syntax: **FLDL2E**

Forms: **FLDL2E**

Operands:

Inst	Source Operand		Encoding		Cycles
FLDL2E	80-bit Const.	$D9	11 101 010		7

Operation: The 80-bit extended precision constant approximating $Log_2(e)$ is pushed onto the Stack. The constant is rounded according to the RC mode in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FLDL2T — Load Floating Constant= Log2(10) — FLDL2T

Syntax: **FLDL2T**

Forms: **FLDL2T**

Operands:

Inst	Source Operand	Encoding			Cycles
FLDL2T	80-bit Const.	$D9	11 101 001		8

Operation: The 80-bit extended precision constant approximating $Log_2(10)$ is pushed onto the Stack. The constant is rounded according to the RC mode in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FLDLG2 Load Floating Constant= $Log_{10}(2)$ FLDLG2

Syntax: **FLDLG2**

Forms: **FLDLG2**

Operands:

Inst	Source Operand	Encoding			Cycles
FLDLG2	80-bit Const.	$D9	11 101 100		8

Operation: The 80-bit extended precision constant approximating $Log_{10}(2)$ is pushed onto the Stack. The constant is rounded according to the RC mode in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FLDLN2 Load Floating Constant= Ln(2) FLDLN2

Syntax: **FLDLN2**

Forms: **FLDLN2**

Operands:

Inst	Source Operand		Encoding		Cycles
FLDLN2	80-bit Const.	$D9	11 101 101		6

Operation: The 80-bit extended precision constant approximating Ln(2) is pushed onto the Stack. The constant is rounded according to the RC mode in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FLDPI

Load Floating Constant= π.

FLDPI

Syntax: **FLDPI**

Forms: **FLDPI**

Operands:

Inst	Source Operand		Encoding		Cycles
FLDPI	80-bit Const.	$D9	11 101 011		6

Operation: The 80-bit extended precision constant approximating π (Pi) is pushed onto the Stack. The constant is rounded according to the RC mode in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FLDZ — Load Floating Constant= 0.0. — FLDZ

Syntax: **FLDZ**

Forms: **FLDZ**

Operands:

Inst	Source Operand		Encoding		Cycles
FLDZ	80-bit Const.	SD9	11 101 110		4

Operation: The 80-bit extended precision constant zero (0.0) is pushed onto the Stack.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Dest. Reg Full	U	U	1	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

Zero/Infinity: None.

Notes: None.

FMUL — Floating Point Multiply — FMUL

Syntax: **FMUL(P) {(<DST>,)<SRC>)**

Forms:
FMUL <TOS>,<Memory>
FMUL <TOS>,<Register>
FMUL(P) <Register>,<TOS>

Operands:

Inst	Source Operand	Encoding			Cycles
FIMUL	16-bit Integer	$DE	MD 001 R/M	SIB,DISP	16
FIMUL	32-bit Integer	$DA	MD 001 R/M	SIB,DISP	18
FMUL	32-bit Real	$D8	MD 001 R/M	SIB,DISP	18
FMUL	64-bit Real	$DC	MD 001 R/M	SIB,DISP	22
FMUL	80-bit Register	$D8	11 001 REG		10
FMUL	Top of Stack	$DC	11 001 REG		10
FMULP	Top of Stack	$DE	11 001 REG		10

Operation: The source and destination operands are fetched. The source is converted to extended precision format if necessary. The operands are multiplied and the result is normalized and rounded according to the RC mode in effect at the precision specified by the PC mode bits. The result is stored in the destination register. When the "pop" form is used, the top of stack is popped.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	Masked	R(∞)	-	-	-	1	-	-	-
	Unmasked	Round & Scale	-	-	-	1	-	-	-
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FMUL Floating Point Multiply FMUL

Zero/Infinity:

OP1	OP2	Result	OP1	OP2	Result
+0	+0	+0	+∞	+∞	+∞
+0	-0	-0	+∞	-∞	-∞
-0	+0	-0	-∞	+∞	-∞
-0	-0	+0	-∞	-∞	+∞
+X	+0	+0	+∞	+X	+∞
+X	-0	-0	+∞	-X	-∞
-X	+0	-0	-∞	+X	-∞
-X	-0	+0	-∞	-X	+∞
+X	+Y	$+0^2$	+∞	+0	Inv. Op.
+X	-Y	-0^2	+∞	-0	Inv. Op.
-X	+Y	-0^2	-∞	+0	Inv. Op.
-X	-Y	$+0^2$	-∞	-0	Inv. Op.

Notes:

1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

2. For cases in which X times Y produces extreme underflow and Underflow Exception is masked, the result is denormalized to zero.

FNOP No Operation FNOP

Syntax: **FNOP**

Forms: **FNOP**

Operands:

Inst	Source Operand		Encoding		Cycles
FNOP	None	SD9	11 010 000		4

Operation: No operation is performed in the CX-83S87.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	P	U	O	Z	D	I	S
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: None.

FPATAN — Function Eval: $\tan^{-1}(\frac{y}{x})$ — FPATAN

Syntax: **FPATAN**

Forms: **FPATAN**

Operands:

Inst	Source Operand		Encoding		Cycles
FPATAN	Top of Stack	$D9	11 110 011		90-127

Operation: The Top of Stack contains the first source operand (x). The next to Top of Stack contains the second source operand (y). The instruction computes (z) in radians such that $z = \tan^{-1}(\frac{y}{x})$. The result (z) is normalized and rounded according to the RC mode in effect. The Top of Stack (x) is popped. The value (z) replaces the contents of the new Top of Stack (y). The source operands (x) and (y) are unrestricted in range.

The result (z) falls in the range $-\pi \le z \le +\pi$ according to the following table:

Sgn(y)	Sgn(x)	$\lvert\frac{y}{x}\rvert$	Result
+	+	<1	$0 < z < \frac{\pi}{4}$
+	+	>1	$\frac{\pi}{4} < z < \frac{\pi}{2}$
+	-	>1	$\frac{\pi}{2} < z < \frac{3\pi}{4}$
+	-	<1	$\frac{3\pi}{4} < z < \pi$
-	+	<1	$0 > z > -\frac{\pi}{4}$
-	+	>1	$-\frac{\pi}{4} > z > -\frac{\pi}{2}$
-	-	>1	$-\frac{\pi}{2} > z > -\frac{3\pi}{4}$
-	-	<1	$-\frac{3\pi}{4} > z > -\pi$

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal completion:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

FPATAN Function Eval: $\text{Tan}^{-1}(\frac{y}{x})$ FPATAN

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

y	x	Result
$y = +0$	$+\infty \geq x \geq +0$	$z = +0$
$y = -0$	$+\infty \geq x \geq +0$	$z = -0$
$y = +0$	$-\infty \leq x \leq -0$	$z = +\pi$
$y = -0$	$-\infty \leq x \leq -0$	$z = -\pi$
$y > +0$	$x = 0$	$z = +\frac{\pi}{2}$
$y > +0$	$x = +\infty$	$z = +0$
$y > +0$	$x = -\infty$	$z = +\pi$
$y < -0$	$x = 0$	$z = -\frac{\pi}{2}$
$y < -0$	$x = +\infty$	$z = -0$
$y < -0$	$x = -\infty$	$z = -\pi$
$y = +\infty$	$-\infty < x < +\infty$	$z = +\frac{\pi}{2}$
$y = +\infty$	$x = +\infty$	$z = +\frac{\pi}{4}$
$y = +\infty$	$x = -\infty$	$z = +\frac{3\pi}{4}$
$y = -\infty$	$-\infty < x < +\infty$	$z = -\frac{\pi}{2}$
$y = -\infty$	$x = +\infty$	$z = -\frac{\pi}{4}$
$y = -\infty$	$x = -\infty$	$z = -\frac{3\pi}{4}$

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FPREM Floating Point Remainder (NON-IEEE) FPREM

Syntax: **FPREM**

Forms: **FPREM**

Operands:

Inst	Source Operand		Encoding		Cycles
FPREM	Top of Stack	SD9	11 111 000		48

Operation: The Top of Stack contains the source operand (x). The next to Top of Stack contains the operand (y). The remainder (z) is calculated such that $z = x - y*q$ where (q) is the quotient $q = \frac{x}{y}$ obtained by chopping the exact value toward zero. The result (z) is replaces the contents of the Top of Stack.

FPREM will reduce the exponent of the argument (x) by 63 or more in each pass. The C2 status bit indicates whether the result has been reduced completely, i.e., whether $|z| < |y|$. When completely reduced, the quotient (q) modulo 8 may be read from the condition register.

Status:

Result of Instruction		C3	C2	C1	C0
Reduction Incomplete:		0	1	0	0
Reduction Complete:(q)mod8=	0	0	0	0	0
	1	0	0	1	0
	2	1	0	0	0
	3	1	0	1	0
	4	0	0	0	1
	5	0	0	1	1
	6	1	0	0	1
	7	1	0	1	1
Register Error: Source Reg Empty		U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	Masked	Denormal	-	-	-	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FPREM — Floating Point Remainder (NON-IEEE) — FPREM

Zero/Infinity:

x	y	Result
x = 0	y = 0	Invalid Op.
x ≠ 0	y = 0	Invalid Op.
x = -0	y ≠ 0	-0
x = +0	y ≠ 0	+0
x = ∞	--	Invalid Op.
-∞ < x < +∞	y = ∞	x; q = 0

Notes: The sign of the remainder is the same as the sign of (x). If (y) evenly divides (x) the remainder is zero.

FPREM1 Floating Point Remainder (IEEE) FPREM1

Syntax: **FPREM1**

Forms: **FPREM1**

Operands:

Inst	Source Operand		Encoding		Cycles
FPREM1	Top of Stack	$D9	11 110 101		49

Operation: The Top of Stack contains the source operand (x). The next to Top of Stack contains the operand (y). The remainder (z) is calculated such that $z = x - y*q$ where (q) is the quotient $q = \frac{x}{y}$ obtained by rounding the exact value to nearest/even. The result (z) replaces the contents of the Top of Stack. This instruction is compatible with the IEEE 754-1985 specification.

FPREM1 will reduce the exponent of the argument (x) by 63 or more in each pass. The C2 status bit indicates whether the result has been reduced completely, i.e., whether $|z| \leq |\frac{y}{2}|$. When completely reduced, the quotient (q) modulo 8 may be read from the condition register.

Status:

Result of Instruction		C3	C2	C1	C0
Reduction Incomplete:		0	1	0	0
Reduction Complete:(q)mod8=	0	0	0	0	0
	1	0	0	1	0
	2	1	0	0	0
	3	1	0	1	0
	4	0	0	0	1
	5	0	0	1	1
	6	1	0	0	1
	7	1	0	1	1
Register Error: Source Reg Empty		U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	Masked	Denormal	-	-	-	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FPREM1 Floating Point Remainder (IEEE) FPREM1

Zero/Infinity:

x	y	Result
x = 0	y = 0	Invalid Op.
x ≠ 0	y = 0	Invalid Op.
x = -0	y ≠ 0	-0
x = +0	y ≠ 0	+0
x = ∞	–	Invalid Op.
-∞ < x < +∞	y = ∞	x; q = 0

Notes: The sign of the remainder is not necessarily the same as the sign of (x). If (y) evenly divides (x) the remainder is zero.

FPTAN — Function Eval: Tan(x) — FPTAN

Syntax: **FPTAN**

Forms: **FPTAN**

Operands:

Inst	Source Operand	Encoding			Cycles
FPTAN	Top of Stack	$D9	11 110 010		5-82

Operation: The Top of Stack contains the source operand (x). The instruction requires (x) in radians and returns (y) such that y= tan(x). The result (y) is normalized and rounded according to the RC mode in effect. The value (y) replaces the contents of the Top of Stack. The value 1 is pushed onto the stack into a new Top of Stack register. The source operand (x) must be in the range $|x| \leq 2^{63}$.

Status:

Result of Instruction		C3	C2	C1[1]	C0
Incomplete Reduction of (x):		U	1	0	U
Normal completion:		U	0	0	U
Register Error:	Dest Reg Full	U	U	1	U
	Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN[2]	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN[2]	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	Result
+0	y=+0
-0	y=-0
+∞	Invalid Op.
-∞	Invalid Op.

Notes:

1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

2. The QNaN replaces both (x) and (y).

FRNDINT Round to Integer FRNDINT

Syntax: **FRNDINT**

Forms: **FRNDINT**

Operands:

Inst	Source Operand	Encoding			Cycles
FRNDINT	Top of Stack	$D9	11 111 100		6

Operation: The contents of the Top of Stack are rounded to an integer. The rounding operation to be performed is specified by the RC mode bits of the control word.

Status:

Result of Instruction		C3	C2	C1	C0
Normal Execution:	Round Down	U	U	0	U
	Round Up	U	U	1	U
Register Error:	Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op[1]	Masked	QNaN	0	-	-	-	-	-	1
	Unmasked	Unaltered	0	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

TOS	Result
+0	+0
-0	-0
+∞	+∞
-∞	-∞

Notes: 1. If TOS contains an unsupported, the Invalid Exception is caused.

FRSTOR Load CX-83S87 Environment & Registers FRSTOR

Syntax: **FRSTOR <DST>**

Forms: **FRSTOR <Memory>**

Operands:

Inst	Source Operand		Encoding		Cycles
FRSTOR	94 or 108 Bytes	SDD	MD 100 R/M	SIB,DISP	135

Operation: The CX-83S87 "Environment" and the CX-83S87 registers are loaded from the memory location specified. The "Environment" consists of the Mode Control Word, the Status Register, and the Tag Word which are loaded into the CX-83S87. The "Environment" also includes the CX-83S87 Instruction Pointer and the CX-83S87 Data Pointer which are loaded into 80386 CPU registers during execution of this instruction.

The format of the data structure saved in memory is dependent on the operating mode of the 80386 CPU and the operand size in effect. See FSAVE for complete details.

Restoring an "Environment" that contains a Status Register field with an exception indicated and an MCR with that exception enabled causes an ERROR# trap sequence when the next WAIT or exception checking CX-83S87 instruction is executed.

The ERROR# signal is unconditionally de-asserted while the "Environment" and register data is loaded. If the newly loaded "Environment" calls for an exception trap, ERROR# will be asserted upon completion of all data transfers. A subsequent WAIT or exception checking instruction will execute a trap sequence.

The CX-83S87 registers are loaded from the 80 consecutive byte memory locations following the "Environment" regardless of the 80386 mode or data size in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Loaded from Memory:	M	M	M	M

Exceptions:

Type	Mode	Result	P	U	O	Z	D	I	S
Loaded	N/A	N/A	M	M	M	M	M	M	M

Zero/Infinity: None.

Notes: None.

FSAVE — Save CX-83S87 Environment & Registers — FSAVE

Syntax: **FSAVE <DST>**

Forms: **FSAVE <Memory>**

Operands:

Inst	Dest Operand		Encoding		Cycles
FSAVE	94 or 108 Bytes	SDD	MD 110 R/M	SIB,DISP	142

Operation: The CX-83S87 "Environment" and the CX-83S87 registers are saved to the memory location specified. The "Environment" consists of the Mode Control Word, the Status Register, and the Tag Word from the CX-83S87 and the CX-83S87 Instruction Pointer and CX-83S87 Data Pointer from the 80386 CPU internal registers.

The CX-83S87 is set to its condition following an FINIT. The CX-83S87 Mode Control Register is set to $037F_{16}$, the CX-83S87 Status Register is reset to $0000, and all data registers are marked empty (Tag word=$FFFF).

This action sets Rounding control to "round to nearest or even" (RC=00), Precision control to 64-bit extended precision (PC=11), and the Top of Stack register number to zero (SSS=000). All exceptions are cleared (=0), all condition codes are cleared (C_0-C_3=0), and all exceptions are masked (=1).

This instruction differs from a hardware reset by setting MCR bit 0 (Invalid Exception Mask) to 1 and Status Register bits 7 & 0 (Error and Invalid Exception) to 0.

The format of the "Environment" data structure is dependent on the operating mode of the 80386 CPU and the operand size in effect. See below for complete details. The CX-83S87 data registers are saved to consecutive memory locations following the "Environment" block regardless of the 80386 mode or data size in effect.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	0	0	0	0

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	0	0	0	0	0	0	0

Zero/Infinity: None.

Notes: None.

FSAVE Save CX-83S87 Environment & Registers FSAVE

32-bit Protected Mode:

31	15	0
Reserved	Mode Control Word	00
Reserved	Status Word	04
Reserved	Tag Word	08
Instruction Pointer Offset		0C
00000 Opcode(10:0)	Code Segment Selector	10
Data Operand Offset		14
Reserved	Operand Seg Selector	18
R0 Significand(31:0)		1C
R0 Significand(63:32)		20
R1 Significand(15:0)	S R0 Exponent	24
R1 Significand(47:16)		28
S R1 Exponent	R1 Significand(63:48)	2C
R2 Significand(31:0)		30
R2 Significand(63:32)		34
R3 Significand(15:0)	S R2 Exponent	38
R3 Significand(47:16)		3C
S R3 Exponent	R3 Significand(63:48)	40
R4 Significand(31:0)		44
R4 Significand(63:32)		48
R5 Significand(15:0)	S R4 Exponent	4C
R5 Significand(47:16)		50
S R5 Exponent	R5 Significand(63:48)	54
R6 Significand(31:0)		58
R6 Significand(63:32)		5C
R7 Significand(15:0)	S R6 Exponent	60
R7 Significand(47:16)		64
S R7 Exponent	R7 Significand(63:48)	68

FSAVE Save CX-83S87 Environment & Registers FSAVE

32-bit Real Mode:

31		15		0	
Reserved			Mode Control Word		00
Reserved			Status Word		04
Reserved			Tag Word		08
Reserved			Instruction Ptr(15:0)		0C
0000	Instruction Ptr(31:16)		0	Opcode(10:0)	10
Reserved			Operand Ptr(15:0)		14
0000	Operand Ptr(31:16)		0000	0000	18
R0 Significand(31:0)					1C
R0 Significand(63:32)					20
R1 Significand(15:0)		S	R0 Exponent		24
R1 Significand(47:16)					28
S	R1 Exponent		R1 Significand(63:48)		2C
R2 Significand(31:0)					30
R2 Significand(63:32)					34
R3 Significand(15:0)		S	R2 Exponent		38
R3 Significand(47:16)					3C
S	R3 Exponent		R3 Significand(63:48)		40
R4 Significand(31:0)					44
R4 Significand(63:32)					48
R5 Significand(15:0)		S	R4 Exponent		4C
R5 Significand(47:16)					50
S	R5 Exponent		R5 Significand(63:48)		54
R6 Significand(31:0)					58
R6 Significand(63:32)					5C
R7 Significand(15:0)		S	R6 Exponent		60
R7 Significand(47:16)					64
S	R7 Exponent		R7 Significand(63:48)		68

FSAVE Save CX-83S87 Environment & Registers FSAVE

16-bit Protected Mode:

15		0
Mode Control Word		00
Status Word		02
Tag Word		04
Instruction Ptr Offset		06
Code Segment Selector		08
Data Operand Offset		0A
Operand Seg Selector		0C
R0 Significand(15:0)		0E
R0 Significand(31:16)		10
R0 Significand(47:32)		12
R0 Significand(63:48)		14
S	R0 Exponent	16
Repeat for R1		18
Repeat for R2		22
Repeat for R3		2C
Repeat for R4		36
Repeat for R5		40
Repeat for R6		4A
R7 Significand(15:0)		54
R7 Significand(31:16)		56
R7 Significand(47:32)		58
R7 Significand(63:48)		5A
S	R7 Exponent	5C

FSAVE Save CX-83S87 Environment & Registers FSAVE

16-bit Real Mode:

15				0	
Mode Control Word					00
Status Word					02
Tag Word					04
Instruction Ptr(15:0)					06
IP(19:16)	0	Opcode(10:0)			08
Operand Ptr(15:0)					0A
DP(19:16)	0000	0000	0000		0C
R0 Significand(15:0)					0E
R0 Significand(31:16)					10
R0 Significand(47:32)					12
R0 Significand(63:48)					14
S	R0 Exponent				16
Repeat for R1					18
Repeat for R2					22
Repeat for R3					2C
Repeat for R4					36
Repeat for R5					40
Repeat for R6					4A
R7 Significand(15:0)					54
R7 Significand(31:16)					56
R7 Significand(47:32)					58
R7 Significand(63:48)					5A
S	R7 Exponent				5C

FSCALE Floating Multiply by 2^n FSCALE

Syntax: **FSCALE**

Forms: **FSCALE**

Operands:

Inst	Source Operand	Encoding			Cycles
FSCALE	Top of Stack	$D9	11 111 101		8

Operation: The contents of the Top of Stack (x) are multiplied by 2^n where (n) is the contents of the next to Top of Stack. The result $y = x * 2^n$ is normalized and rounded according to the RC mode in effect. The result (y) replaces the contents of the Top of Stack. The value (n) is an integer determined by chopping actual contents of the next to Top of Stack toward zero.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale[2]	-	-	1	-	-	-	-
Overflow	Masked	R(∞)	-	-	-	1	-	-	-
	Unmasked	Round & Scale[2]	-	-	-	1	-	-	-
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	n	Result
+0	$-\infty$	+0
-0	$-\infty$	-0
0	$+\infty$	Inval. Op.
$+\infty$	$-\infty$	Inval. Op.
$-\infty$	$-\infty$	Inval. Op.
$+\infty$	$-\infty<n\leq+\infty$	$+\infty$
$-\infty$	$-\infty<n\leq+\infty$	$-\infty$
$x\neq0$	$+\infty$	$sgn(x)*\infty$
$x\neq0$	$-\infty$	$sgn(x)*0$

FSCALE Floating Multiply by 2^n FSCALE

Notes:

1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

2. In the event of "massive" over/underflow in which exponent adjustment by -/+ 24,576 does not yield a normal result, the instruction returns a signed ∞ or zero respectively.

FSIN

Function Evaluation: Sin(x).

FSIN

Syntax: **FSIN**

Forms: **FSIN**

Operands:

Inst	Source Operand	Encoding			Cycles
FSIN	Top of Stack	$D9	11 111 110		5-63

Operation: The Top of Stack contains the source operand (x). The instruction requires (x) in radians and returns (y) such that y= sin(x). The result (y) is normalized and rounded according to the RC mode in effect. The value (y) replaces the contents of the Top of Stack. The source operand (x) must be in the range $|x| \leq 2^{63}$.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Incomplete Reduction of (x):	U	1	0	U
Normal completion:	U	0	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	Result
+0	+0
-0	-0
+∞	Invalid Op.
-∞	Invalid Op.

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FSINCOS Function Eval: Sin(x) & Cos(x) FSINCOS

Syntax: **FSINCOS**

Forms: **FSINCOS**

Operands:

Inst	Source Operand		Encoding		Cycles
FSINCOS	Top of Stack	$D9	11 111 011		5-104

Operation: The Top of Stack contains the source operand (x). The instruction requires (x) in radians and returns (y) and (z) such that y= sin(x) and z= cos(x). The results (y) & (z) are normalized and rounded according to the RC mode in effect. The value (y) replaces the contents of the Top of Stack. The value (z) is pushed onto the stack into a new Top of Stack register. The source operand (x) must be in the range $|x| \leq 2^{63}$.

Status:

Result of Instruction		C3	C2	C1[1]	C0
Incomplete Reduction of (x):		U	1	0	U
Normal completion:		U	0	0	U
Register Error:	Dest Reg Full	U	U	1	U
	Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	Result
+0	y=+0;z=+1
-0	y=-0;z=+1
$+\infty$	Invalid Op.
$-\infty$	Invalid Op.

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FSQRT — Floating Point Square Root — FSQRT

Syntax: **FSQRT**

Forms: **FSQRT**

Operands:

Inst	Source Operand		Encoding		Cycles
FSQRT	Top of Stack	SD9	11 111 010		26

Operation: The contents of the Top of Stack (x) are replaced by $\sqrt{x}$. The result is normalized and rounded according to the RC mode in effect at the precision specified by the PC mode bits.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	Result
+0	+0
-0	-0
$-\infty \le x < -0$	Inval. Op.
$+\infty$	$+\infty$

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FST — Store CX-83587 Register — FST

Syntax: **FST(P) (<DST>)**

Forms: **FST(P) <Memory>**
FST(P) <Register>

Operands:

Inst	Dest Operand		Encoding		Cycles
FIST(P)	16-bit Integer	$DF	MD 01p R/M	SIB,DISP	11
FIST(P)	32-bit Integer	$DB	MD 01p R/M	SIB,DISP	13
FISTP	64-bit Integer	$DF	MD 111 R/M	SIB,DISP	15
FBSTP	18 dig BCD Int.	$DF	MD 110 R/M	SIB,DISP	65
FST(P)	32-bit Real	$D9	MD 01p R/M	SIB,DISP	11
FST(P)	64-bit Real	$DD	MD 01p R/M	SIB,DISP	15
FSTP	80-bit Real	$DB	MD 111 R/M	SIB,DISP	17
FST(P)	80-bit Register	$DD	11 01p REG		4

Operation: The source operand (x) is fetched from the Top of Stack and, if necessary, converted to the destination data format and rounded according to the RC mode in effect. The result is stored in the destination. When the "pop" form is used, the Top of Stack is popped upon completion.

The operand is rounded to the width of the destination according to the RC mode specified. If the destination type is 32-bit or 64-bit real and the operand is zero, ∞, or NaN, the significand and the exponent are chopped and transferred as-is.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow[2]	Masked	Rounded	-	-	1	-	-	-	-
	Unmasked	Trap/Abort	-	-	1	-	-	-	-
Overflow[2]	Masked	Rounded	-	-	-	1	-	-	-
	Unmasked	Trap/Abort	-	-	-	1	-	-	-
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op[3]	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Trap/Abort	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Trap/Abort	1	-	-	-	-	-	1

FST — Store CX-83S87 Register — FST

Zero/Infinity:

x	Result
Empty	Invalid Op.
NaN->Integer	Invalid Op.
∞->Integer	Invalid Op.
|x|>Int Range	Invalid Op.

Notes:

1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

2. Exception can only occur when the destination operand is 32 or 64 bit real format.

3. Storing into an integer or BCD destination when the operand is of greater magnitude than the destination format supports produces this exception.

FSTCW Store CX-83S87 Mode Control Register FSTCW

Syntax: **FSTCW <DST>**

Forms: **FSTCW <Memory>**

Operands:

Inst	Dest Operand	Encoding			Cycles
FSTCW	2 Bytes	SD9	MD 111 R/M	SIB,DISP	4

Operation: The contents of the CX-83S87 Mode Control Register are stored into the specified memory location.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: None.

FSTENV Load CX-83S87 Environment FSTENV

Syntax: **FSTENV <DST>**

Forms: **FSTENV <Memory>**

Operands:

Inst	Source Operand	Encoding			Cycles
FSTENV	14 or 28 Bytes	SD9	MD 110 R/M	SIB,DISP	17

Operation: The CX-83S87 "Environment" is saved to the memory location specified. The "Environment" consists of the Mode Control Word, the Status Register, and the Tag Word which are saved from the CX-83S87. The "Environment" also includes the CX-83S87 Instruction Pointer and the CX-83S87 Data Pointer which are saved from 80386 CPU registers during execution of this instruction.

The FSTENV instruction sets all the exception mask bits of the MCR to 1 thereby masking all exceptions. This causes the ERROR# signal to be de-asserted.

The FSTENV instruction is designed for use in exception handlers to help analyze the exception condition. The format of the "Environment" data structure is dependent on the operating mode of the 80386 CPU and the operand size in effect.

32-bit Protected Mode:

31	15	0
Reserved		Mode Control Word
Reserved		Status Word
Reserved		Tag Word
Instruction Pointer Offset		
00000	Opcode(10:0)	Code Segment Selector
Data Operand Offset		
Reserved		Operand Seg Selector

32-bit Real Mode:

31	15	0
Reserved		Mode Control Word
Reserved		Status Word
Reserved		Tag Word
Reserved		Instruction Ptr(15:0)
0000	Instruction Ptr(31:16)	0 Opcode(10:0)
Reserved		Operand Ptr(15:0)
0000	Operand Ptr(31:16)	0000 0000 0000

FSTENV Store CX-83S87 Environment FSTENV

16-bit Protected Mode:

15 0
Mode Control Word
Status Word
Tag Word
Instruction Ptr Offset
Code Segment Selector
Data Operand Offset
Operand Seg Selector

16-bit Real Mode:

15			0
Mode Control Word			
Status Word			
Tag Word			
Instruction Ptr(15:0)			
IP(19:16)	0	Opcode(10:0)	
Operand Ptr(15:0)			
DP(19:16)	0000	0000	0000

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: None.

FSTSW — Store CX-83S87 Status Register — FSTSW

Syntax: **FSTSW** **<DST>**

Forms: **FSTSW** **<Memory>**

Operands:

Inst	Dest Operand		Encoding		Cycles
FSTSW	2 Bytes	$DD	MD 111 R/M	SIB,DISP	4

Operation: The contents of the CX-83S87 Status Register are stored into the specified memory location.

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: None.

FSTSWAX Store CX-83S87 Status to AX FSTSWAX

Syntax: **FSTSWAX**

Forms: **FSTSWAX**

Operands:

Inst	Dest Operand		Encoding		Cycles
FSTSWAX	80386 AX Reg	SDF	11 100 000		4

Operation: The contents of the CX-83S87 Status Register are stored into the 80386 AX register. The contents of the AX register may then be transferred to the 80386 flags with the SAHF instruction. The following table shows how the 80386 conditional branch instructions can be used to decode CX-83S87 status reflecting the results of FCOM execution:

80386 Branch	Result of FCOM	C3	C2	C1	C0
JA	DST > SRC	0	0	0	0
JB	DST < SRC	0	0	0	1
JE	DST = SRC	1	0	0	0
JP	Unordered	1	1	0	1

Status:

Result of Instruction	C3	C2	C1	C0
Unconditional:	U	U	U	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: This instruction transfers the contents of the CX-83S87 Status Register to the AX register before 80386 instruction execution may proceed.

FSUB

Floating Point Subtract

FSUB

Syntax: **FSUB(R)(P) ((<DST>,)<SRC>)**

Forms:

FSUB(R) <TOS>,<Memory>
FSUB(R) <TOS>,<Reg>
FSUB(R)(P) <Reg>,<TOS>

Operands:

Inst	Source Operand	Encoding			Cycles
FISUB	16-bit Integer	$DE	MD 10r R/M	SIB,DISP	13
FISUB	32-bit Integer	$DA	MD 10r R/M	SIB,DISP	15
FSUB	32-bit Real	$D8	MD 10r R/M	SIB,DISP	15
FSUB	64-bit Real	$DC	MD 10r R/M	SIB,DISP	19
FSUB	80-bit Register	$D8	11 10r REG		6
FSUB	Top of Stack	$DC	11 101 REG		6
FSUBR	Top of Stack	$DC	11 100 REG		6
FSUBP	80-bit Register	$DE	11 101 REG		6
FSUBRP	80-bit Register	$DE	11 100 REG		6

Operation:

The source and destination operands are fetched. The source is converted to extended precision format if necessary. The source operand is subtracted from the destination and the result is normalized and rounded according to the RC mode in effect at the precision specified by the PC mode bits. The result is stored in the destination register. When the "pop" form is used, the top of stack is popped.

The "reverse" form causes the destination operand to be subtracted from the source operand.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	Masked	R(∞)	-	-	-	1	-	-	-
	Unmasked	Round & Scale	-	-	-	1	-	-	-
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FSUB Floating Point Subtract FSUB

Zero/Infinity:

OP1	OP2	Result	OP1	OP2	Result
+0	+0	R(0)	+∞	+∞	Inv. Op.
-0	-0	R(0)	-∞	-∞	Inv. Op.
+0	-0	+0	+∞	-∞	+∞
-0	+0	-0	-∞	+∞	-∞
+X	+X	R(0)	+∞	X	+∞
-X	-X	R(0)	-∞	X	-∞
			X	-∞	+∞
			X	+∞	-∞

Notes: After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FTST Test Top of Stack FTST

Syntax: **FTST**

Forms: **FTST**

Operands:

Inst	Dest Operand		Encoding		Cycles
FTST	Top of Stack	$D9	11 100 100		6

Operation: The contents of the Top of Stack are compared to zero. The condition code results are the same as those produced by the FCOM instruction with the exception of detecting equality to -0. The Top of Stack is the destination and the constant zero is the source.

The result "unordered" is produced when the operand is NaN, unsupported or when Stack Fault occurs.

Status:

Result of Instruction		C3	C2	C1	C0
Normal Execution:	DST > 0	0	0	0	0
	DST < 0	0	0	0	1
	DST = +0	1	0	0	0
	DST = -0	1	0	1	0
	Unordered	1	1	0	1
Register Error:	Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	P	U	O	Z	D	I	S

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Denorm Used	-	-	-	-	-	1	-
Invalid Op[1]	Masked	Unaltered	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	Unaltered	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

DST	Result
+0	=0
-0	=0
+∞	>0
-∞	<0

Notes: 1. QNAN operands produce the Invalid Exception in this instruction.

FUCOM Unordered Compare FUCOM

Syntax: **FUCOM(P) ((<DST>,)<SRC>)**

Forms: **FUCOM(P)(P) <TOS>,<Reg>**

Operands:

Inst	Source Operand		Encoding		Cycles
FUCOM	80-bit Register	SDD	11 100 REG		4
FUCOMP	80-bit Register	SDD	11 101 REG		4
FUCOMPP	80-bit Register	SDA	11 101 001		4

Operation: The source operand is fetched and subtracted from the destination (Top of Stack) and the condition codes are set according to the result. When the "P" form is used, the Top of Stack is popped. The "PP" form compares the Top of Stack and the next to Top of Stack and causes two "pop" operations upon completion.

This instruction has the same effect as the FCOM instruction except that it does not cause the Invalid Exception when one of the operands is a QNaN.

The result "unordered" is produced when the operands are NaNs, unsupported or when Stack Fault occurs.

Status:

Result of Instruction		C3	C2	C1	C0
Normal Execution:	DST > SRC	0	0	0	0
	DST < SRC	0	0	0	1
	DST = SRC	1	0	0	0
	Unordered	1	1	0	1
Register Error:	Source Reg Empty	1	1	0	1

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Denorm Used	-	-	-	-	-	1	-
Invalid Op	Masked	Unordered	-	-	-	-	-	1	-
	Unmasked	Unaltered	-	-	-	-	-	1	-
Register Error:	Masked	Unordered	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FUCOM Unordered Compare FUCOM

Zero/Infinity:

DST -	SRC	Result	DST -	SRC	Result
+0	+0	=	+∞	+∞	=
-0	-0	=	-∞	-∞	=
+0	-0	=	+∞	-∞	DST>SRC
-0	+0	=	-∞	+∞	DST<SRC
+0	+X	DST<SRC	+∞	X	DST>SRC
-0	+X	DST<SRC	-∞	X	DST<SRC
+0	-X	DST>SRC	X	-∞	DST>SRC
-0	-X	DST>SRC	X	+∞	DST<SRC

Notes: None.

FXAM — Report Class of Operand — FXAM

Syntax: **FXAM**

Forms: **FXAM**

Operands:

Inst	Source Operand		Encoding		Cycles
FXAM	80-bit Register	SD9	11 100 101		3

Operation: The Top of Stack is always the source operand. The Top of Stack is examined and condition codes are set according to its class as specified below.

Status: The "C1" Status bit indicates the sign of the Top of Stack operand: "0"=positive; "1"=negative.

Contents of TOS	C3	C2	C0
Unsupported	0	0	0
NaN	0	0	1
Normal	0	1	0
Infinity	0	1	1
Zero	1	0	0
Empty[1]	1	0	1
Denormal	1	1	0

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
None	N/A	N/A	-	-	-	-	-	-	-

Zero/Infinity: None.

Notes: 1. If the stack is empty, the result is "Empty" and the sign is undefined. No exception is generated.

FXCH — Exchange Register with TOS — FXCH

Syntax: **FXCH (SRC)**

Forms: **FXCH <Reg>**

Operands:

Inst	Source Operand		Encoding		Cycles
FXCH	80-bit Register	$D9	11 001 REG		4

Operation: The contents of the Top of Stack and the source register are exchanged.

Status:

Result of Instruction	C3	C2	C1	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	1	1	0	1

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	N/A	N/A							
Invalid Op	N/A	N/A							
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

Operand	Result
Empty	Invalid Ex.

Notes: None.

FXTRACT Extract Exponent FXTRACT

Syntax: **FXTRACT**

Forms: **FXTRACT**

Operands:

Inst	Source Operand		Encoding		Cycles
FXTRACT	Top of Stack	$D9	11 110 100		7

Operation: The Top of Stack contains the source operand (x). The exponent field of (x) is converted to an 80-bit extended precision real number (y) and pushed on the stack, i.e. y=INT(Log_2(x)) is pushed. The original operand (x) is modified by having its exponent set to zero, i.e. after execution $1.0 \le |x| < 2.0$. The sign of (x) is preserved.

Status:

Result of Instruction		C3	C2	C1	C0
Normal Execution:		U	U	0	U
Register Error:	Dest Reg Full	U	U	1	U
	Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	N/A	N/A							
Underflow	N/A	N/A							
Overflow	N/A	N/A							
Div by Zero	Masked	TOS=0,ST1=-∞	-	-	-	-	1	-	-
	Unmasked	Trap/Abort	-	-	-	-	1	-	-
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

x	Result
+0	y = +0; x = -∞; Zero Div. Ex.
-0	y = -0; x = -∞; Zero Div. Ex.
-∞	y = -∞; x = +∞
+∞	y = +∞; x = +∞

Notes: None.

FYL2X — Function Evaluation: $y*Log_2(x)$. — FYL2X

Syntax: **FYL2X**

Forms: **FYL2X**

Operands:

Inst	Source Operand	Encoding			Cycles
FYL2X	Top of Stack	$D9	11 110 001		6-93

Operation: The Top of Stack contains the source operand (x). The next to Top of Stack contains the source operand (y). The function $z=y*log_2(x)$ is evaluated and the result is normalized and rounded according to the RC mode in effect. The stack is popped and the result (z) is replaces (y) as the new Top of Stack. The source operand (x) must be in the range $0 < x \leq +\infty$.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	Masked	R(∞)	-	-	-	1	-	-	-
	Unmasked	Round & Scale	-	-	-	1	-	-	-
Div by Zero	Masked	TOS=-∞	-	-	-	-	1	-	-
	Unmasked	Trap/Abort	-	-	-	-	1	-	-
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

Zero/Infinity:

y	x	Result
--	$x < 0$	Invalid Op.
$y \neq 0$	$x = 0$	Zero Div. Ex.
$y = 0$	$x = 0$	Invalid Op.
$y = \infty$	$x = 1$	Invalid Op.
$y = \infty$	$x > 1$	y
$y = \infty$	$0 < x < 1$	-y
$y > +0$	$x = \infty$	$+\infty$
$y < -0$	$x = \infty$	$-\infty$
$y = 0$	$x = \infty$	Invalid Op.

FYL2X — Function Evaluation: $y*Log_2(x)$. — FYL2X

Notes: 1. After a Precision Exception the "C1" status bit indicates whether rounding was away from zero.

FYL2XP1 Function Eval: y*Log2(x+1). FYL2XP1

Syntax: **FYL2XP1**

Forms: **FYL2XP1**

Operands:

Inst	Source Operand		Encoding		Cycles
FYL2XP1	Top of Stack	$D9	11 111 001		6-90

Operation: The Top of Stack contains the source operand (x). The next to Top of Stack contains the source operand (y). The function $z=y*\log_2(x+1)$ is evaluated and the result is normalized and rounded according to the RC mode in effect. The stack is popped and the result (z) replaces (y) as the new Top of Stack. The operand (x) must be in the range

$$\frac{\sqrt{2}}{2}-1 < x < 1-\frac{\sqrt{2}}{2}.$$

Use FYL2XP1 to calculate log2(x) when |x| is close to 1. FYL2XP1 provides greater accuracy than FYL2X in this case. The magnitude of the input argument must be near zero, so be sure to subtract one from (x) before using FYL2XP1.

Status:

Result of Instruction	C3	C2	C1[1]	C0
Normal Execution:	U	U	0	U
Register Error: Source Reg Empty	U	U	0	U

Exceptions:

Type	Mode	Result	S	P	U	O	Z	D	I
Precision	Masked	Rounded	-	1	-	-	-	-	-
	Unmasked	Rounded	-	1	-	-	-	-	-
Underflow	Masked	Denorm/Zero	-	1	1	-	-	-	-
	Unmasked	Round & Scale	-	-	1	-	-	-	-
Overflow	N/A	N/A							
Div by Zero	N/A	N/A							
Denormal	Masked	Denorm Used	-	-	-	-	-	1	-
	Unmasked	Trap/Abort	-	-	-	-	-	1	-
Invalid Op	Masked	QNaN	-	-	-	-	-	-	1
	Unmasked	Unaltered	-	-	-	-	-	-	1
Register Error:	Masked	QNaN	1	-	-	-	-	-	1
	Unmasked	Unaltered	1	-	-	-	-	-	1

FYL2XP1 Function Eval: $y*Log_2(x+1)$. FYL2XP1

Zero/Infinity:

y	x	Result
$y \geq +0$	$x = -0$	-0
$y \geq +0$	$x = +0$	+0
$y \leq -0$	$x = -0$	+0
$y < -0$	$x = +0$	-0
$y = \infty$	$x = 0$	Invalid Op.
$y = \infty$	$x > 0$	y
$y = \infty$	$-1 < x < 0$	-y
$y > +0$	$x = \infty$	$+\infty$
$y < -0$	$x = \infty$	$-\infty$
$y = 0$	$x = \infty$	Invalid Op.

Notes: 1. After a Precisian Exception the "C1" status bit indicates whether rounding was away from zero.

4.13 Instruction Execution Times

This section presents two values for CX-83S87 instruction execution times. The Basic Execution Time (BET) is the number of clock cycles that the CX-83S87 underlying architecture will execute a given instruction and are the times given in the individual instruction descriptions. The Intel System Time (IST) is the time required by an CX-83S87/Intel 386SX combination which includes the 386SX overhead of instruction setup and operand transfer. The difference between the BET and the IST is due entirely to 386SX PEREQ & BUSY# protocol overhead. In both cases, no wait states are included and no allowance for DMA overhead is included.

IST values are obtained by submitting instruction streams consisting of several repetitions of a given instruction to the 386SX/CX-83S87 pair and the 386SX/387SX pair and measuring the instruction-to-instruction delay time. This technique provides performance figures based upon observed in-system throughput. The IST observed in a stream of dissimilar instructions will vary substantially based on the prior and following instructions. This occurs because of the different activities in the 386SX during the various instructions and because the execution times of some instructions have a dependency on the value of the input data. When measuring IST times only valid input arguments where used and no exceptions were generated except for the precision exception. For the transcendental functions, the restrictions listed below were placed on the input arguments.

All forward trigonometric functions	$0 < ST(0) < \pi/4$.	
F2XM1	$0 < ST(0) < 0.5$.	
FPATAN	$1 < ST(0) < 2$	& $0 < ST(1) < \pi/32$.
FYL2X	$0 < ST(0) < SQRT(2)$	& $1 < ST(1) < 2$.
FYL2XP1	$0 < ST(0) < SQRT(2)$	& $1 < ST(1) < 2$.

Mnemonic	Result	Operation	83S87-BET	83S87-IST	'387SX IST
F2XM1	TOS ←	$2^{TOS}-1$	14-66	63	339
FABS	TOS ←	|TOS|	4	11	29
FADD	ST(i) ←	ST(i)+TOS	6	15	34
FADD	TOS ←	TOS+ST(i)	6	15	34
FADD	TOS ←	TOS+M.DR	19	24	37
FADD	TOS ←	TOS+M.SR	15	19	32
FADDP	ST(i-1) ←	ST(i)+TOS	6	15	34
FIADD	TOS ←	TOS+M.SI	15	19	54
FIADD	TOS ←	TOS+M.WI	13	26	76
FCHS	TOS ←	TOS*(-1)	4	11	34
FCLEX		Clear Exceptions	4	15	15
FCOM	CC ←	TOS-ST(i)	4	11	29
FCOM	CC ←	TOS-M.DR	17	24	37
FCOM	CC ←	TOS-M.SR	13	19	30
FCOMP	CC ←	TOS-ST(i)	4	11	29
FCOMP	CC ←	TOS-M.DR	17	24	37
FCOMP	CC ←	TOS-M.SR	13	19	28
FCOMPP	CC ←	TOS-ST(1)	4	11	34
FICOM	CC ←	TOS-M.SI	13	19	51
FICOM	CC ←	TOS-M.WI	11	26	70
FICOMP	CC ←	TOS-M.SI	13	19	53
FICOMP	CC ←	TOS-M.WI	11	26	70
FCOS	TOS ←	COS(TOS)	5-97	87	569
FDECSTP	SP ←	SP-1	5	15	29
FDIV	ST(i) ←	ST(i)/TOS	14-25	31	99
FDIV	TOS ←	TOS/ST(i)	13-24	27	94
FDIV	TOS ←	TOS/M.DR	25-36	34	103
FDIV	TOS ←	TOS/M.SR	22-33	29	95
FDIVP	ST(i-1) ←	ST(i)/TOS	14-25	19	99
FDIVR	TOS ←	ST(i)/TOS	13-24	27	94
FDIVR	ST(i) ←	TOS/ST(i)	14-25	19	99
FDIVR	TOS ←	M.DR/TOS	26-37	34	103
FDIVR	TOS ←	M.SR/TOS	23-34	29	95
FDIVRP	ST(i-1) ←	TOS/ST(i)	13-24	27	99
FIDIV	TOS ←	TOS/M.SI	22-33	29	120
FIDIV	TOS ←	TOS/M.WI	20-31	41	136
FIDIVR	TOS ←	M.SI/TOS	23-34	29	120
FIDIVR	TOS ←	M.WI/TOS	21-32	41	142
FFREE	TAG(i) ←	Empty	5	15	24
FINCSTP	SP ←	SP+1	5	15	29
FINIT	---	Initialize	5	15	15
FLD	TOS ←	ST(i)	4	11	19
FLD	TOS ←	M.DR	16	26	34
FLD	TOS ←	M.SR	12	21	25
FLD	TOS ←	M.XR	15	38	52
FBLD	TOS ←	M.BCD	36	52	274
FILD	TOS ←	M.LI	13	25	58
FILD	TOS ←	M.SI	9	20	50
FILD	TOS ←	M.WI	7	21	58

Mnemonic	Result	Operation	83S87-BET	83S87-IST	'387SX IST
FLD1	TOS ←	One	6	11	29
FLDCW	Ctl Word ←	Memory	4	20	32
FLDENV	Env Regs ←	Memory	22	83	109
FLDL2E	TOS ←	$Log_2(e)$	7-8	11	44
FLDL2T	TOS ←	$Log_2(10)$	8-9	11	44
FLDLG2	TOS ←	$Log_{10}(2)$	8-9	15	44
FLDLN2	TOS ←	$Log_e(2)$	6-7	11	44
FLDPI	TOS ←	π	6-7	11	44
FLDZ	TOS ←	Zero	7	11	29
FMUL	ST(i) ←	ST(i)*TOS	10	19	59
FMUL	TOS ←	TOS*ST(i)	10	19	54
FMULP	ST(i-1) ←	ST(i)*TOS	10	19	59
FMUL	TOS ←	TOS*M.DR	22	25	45
FMUL	TOS ←	TOS*M.SR	18	20	35
FIMUL	TOS ←	TOS*M.SI	18	20	60
FIMUL	TOS ←	TOS*M.WI	16	31	76
FNOP	---	No Operation	4	11	19
FPATAN	TOS ←	$ATAN(\frac{ST(1)}{TOS})$	90-127	83	434
FPREM	TOS ←	$Rem(\frac{TOS}{ST(1)})$	48	51	119
FPREM1	TOS ←	$Rem(\frac{TOS}{ST(1)})$	49	51	144
FPTAN	TOS:ST(1) ←	1; TAN(TOS)	5-82	75	354
FRNDINT	TOS ←	Round(TOS)	6	15	74
FRSTOR	---	Restore state	135	325	458
FSAVE	---	Save state	142	366	557
FSCALE	TOS ←	$TOS*2^{(ST(1))}$	8	15	79
FSIN	TOS ←	SIN(TOS)	5-63	63	509
FSINCOS	TOS:ST(1) ←	COS:SIN(TOS)	5-104	95	614
FSQRT	TOS ←	$\sqrt{TOS}$	26	31	129
FST	ST(i) ←	TOS	4	11	19
FST	M.DR ←	TOS	15	34	54
FST	M.SR ←	TOS	11	28	40
FSTP	ST(i-1) ←	TOS	4	11	24
FSTP	M.DR ←	TOS	15	34	54
FSTP	M.SR ←	TOS	11	28	40
FSTP	M.XR ←	TOS	17	37	65
FBSTP	M.BCD ←	TOS	65	74	536
FIST	M.SI ←	TOS	13	27	84
FIST	M.WI ←	TOS	11	26	87
FISTP	M.LI ←	TOS	15	32	92
FISTP	M.SI ←	TOS	13	27	84
FISTP	M.WI ←	TOS	11	26	87

Mnemonic	Result	Operation	83S87-BET	83S87-IST	'387SX IST
FSTCW	Memory ←	Control word.	4	14	15
FSTENV	Memory ←	Ctl,Status,IP,DP.	17	92	188
FSTSW	Memory ←	Status	4	14	15
FSTSWAX	AX ←	Status	4	10	11
FSUB	ST(i) ←	ST(i)-TOS	6	15	34
FSUB	TOS ←	TOS-ST(i)	6	15	29
FSUBP	ST(i-1) ←	ST(i)-TOS	6	15	34
FSUB	TOS ←	TOS-M.DR	19	25	37
FSUB	TOS ←	TOS-M.SR	15	20	29
FISUB	TOS ←	TOS-M.WI	13	26	76
FISUB	TOS ←	TOS-M.SI	15	20	54
FSUBR	TOS ←	ST(i)-TOS	6	15	29
FSUBR	ST(i) ←	TOS-ST(i)	6	15	34
FSUBRP	ST(i-1) ←	TOS-ST(i)	6	15	34
FSUBR	TOS ←	M.DR-TOS	19	25	37
FSUBR	TOS ←	M.SR-TOS	15	20	29
FISUBR	TOS ←	M.WI-TOS	13	26	76
FISUBR	TOS ←	M.SI-TOS	15	20	53
FTST	CC ←	TOS-0.0	6	15	34
FUCOM	CC ←	TOS-ST(i)	4	11	29
FUCOMP	CC ←	TOS-ST(i)	4	11	29
FUCOMPP	CC ←	TOS-ST(1)	4	11	34
FXAM	CC ←	Class of TOS	3	11	39
FXCH	TOS ↔	ST(i) Exchange	4	11	24
FXTRACT	TOS;ST(1) ←	Signif; Exponent	7	15	74
FYL2X	TOS ←	ST(1)*Log_2(TOS)	6-93	87	519
FYL2XP1	TOS ←	ST(1)*Log_2(1+TOS)	6-90	79	544

5. Host Processor Interface

The CX-83S87 processor interfaces directly to the Intel 386SX microprocessor bus following the standard numeric processor extension protocol. The interface consists of a 16-bit bidirectional data bus, bus control signals, CX-83S87 status signals, and power connections.

In the following discussion of the hardware interface, the '#' symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no '#' is present after the name, the signal is active at a high voltage level.

5.1 Signal Description

Each paragraph identifies the CX-83S87 signals by name, provides the signal function, the active state, signal direction, and reference signal for each. The signal discussions are arranged in alphabetical order to assist locating the desired topic.

CPUCLK2 (386SX clock input)

This signal is an input and is used to synchronize the CX-83S87 bus interface to the processor bus. Most of the interface signals are sampled on the rising edge of this clock or driven relative to the rising edge of this clock. This signal also supplies the clock for the internal processor circuitry. The CPUCLK2 signal is divided by two to obtain the basic internal clock rate of the CX-83S87. This input accepts MOS level inputs. The input to this pin must be the same signal that drives the 386SX processor.

387SXCLK2 (Ignored)

The CX-83S87 ignores this input signal to maintain socket compatibility with the 387SX.

ADS# (Address Strobe)

The ADS# signal is an input to the CX-83S87 and indicates that the information on NPS1#, NPS2, W/R#, and CMD0# is valid and should be sampled at the rising edge of CPUCLK2. (See Table 5.1-1 for current bus cycle definition based upon NPS1#, NPS2, W/R#, and CMD0# inputs). The setup and hold times are referenced to CPUCLK2 This signal is normally connected to the 386SX ADS# signal.

BUSY# (Busy Status)

The BUSY# signal is an output from the CX-83S87 and its active condition indicates that the floating point processor is busy. The signal is referenced to CPUCLK2. This signal is normally connected to the BUSY# input of the 386SX.

CKM (Ignored)

The CX-83S87 ignores this input signal to maintain socket compatibility with the 387SX.

CMD0# (Select Command Port)

The CMD0# input indicates that the current CX-83S87 bus cycle is accessing the command port. When inactive the current CX-83S87 bus cycle is accessing the data port. This signal is sampled with the rising edge of CPUCLK2 when ADS#, NPS1#, NPS2, and W/R# are active. (See Table 5.1-1 for CMD0#'s use in defining the current bus cycle). CMD0# is normally connected to the 386SX A2 output. The setup and hold times are referenced to CPUCLK2.

D15-D0 (Data Bus)

These bidirectional signals are used to transfer information between the host processor and the CX-83S87. D15 is the most significant bit of a transfer. These signals are normally connected to the D15-D0 pins of the 386SX. The timing of these lines is referenced to CPUCLK2.

ERROR# (Error Status)

The ERROR# output normally reflects the status of the ES bit in the status register. Immediately after reset, it identifies the coprocessor as an 387SX compatible device by being active. If ERROR# is going to be asserted during an instruction it will be asserted prior to BUSY# being made inactive.

NPS1# (Numeric Processor Select)

The NPS1# input is used as a select signal to the CX-83S87. This signal is sampled simultaneously with ADS# and NPS2 to determine if the current bus cycle is intended for the CX-83S87. (See Table 5.1-1 for NPS1#'s use in defining the current bus cycle). NPS1# is normally connected to the M/IO# output of the 386SX. Setup and hold times are referenced to the rising edge of CPUCLK2.

NPS2(Numeric Processor Select)

The NPS2 input is used as a select signal to the CX-83S87. This signal is sampled simultaneously with ADS# and NPS1# to determine if the current bus cycle is intended for the CX-83S87. (See Table 5.1-1 for NPS2's use in defining the current bus cycle). NPS2 is normally connected to the A31 output of the 386SX. Setup and hold times are referenced to the rising edge of CPUCLK2.

PEREQ (Processor Extension Request)

The PEREQ signal is an output from the CX-83S87 to the 386SX processor. When active, this signal indicates that the CX-83S87 is ready for a data transfer with the host processor. When all data transfers have been completed the signal will go inactive. PEREQ will not be active when BUSY# is inactive. This signal is normally connected to the 386SX PEREQ input.

READY# (Bus Ready)

The READY# input is sampled on the rising edge of CPUCLK2. The active state of READY# indicates that the current bus cycle is being concluded. The CX-83S87 bus interface uses READY# and ADS# to track the 386SX bus cycles and remain synchronized with the 386SX operation. This signal is normally connected to the some signal that drives the READY# input of the 386SX.

READYO# (CX-83S87 Ready)

The READYO# output is activated when the CX-83S87 is ready to conclude a bus cycle. READYO# is referenced to the rising edge of CPUCLK2. This signal is normally connected to the READY# input of the 386SX or the READY# generator for the system. READYO# in the CX-83S87 operates independently of BUSY# and PEREQ. Therefore, communication protocols other than the standard 386SX Numeric Processor Extension protocol can be more easily implemented.

RESETIN (System Reset)

The RESETIN input performs a total reset of the CX-83S87. It must remain active for a minimum of 20 input clock periods. The active to inactive transition of RESETIN must be synchronous with CPUCLK2 to match internal clock phases with that of the 386SX. After RESETIN goes inactive, READYO#, BUSY#, and PEREQ are set to the inactive state and ERROR# is set active. This signal condition indicates to a 386SX that a 387SX compatible numerics coprocessor is connected. At least 8 clock periods must transpire after RESETIN goes inactive before the first CX-83S87 access. This pin is normally connected to the 386SX RESET input.

STEN (Status Enable)

The STEN input enables the functioning of the CX-83S87 when active. All outputs of the CX-83S87 are tri-stated when this signal is inactive. The other input signals are ignored while STEN is inactive. This signal can be used to assist in board level testing by isolating the CX-83S87 from the remainder of the circuit. This signal is normally connected to VCC through a resistor so that it can be pulled inactive during testing.

W/R# (Write or Read)

The W/R# input to the CX-83S87 indicates the direction of the current bus cycle. This signal is sampled simultaneously with ADS#, CMD0#, NPS1# and NPS2 on the rising edge of CPUCLK2. (See Table 5.1-1 for W/R#'s use in defining the current bus cycle). W/R# is normally connected to the 386SX W/R# output; setup and hold time are referenced to the rising edge of CPUCLK2.

NPS1# (MI/O#)	NPS2 (A31)	CMD0# (A2)	W/R#	Bus Cycle Type
1	-	-	-	Not a Coprocessor bus cycle
-	0	-	-	Not a Coprocessor bus cycle
0	1	0	0	Coproc Status or Mode Cntl Reg Read
0	1	0	1	Coproc Opcode Write
0	1	1	0	Coproc Data Write
0	1	1	1	Coproc Data Read

Table 5.1-1 PC Bus Cycle definition

5.2 Bus Operation

The CX-83S87 is compatible with the 386SX coprocessor protocol and executes reads and writes at zero wait-states. The coprocessor protocol uses the BUSY# and PEREQ signals to ensure that no coprocessor accesses are generated by the 386SX before the coprocessor is able to complete the transfer. This allows the CX-83S87 to issue READYO# immediately after receiving ADS#. Therefore, the processor bus is available for DMA cycles during the coprocessor instructions. However, the time required to check these signals contributes to the coprocessor overhead and slows down execution of the numeric instructions.

The following sections describe the bus activity that occurs for the various categories of numeric instructions. The CX-83S87 is fully synchronous to the 386SX bus operation and supports both pipelined and non-pipelined bus cycles. Examples of the coprocessor's I/O interface operation for the different categories of instructions are provided. The bus cycles run by the 386SX to fetch instructions or to transfer operands to or from memory are not shown. Also, the 386SX usually does not respond to the BUSY#, ERROR# and PEREQ changes as rapidly as indicated herein. The diagrams shown in this section merely reflect the sequence of events and the method of synchronization; they are not meant to imply actual execution times or bus cycle durations. STEN, NPS1#, and NPS2 are assumed active during this cycle and are not shown. The CMD0# and W/R# signals should be valid during the cycle also.

The following table categorizes the CX-83S87 instructions by their interface characteristics. The *# BUS CYCLES* column includes the instruction opcode transfer and its operand transfer(s) in the count of bus cycles. The *EARLY WRITE* column indicates that the 386SX may write the opcode even if BUSY# is active. The *USES BUSY#* and *USES PEREQ* columns indicate if the referenced signal is utilized during the instruction execution. A check mark (√) indicates that the signal is active during the instruction, while a dash (-) indicates that the signal is not active during the instruction. The *TYPE* column identifies the category to which the instruction belongs.

Instruction	Operand	# Bus Cycles	Early Write	Uses BUSY#	Uses PEREQ	Type
FCLEX		1	√	–	–	E
FFREE	ST(i)	1	–	√	–	A
F*funct*		1	–	√	–	A
FINIT		1	√	–	–	E
FLD	64-bit real	5	√	√	–	B
FLD	64-bit integer	5	√	√	–	B
FLD	32-bit integer	3	√	√	–	B
FLD	32-bit real	3	√	√	–	B
FLD	80-bit real	6	–	√	√	C
FLD	80-bit BCD	6	–	√	√	C
FLD	16-bit integer	2	–	√	√	C
FLD*const*		1	–	√	–	A
FLDCW	two Bytes	2	–	√	√	C
FLDENV	block pointer	8	–	√	√	C
F*math*	ST(i),ST	1	–	√	–	A
F*math*	ST,ST(i)	1	–	√	–	A
F*math*	64-bit real	5	√	√	–	B
F*math*	64-bit integer	5	√	√	–	B
F*math*	32-bit integer	3	√	√	–	B
F*math*	32-bit real	3	√	√	–	B
F*math*	16-bit integer	2	–	√	√	C
F*aps*	ST(i)	1	–	√	–	A
FRSTOR	block pointer	48	–	√	√	C
FSAVE	block pointer	48	–	√	√	C
FST	16-bit integer	2	–	√	√	C
FST	32-bit integer	3	–	√	√	C
FST	32-bit real	3	–	√	√	C
FST	64-bit real	5	–	√	√	C
FST	64-bit integer	5	–	√	√	C
FST	80-bit real	6	–	√	√	C
FST	80-bit BCD	6	–	√	√	C
FSTCW	two Bytes	2	–	–	–	D
FSTENV	block pointer	8	–	√	√	C
FSTSW	AX	2	–	–	–	D
FSTSW	two Bytes	2	–	–	–	D

F*math* represents the FADD, FCOM, FCOMP, FDIV, FDIVR, FMUL, FSUB, and FSUBR instructions.

FLD*const* represents the FLD1, FLDL2T, FLDL2E, FLDPI, FLDLG2, FLDLN2, and FLDZ instructions.

F*aps* represents the FLD ST(i), FXCH ST(i), FNOP, FCHS, FABS, FTST, FXAM, FUCOM, FUCOMP, FUCOMPP, and FCOMPP instructions.

F*funct* represents the F2XM1, FYL2X, FPTAN, FPATAN, FXTRACT, FPREM1, FDECSTP, FINCSTP, FPREM, FYL2XP1, FSQRT, FSINCOS, FRNDINT, FSCALE, FSIN, and FCOS instructions.

5.3 Category A Instructions

The category A instructions are the simplest type of CX-83S87 instruction. These instructions operate on internal registers and have no operand transfers between the 386SX and the CX-83S87. The 386SX coprocessor protocol tests BUSY# and waits for BUSY# to be inactive. The ERROR# signal is then examined. If ERROR# is asserted, the 386SX executes a trap to the coprocessor exception trap routine. If ERROR# is not asserted, the 386SX writes the opcode to the coprocessor and continues to the next instruction. The coprocessor asserts BUSY# during execution of the requested operation. ERROR# will be asserted if an unmasked exception condition occurs during the operation.

Figure 5.1 shows the bus signals during two successive category A instructions in the non-pipelined bus cycle operation. The first access occurs when the CX-83S87 is idle and starts immediately. The second access is delayed by the 386SX until BUSY# goes inactive.

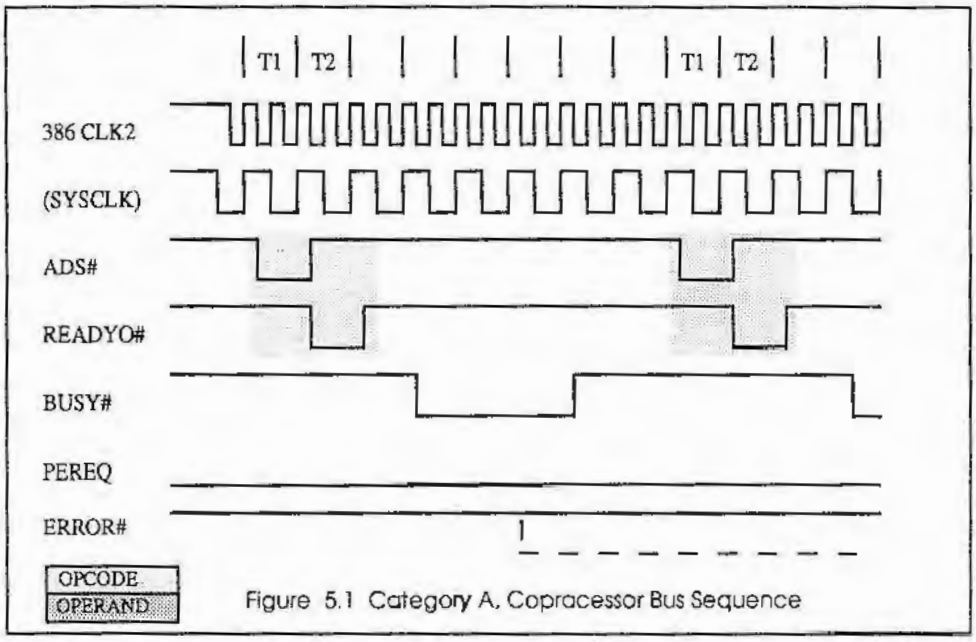


Figure 5.1 Category A, Coprocessor Bus Sequence

5.4 Category B Instructions

The category B instructions require the transfer of two or four 16-bit words from the 386SX to the CX-83S87. The execution of this instruction category is characterized by the 386SX writing the opcode whether or not BUSY# is active. The 386SX then waits for BUSY# to become inactive (if necessary) and examines the ERROR# signal. If ERROR# is asserted, the 386SX executes a trap to the coprocessor exception trap routine. If ERROR# is not asserted, the 386SX then transfers the operand (2 memory cycles for 32-bit operands, 4 memory cycles for 64-bit operands). Since synchronization is accomplished using BUSY# during the operand transfer phase, there is no activity on the PEREQ line. The 386SX then proceeds to the next instruction. The CX-83S87 asserts BUSY# after the operand is

accepted and removes BUSY# when the operation is complete. ERROR# is asserted if an unmasked exception condition occurs during the operation.

Figure 5.2 shows the bus activity for two successive category B instructions with a 32-bit operand in non-pipelined bus cycle operation. The initial state of the CX-83S87 is idle allowing the 386SX to immediately transfer both the instruction opcode and the operand. The second instruction is initiated while the CX-83S87 is busy. The opcode transfer is completed but the operand transfer is not initiated by the 386SX until BUSY# goes inactive.

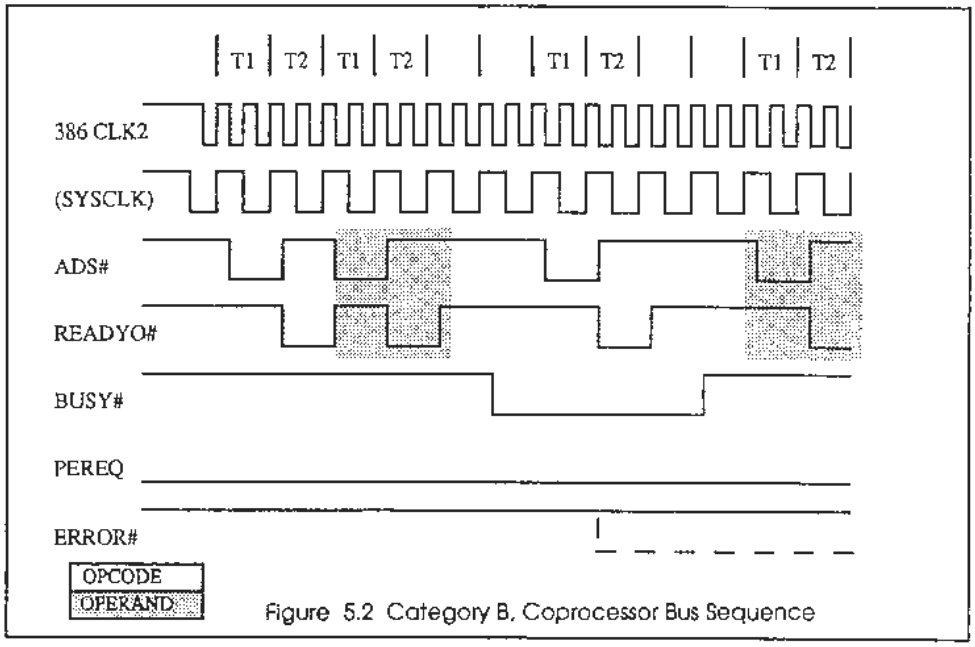


Figure 5.2 Category B, Coprocessor Bus Sequence

5.5 Category C Instructions

The category C instructions are characterized by the use of the PEREQ signal to synchronize the movement of the operand. All store operand instructions and the load operand with 16-bit and 80-bit data values comprise this category. The coprocessor bus activity is characterized by the 386SX waiting for BUSY# to be inactive before writing the opcode to the CX-83S87. The 386SX then waits for BUSY# and PEREQ to be active before transferring the operand.

The operand transfer can take 1, 2, 4 or 5 bus cycles as determined by its size. One bus cycle is required to transfer the 16-bit word integer and two bus cycles are required to transfer 32-bit short integer and single precision real formats. Four bus cycles are used to transfer the 64-bit long integer and 64-bit double precision real formats. Five bus cycles are used to transfer the 80-bit extended real and the 80-bit packed BCD formats.

BUSY# is active and PEREQ is inactive during the time that the CX-83S87 is generating the properly rounded value for the designated format. When the value is ready to be transferred, PEREQ is made active. In the coprocessor operation mode the 386SX will not try to read the value until PEREQ is active.

Figure 5.3 shows a non-pipelined bus cycle FSTP 32-bit integer with the CX-83S87 initially busy.

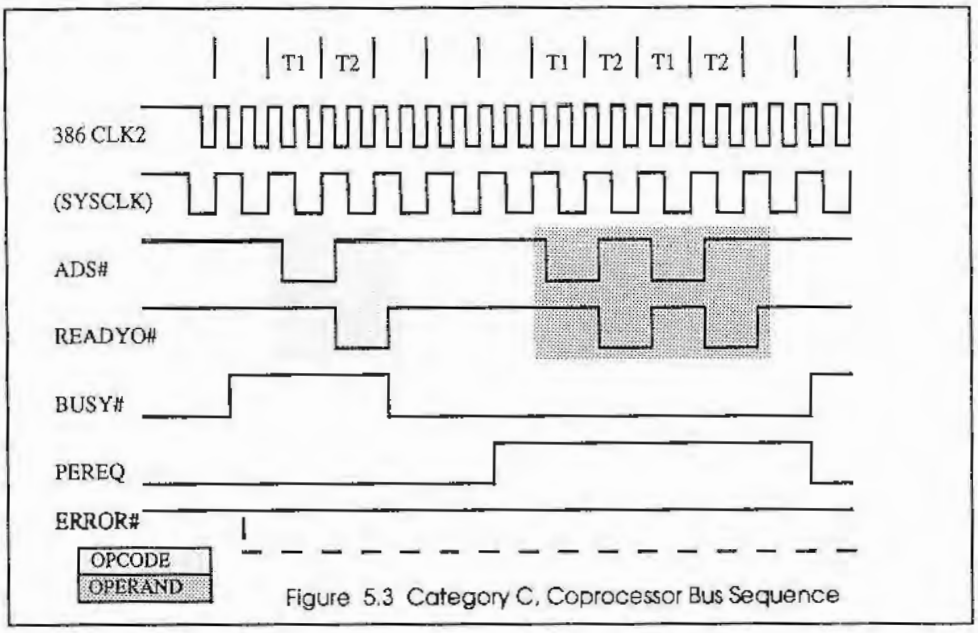


Figure 5.3 Category C, Coprocessor Bus Sequence

5.6 Category D Instructions

The category D instructions are those operations that wait for BUSY# to be inactive before transferring the instruction opcode and then transfer the operand (if there is one) immediately without using PEREQ. These instructions involve only the store of the status or control registers. The duration of the operation is so short that the BUSY# signal is not asserted. Figure 5.4 shows the coprocessor mode synchronization waiting for BUSY# to go inactive.

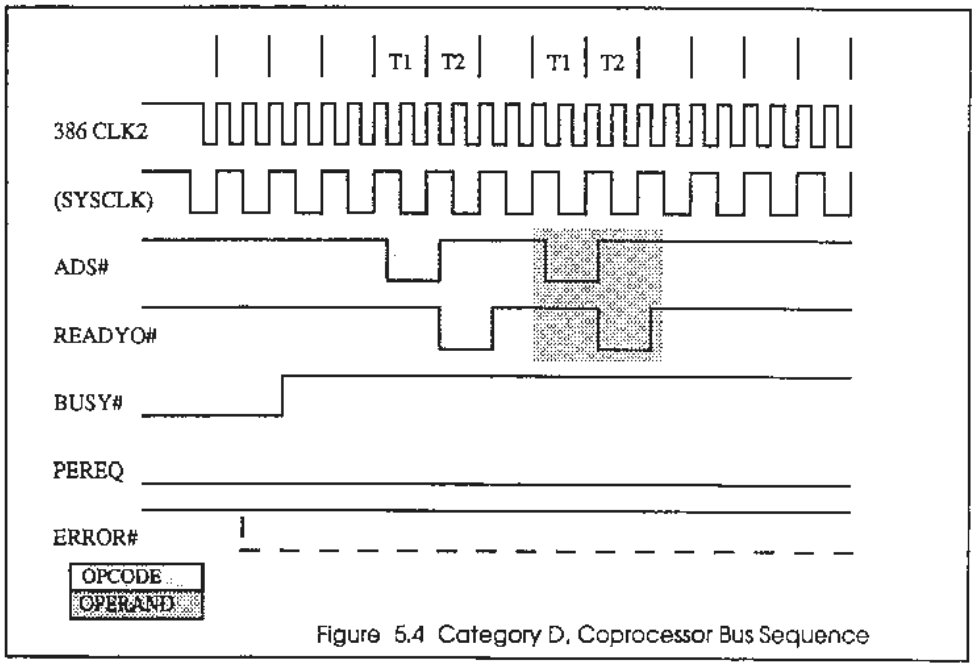


Figure 5.4 Category D, Coprocessor Bus Sequence

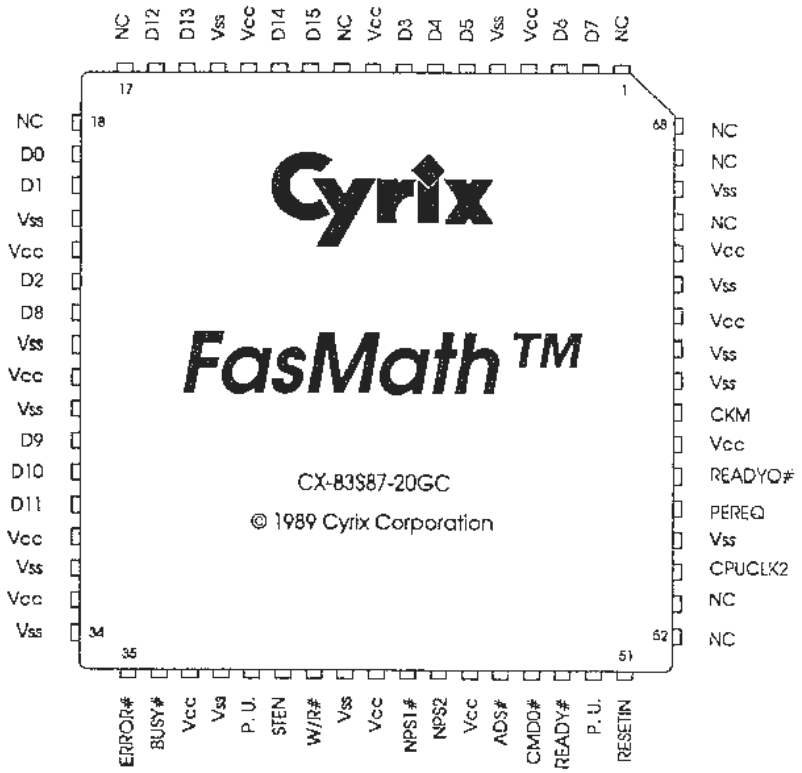
5.7 Category E Instructions

The category E instructions represent those instructions that are executed regardless of the condition of BUSY#. BUSY# will be asserted during the instruction if it is not already asserted. PEREQ is inactive during the execution of these instructions. These instructions operate on the status and control portion of the CX-83S87. The entire instruction consists of writing the operation code to the coprocessor. There are no operands to transfer and no synchronization to be performed.

6. Mechanical Specifications

The CX-83S87 is packaged in a 68 pin j-lead cerquad package. The following diagram details the pinout of the CX-83S87.

Top Side View



The following chart provides a cross reference of signal name to pin number

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
NC	1	NC	18	ERROR#	35	NC	52
D7	2	D0	19	BUSY#	36	NC	53
D6	3	D1	20	Vcc	37	CPUCLK2	54
Vcc	4	Vss	21	Vss	38	Vss	55
Vss	5	Vcc	22	P.U.	39	PEREQ	56
D5	6	D2	23	STEN	40	READYO#	57
D4	7	D8	24	W/R#	41	Vcc	58
D3	8	Vss	25	Vss	42	CKM	59
Vcc	9	Vcc	26	Vcc	43	Vss	60
NC	10	Vss	27	NPS1#	44	Vss	61
D15	11	D9	28	NPS2	45	Vcc	62
D14	12	D10	29	Vcc	46	Vss	63
Vcc	13	D11	30	ADS#	47	Vcc	64
Vss	14	Vcc	31	CMD0#	48	NC	65
D13	15	Vss	32	READY#	49	Vss	66
D12	16	Vcc	33	P.U.	50	NC	67
NC	17	Vss	34	RESETIN	51	NC	68

Note: Pins 39 and 50 shown as P.U. ("Pull Up") **must be tied to Vcc** for the device to operate properly.

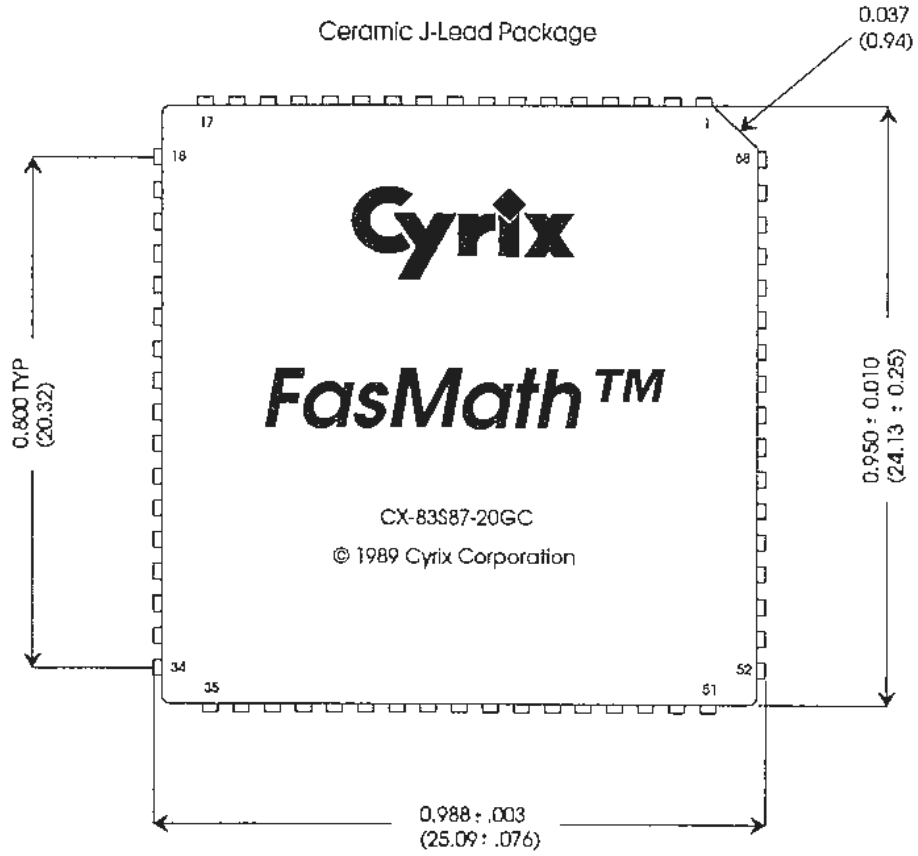
The Cyrix CX-83S87 is available in a ceramic j-lead cerquad package. This package is identical in size,shape, and pin-out to the Intel 387SX 68-pin PLCC package. The package is described in the following section.

6.1 Mechanical Package Specifications:

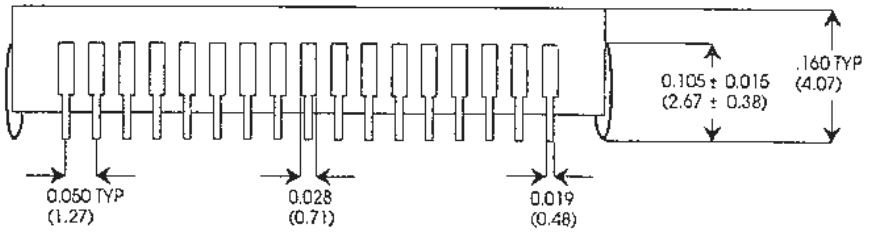
The dimensions for both the cerquad package and the PLCC package are detailed on the following pages.

Top Side View

Ceramic J-Lead Package



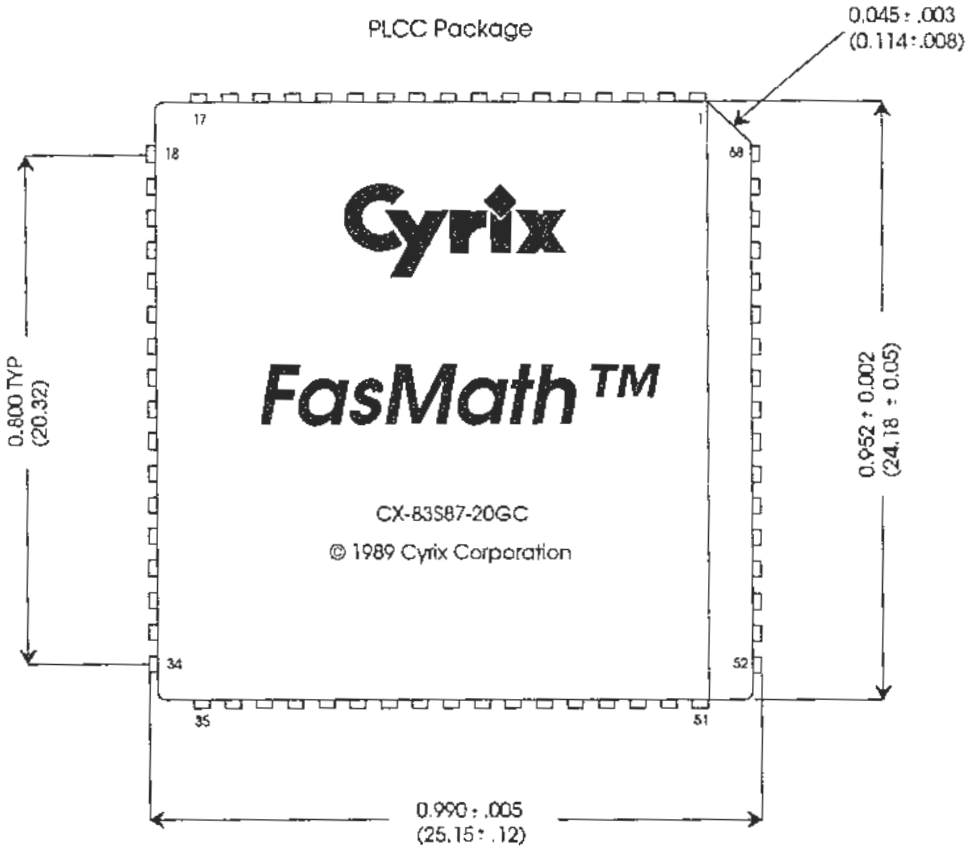
Ceramic J-Lead Package



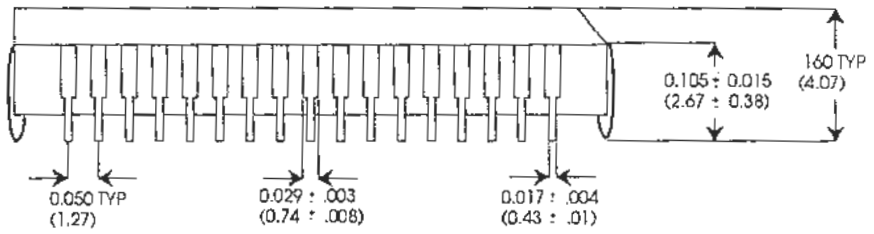
Note: 1. All dimensions in inches (millimeters).
2. Solder dip finish.

Top Side View

PLCC Package



PLCC Package



Note: 1. All dimensions in inches (millimeters).
2. Solder plate finish.
3. Seating plane and Tweezing coplanarity are within .004 inches

7. Electrical Specifications

7.1 Absolute Maximum Ratings

The following table lists obsolute maximum ratings for the CX-83S87 device. Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and do not imply that operation under any conditions other than those listed under "Recommended Operating Conditions" is possible. Exposure to conditions beyond the "Absolute Maximum Ratings" (1) will reduce device reliability and (2) result in premature failure *even when there is no immediately apparent sign of failure*. Prolonged exposure to conditions at or near "Absolute Maximum Ratings" may also result in reduced useful life and reliability.

Parameter	Min.	Max.	Units	Notes
Case Temperature	-0°	+100°	°C	Power Applied
Storage Temperature	-65°	+150°	°C	No Bias
Supply Voltage, VCC	-0.5	+6.0	Volts	With respect to VSS
Voltage On Any Pin	-0.5	VCC+0.5	Volts	With Respect to VSS
Power Dissipation		1.6	Watts	
Input Clamp Current, IIK		10	mA	VI<VSS or VI>VCC
Output Clamp Current, IOK		25	mA	VO<VSS or VO>VCC

7.2 Recommended Operating Conditions

The following table presents the recommended operating conditions for the device:

Parameter		Min.	Max.	Units	Notes
TC	Case Temperature	-0°	+85°	°C	Power Applied
VCC	Supply Voltage	+4.5	+5.5	Volts	With respect to VSS
VIH	High Level Input	2.0	VCC	Volts	
VIL	Low Level Input	0.0	0.8	Volts	
IOH	Output Current(High)		-1.0	mA	VOH=VOH(min)
IOL	Output Current (Low)		+4.0	mA	VOL=VOL(max)
IIK	Input Clamp Current		±10	mA	VI<VSS or VI>VCC
IOK	Output Clamp Current		±25	mA	VO<VSS or VO>VCC

7.3 DC Electrical Characteristics

Parameter	Min.	Max.	Units	Notes
V_{CL} Clock Input Low	0	0.8	Volts	With respect to V_{SS}
V_{CH} Clock Input High	3.7	Vcc	Volts	
V_{OL} Output Low Voltage		+0.45	Volts	I_{OL}=4.0 mA
V_{OH} Output High Voltage	2.4		Volts	I_{OH}=1.0 mA
I_{CC_I} Supply Current - Idle		9	mA	CLK2=20 Mhz (Typ=6)
I_{CC_R} Supply Current - Running		300	mA	CLK2=20 Mhz (Typ=150)
I_{LI} Input Leakage		±15	µA	0<VIN<VCC
I_{LO} I/O Leakage		±15	µA	0.45<VO<VCC
C_{IN} Input Capacitance		10	pf	fc=1 Mhz
C_O I/O Capacitance		12	pf	fc=1 Mhz
C_{CLK} Clock Capacitance		20	pf	fc=1 Mhz

7.4 CX-83S87 Switching Characteristics

The following table summarizes the timing requirements of the CX-83S87-20 and -16 devices.

CX-83S87 Speed			20 Mhz		16 Mhz			
Pin	Sym	Parameter	Min (nsec)	Max (nsec)	Min (nsec)	Max (nsec)	Fig.	Notes
CLK2	T1	Period	25	500	31.25	500	7.1	2.0V
CLK2	T2a	High Time	8		9			2.0V
CLK2	T2b	High Time	5		5			3.8V
CLK2	T3a	Low Time	8		9			2.0V
CLK2	T3b	Low Time	6		7			0.8V
CLK2	T4	Fall Time		8		8		3.7V To 0.8V
CLK2	T5	Rise Time		8		8		0.8V To 3.7V
READYO#	T7	Out Delay	3	23	4	27	7.2	CL=75 pf
PEREQ	T7	Out Delay	5	28	5	30		CL=75 pf
BUSY#	T7	Out Delay	5	28	5	30		CL=75 pf
ERROR#	T7	Out Delay	5	28	5	30		CL=75 pf
D15-D0	T8	Out Delay	1	32	1	37	7.3	CL=120 pf
D15-D0	T10	Setup Time	10		10			
D15-D0	T11	Hold Time	11		11			
D15-D0	T12	Float Time	6	27	6	33		CL=120 pf
PEREQ	T13	Float Time	1	40	1	45		CL=75 pf
BUSY#	T13	Float Time	1	40	1	45		CL=75 pf
ERROR#	T13	Float Time	1	40	1	45		CL=75 pf
READYO#	T13	Float Time	1	40	1	45		CL=75 pf
ADS#	T14	Setup Time	15		20		7.3	
ADS#	T15	Hold Time	5		5			
READY#	T16	Setup Time	10		10		7.3	
READY#	T17	Hold Time	4		4			
CMD0#, NPS1#, NPS2, W/R#	T16	Setup Time	15		20		7.3	
CMD0#, NPS1#, NPS2, W/R#	T17	Hold Time	0		0			
STEN	T16	Setup Time	15		20		7.3	
STEN	T17	Hold Time	2		2			
RESETIN	T18	Setup Time	12		13		7.4	
RESETIN	T19	Hold Time	4		4			

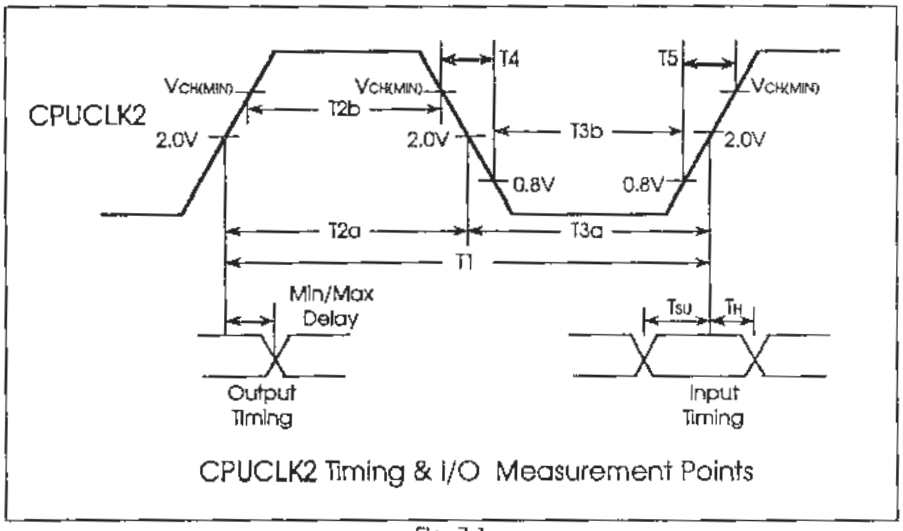

CPUCLK2
VCH(MIN)
VCH(MIN)
VCH(MIN)
T4
T5
T2b
T3b
2.0V
2.0V
2.0V
0.8V
0.8V
T2a
T3a
T1
Min/Max
Delay
TSU
TH
Output
Timing
Input
Timing
CPUCLK2 Timing & I/O Measurement Points

Fig. 7.1

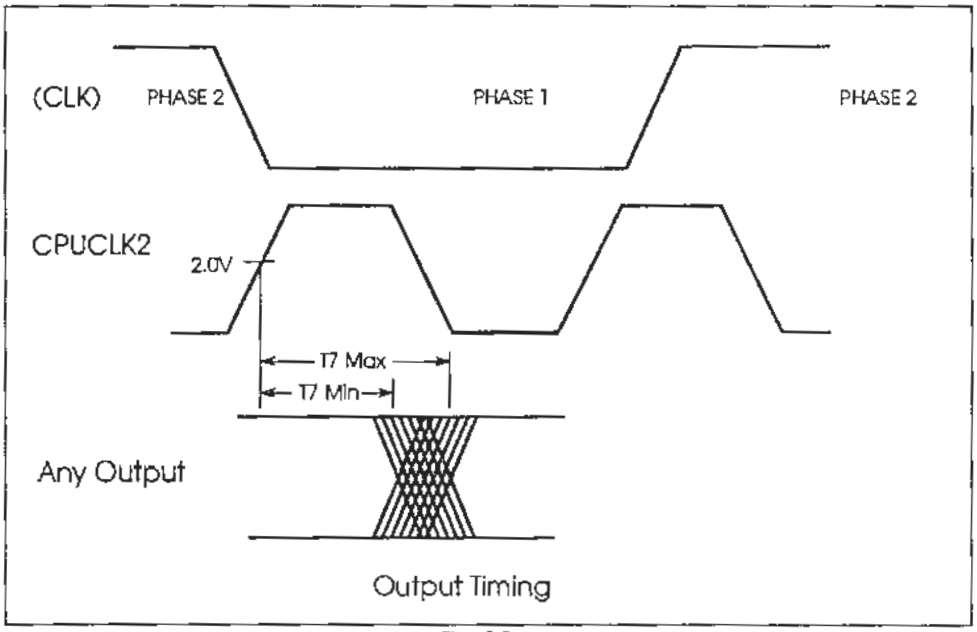

(CLK)
PHASE 2
PHASE 1
PHASE 2
CPUCLK2
2.0V
T7 Max
T7 Min
Any Output
Output Timing

Fig. 7.2

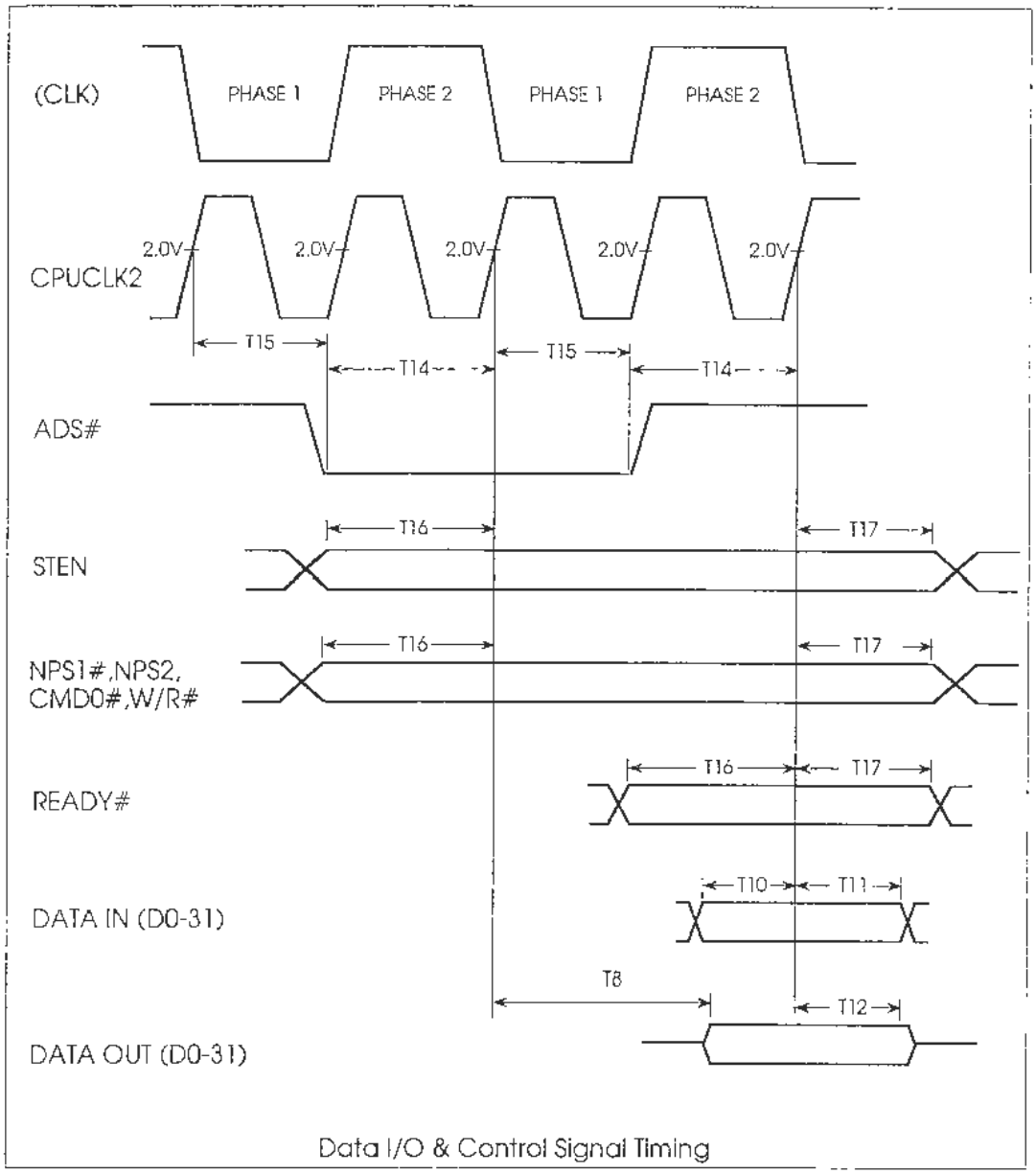

(CLK)
PHASE 1
PHASE 2
PHASE 1
PHASE 2
CPUCLK2
2.0V
2.0V
2.0V
2.0V
2.0V
T15
T14
T15
T14
ADS#
T16
T17
STEN
T16
T17
NPS1#,NPS2,
CMD0#,W/R#
T16
T17
READY#
T10
T11
DATA IN (D0-31)
T8
T12
DATA OUT (D0-31)
Data I/O & Control Signal Timing

Fig. 7.3

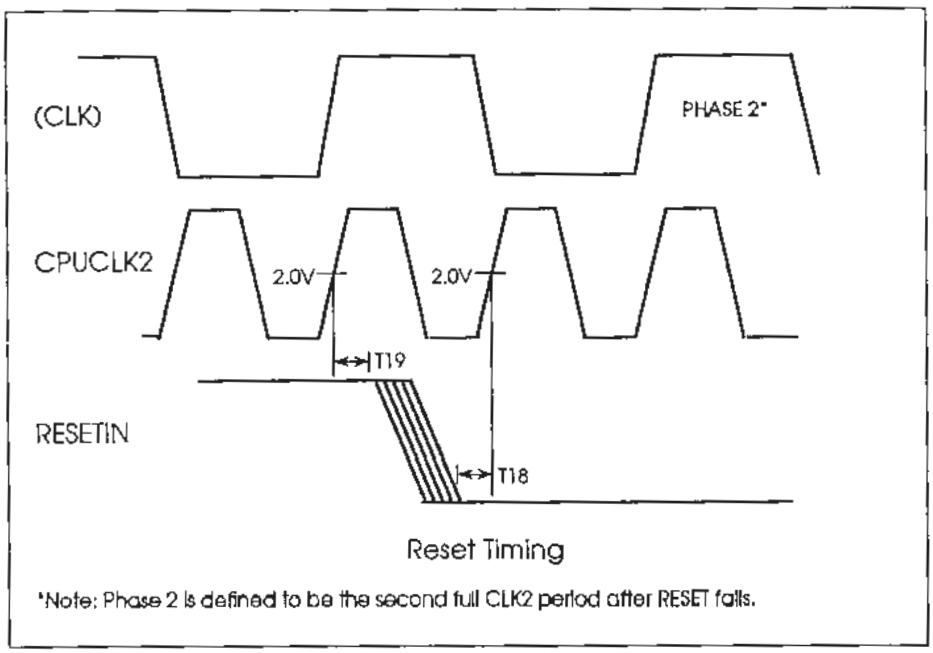
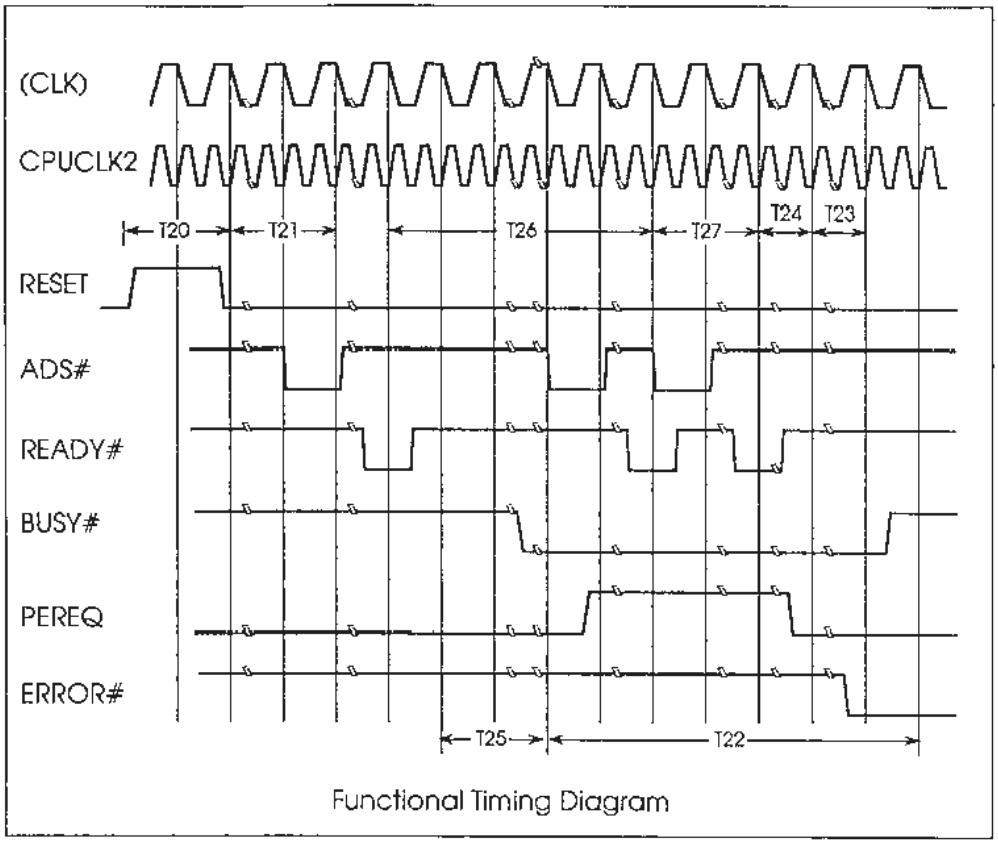


Fig. 7.4

7.5 Interface Timing Parameters

The following table sets forth the clock cycle timing requirements for major CX-83S87 interface functions. **Times are specified in CPUCLK2** cycle counts. Please refer to figure 7.5 for timing reference information.

Pin	Symbol	Parameter	Min	Max	Notes
RESETIN	T20	Time Active	20		
RESETIN	T21	Time Inactive	8		Before 1st Opcode Write
BUSY#	T22	Time Active	6		
BUSY#	T23	Delay Inactive	6		From ERROR# Inactive
ERROR#	T24	Delay Active	6		From PEREQ Inactive
BUSY#	T25	Delay Active		4	From READY# Active
READY#	T26	Delay	0		Opcode Write to Next Cycle
READY#	T27	Delay	0		Operand cycle to next operand cycle



Functional Timing Diagram

Fig. 7.5

Appendices

FasMath™ Glossary:

BASE: (1) a term used to describe the fundamental number for a logarithmic or exponential function. In both cases, it is a number that is raised to a power. The two equations ($y = \log_B$ of x) and ($B^y = x$) are the same. Base is sometimes called a radix. (2) a number system pertaining to the base. Example: Base 2 is the binary number system; base 10 is the decimal number system; base 16 is the hexadecimal number system. In each case a succeeding digit is an increase by a factor of the base.

BIAS: A constant.

BIASED EXPONENT: An exponent which has been divided by a bias so as to have a valid positive and negative range. For example, the double precision real format has a bias of 1023 ($(2^{11}/2)-1$) when an exponent is non-zero. Given an 11-bit biased exponent of 11111111100, which is 2044, the true exponent is 2044 - 1023 which equals +1021.

BINARY CODED DECIMAL (BCD): A decimal number representation which occupies a nibble (4-bits). A nibble is most often used to represent a hexadecimal digit; however when representing a BCD value the hexadecimal values A through F (1010 - 1111 binary) are not used. The FasMath™ processor supports the use of the BCD format by storing 18 decimal digits and the sign (+/-) into 10 bytes.

BINARY FLOATING-POINT NUMBER: A binary string made up of the following three components: a sign, a signed exponent, and a significand. Its numerical value, if any, is the signed product of its significand and two raised to the power of its exponent.

BINARY POINT: A point or dot expressing the denominator of 2^{-n}. Functionally the same as a decimal point.

C0-C3 Four bits of the FasMath™ processor status register known as the "condition code". These bits are set to certain values by the compare, test, examine, and remainder functions of the FasMath™ processor. Additionally, the C1 bit defines the direction of round-up if precision exception is set.

CONDITION CODE: Four bits of the FasMath™ processor status register which indicate the results of the compare, test, examine, and remainder functions of the FasMath™ processor as well a the direction of round-up if precision exception is set.

CONTROL MODE REGISTER: A 16-bit FasMath™ processor register which is programmed to control the modes of computation as well as which exception can cause an interrupt. After a hardware reset invalid exception is masked and all other exceptions are unmasked. Precision control is set to extended and rounding control is set to round nearest. The control mode register contains the value $037E after reset.

DENORMAL: A nonzero floating-point number whose biased exponent has the reserved value of zero as well as having o significand with a value of zero left of the binary point. On the FasMath™ processor a denormal is a number represented by:

$$0.\text{binary\_fraction} * 2^{-126} \qquad \text{Where binary\_fraction} > 0.$$

DOUBLE EXTENDED FORMAT: An 80-bit format consisting of a sign bit, a 15-bit biased exponent, and a 64-bit significand with an explicit integer bit and 63 fractional-part bits. Also called the extended double precision format.

DOUBLE PRECISION FORMAT: A 64-bit format consisting of a sign bit, an 11-bit biased exponent, and a 52-bit significand with an implicit leading integer bit.

ENVIRONMENT: Current state of the FasMath™ processor.

EXCEPTION: Any of the six conditions (invalid operand, denormal, numeric overflow, numeric underflow, zero-divide, and precision) detected by the FasMath™ processor which causes an interrupt or the status register to be changed.

EXCEPTION POINTERS: One or two pointers maintained by the 80386 to help exception handlers identify the cause of a numerics processor exception. One pointer always points to the most recently executed numerics processor instruction. The second pointer points to the memory operand of the instruction, only if the instruction had a memory operand. The exception pointers can be accessed with either the FSTENV or FSAVE instructions.

EXPONENT: (1) any number which denotes the power to which another number is raised. (2) the field of an IEEE-754-1985 floating-point number that indicates the magnitude of the number minus the bias for the particular precision format.

EXTENDED DOUBLE PRECISION FORMAT: An 80-bit format consisting of a sign bit, a 15-bit biased exponent, and a 64-bit significand with an explicit integer bit and 63 fractional-part bits. Also called double extended format.

FLOATING-POINT: A number that is expressed as a signed significand times a base raised to the power of a signed exponent. In the case of the FasMath™ processor, the base is 2.

FRACTION: The field of the significand that lies to the right of its implied binary point.

GRADUAL UNDERFLOW: A method of minimizing the loss of accuracy in a result when an underflow error condition occurs.

IMPLICIT INTEGER BIT: an implyed "1" (single bit) left of the significand's binary point in the single and double precision real formats that is not explicitly represented as part of the IEEE-754-1985 data format. That is the IEEE formats assume a given significand lies to the right of the binary point with a implied single bit of 1 to the left, for all numbers except denormals. Denormals have an implied single bit of 0.

INDEFINITE: A result from a function that cannot be precisely represented when the inputs are such that no other reassonable answer is possible. IEEE-754-1985 defines a quiet NaN for each floating-point format to represent the indefinite value. For binary integer formats the most negative number is considered the indefinite value. For the BCD format, the sign byte and the two uppermost digit bytes contains all 1's.

INEXACT: A result which could not be precisely represented and sets precision exception in the FasMath™ processor's status register.

INFINITY: A number without bound. The IEEE-754-1985 specification considers infinity as another number, subject to special rules of arithmetic. All three floating-point formats (single, double and extended double precision) provide representations of $+\infty$ and $-\infty$.

INTEGER: A whole number (positive, negative, or zero) that has no fractional part. It is quite common for integers to be represented in a floating-point format; this is what the FasMath™ processor does whenever an integer is pushed onto the FasMath™ processor stack.

INTEGER BIT: The part of the significand that lies to the left of the binary point. The integer bit is always one, except for denormals. In the extended double precision format the integer bit is explicit. In the single and double precision formats the integer bit is implicit and is not actually stored in memory.

INVALID OPERATION: Any exception which is not covered by other exceptions reported in the status register.

LONG INTEGER: A 64-bit two's complement number.

LONG REAL: A 64-bit format consisting of a sign bit, an 11-bit biased exponent, and a 52-bit significand with an implicit leading integer bit. More commonly referred to as double precision format.

MANTISSA: The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right. The mantissa is also called the significand.

MASKED: A term used to describe a blocking of exceptions. The FasMath™ processor exception bits (P, U, O, Z, D, I), found in the status register, are masked when a corresponding bit in the control mode register is set to one. When an exception bit is masked, the FasMath™ processor will not generate an interrupt on an exception but will provide its own exception recovery.

MODE: Two fields in the status register, "rounding control" and "precision control" which control the execution of arithmetic operations. Programs can set, sense, save, and restore these fields.

NaN: An abbreviation for "Not a Number"; it is a floating-point entity that does not represent any numeric or infinite value. There are two types of NaNs: (1) Signaling NaNs (SNaN) signal the invalid operation exception whenever they appear as operands. (2) Quiet NaNs (QNaN) propagate through almost every arithmetic operation without signaling an exception.

NORMAL: The representation of a floating-point number in which the significand has an integer bit one (either explicit or implicit).

NORMALIZE: A number which has been converted from a denormal representation to a normal representation.

NUMERIC OVERFLOW: An exception generated when an answer is finite but is too large to be represented in the destination format.

NUMERIC UNDERFLOW: An exception generated when an answer is too small to be represented in the destination format. The IEEE-754-1985 standard specifies that an attempt should be made to represent the number as a denormal.

PACKED DECIMAL: A BCD number. On the FasMath™ processor it is a 10-byte quantity, with 18 binary coded decimal digits and one byte for the sign.

POP: To remove the last item placed on the stack.

PRECISION: The number of bits needed to represent a significand of a floating-point number.

PRECISION CONTROL: Two bits in the FasMath™ processor's control mode register which allows all arithmetic operations to be performed with less precision. Since there is no performance advantage when using this option, its only use is for strict compatibility with the IEEE-754-1985 standard and with older main frame computer systems.

PRECISION EXCEPTION: An exception condition which occurs when the results of a calculation is not exact. The IEEE-754-1985 specification refers to precision exception as an "inexact" result.

PSEUDOZERO: A set of special values in the extended double precision format which consists of numbers with a zero significand and an exponent that is neither all zeros nor all ones.

QUIET NaN (QNaN): A NaN which has a one in the most significant bit of the fractional part of the significand. QNaNs can undergo certain operations without causing an exception.

REAL: Any rational or irrational value (negative, positive, or zero) that can be represented as a quantity or a number; this does not include imaginary numbers.

SHORT INTEGER: A 32-bit two's complement number.

SHORT REAL: A 32-bit format consisting of a sign bit, an 8-bit biased exponent, and a 23-bit significand with a leading implicit integer bit. More commonly referred to as single precision format.

SIGNALING NaN (SNaN): A NaN which has the most significant bit of the fractional part of the significand equal to zero. A SNaN will cause an invalid-operation exception whenever it enters into a calculation or comparison.

SIGNIFICAND: The component of a binary floating-point number that consists of an explicit or implicit leading bit equal to 1 that is left of its implied binary point and has a fraction field to the right. The significand is also called the mantissa.

SINGLE PERCISION FORMAT: A 32-bit format consisting of a sign bit, an 8-bit biased exponent, and a 23-bit significand with a leading implicit integer bit.

SSS: The three-bit field of the status register which points to one of the eight FasMath™ processor stack registers; this register is called the current top of stack.

STACK FAULT: An exception which results from either the source register being empty or the destination register being full.

STATUS REGISTER: A 16-bit register which indicates the current status of the FasMath™ processor. The status register contains condition codes, the stack pointer, busy, and exception flags. The exception flags can be cleared through software. After a hardware reset, the invalid exception is set and the status register will contain a value of $8081.

TAG REGISTER: A 16-bit register that describes each of the eight FasMath™ processor's stack registers. After a hardware reset all stack locations are tagged empty; thus, this register contains all ones ($FFFF).

TEMPORARY REAL: An 80-bit format consisting of a sign bit, a 15-bit biased exponent, and a 64-bit significand with an explicit integer bit and 63 fractional-part bits. More commonly referred to as extended double precision format.

TOS: The stack register which is pointed to by the SSS bits in the status register.

TRANSCENDENTAL: Functions for which polynomial formulas are used to approximate. The FasMath™ processor supports transcendental operations such as trigonometric, exponential, and logarithmic functions.

TWO'S COMPLEMENT: The IEEE-754-1985 defined method for representing integers. The upper most bit determines the sign: 1 for a negative value, 0 for a positive value. For example, the 8-bit number 11111000 is -8, obtained by subtracting 256 (2^8) from 248 ($2^7+2^6+2^5+2^4+2^3$).

UNBIASED EXPONENT: The actual exponent value that tells where to place the binary point of the significand of a floating-point number. For example, if a single precision format biased exponent is 142 (single precision has a Bias of 127) the unbiased exponent is +15. Thus, the real number being represented is the significand with the binary point shifted 15 bits to the right.

UNMASKED: a term that applies to each of the six FasMath™ processor exceptions: I,D,Z,O,U,P. An exception is unmasked if a corresponding bit in the FasMath™ processor control mode register is set to zero. If an exception is unmasked, the FasMath™ processor will generate an interrupt when the exception condition occurs. The user can provide an interrupt routine to customize exception recovery.

WORD INTEGER: A 16-bit two's complement integer format supported by both the 80386 and the FasMath™ processor.

Opcode Map:

The opcode map tables below are included to help in debugging. However, a little explanation on how to use and read these tables is in order.

The tables below are arranged in a spread sheet like manner with the upper byte of the numerics processor's 16-bit opcode identifying a particular table. The low byte's high nibble of the numerics processor's opcode identifies a particular row in the spread sheet like table and the low byte's low nibble identifies the column in the table. A picture of this is shown below.

Given a 16-bit opcode:

8-bits Upper byte	4-bits Low byte high nibble	4-bits Low byte low nibble
Opcode Map Table	Row	Column

D8

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FADD ST(0) + 32-bit real		FMUL ST(0) * 32-bit real	
1, 5, or 9	FCOM ST(0) =? 32-bit real		FCOMP ST(0) =? 32-bit real	
2, 6, or A	FSUB ST(0) - 32-bit real		FSUBR 32-bit real - ST(0)	
3, 7, or B	FDIV ST(0) / 32-bit real		FDIVR 32-bit real / ST(0)	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	FADD ST(0) = ST(0) + ST(i)								FMUL ST(0) = ST(0) * ST(i)							
D	FCOM ST(0) =? ST(i)								FCOMP ST(0) =? ST(i)							
E	FSUB ST(0) = ST(0) - ST(i)								FSUBR ST(0) = ST(i) - ST(0)							
F	FDIV ST(0) = ST(0) / ST(i)								FDIVR ST(0) = ST(i) / ST(0)							
i =	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

Legend

U = Undefined Opcode

Example:

FSQRT = Opcode D9FA

D9

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FLD TOS <- 32-bit real		Undefined	
1, 5, or 9	FST ST(0) -> 32-bit real		FSTP ST(0) -> 32-bit real	
2, 6, or A	FLDENV		FLDCW	
3, 7, or B	FSTENV		FSTCW	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	FLD TOS <- ST(i)								FXCH ST(i) <--> ST(0)							
D	FNOP	Undefined							Undefined							
E	FCHS	FABS	U	U	FTST	FXAM	U	U	FLD1	FLDL2T	FLDL2E	FLDPI	FLDLG2	FLDLN2	FLDZ	U
F	F2XM1	FYL2X	FPTAN	FPATAN	FXTRACT	FPREM1	FDECSTP	FINCSTP	FPREM	FYL2XP1	FSQRT	FSINCOS	FRNDINT	FSCALE	FSIN	FCOS
i =	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

DA

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FIADD ST(0) + 32-bit int		FIMUL ST(0) * 32-bit int	
1, 5, or 9	FICOM ST(0) =? 32-bit Int		FICOMP ST(0) =? 32-bit int	
2, 6, or A	FISUB ST(0) - 32-bit Int		FISUBR 32-bit int - ST(0)	
3, 7, or B	FIDIV ST(0) / 32-bit Int		FIDIVR 32-bit Int / ST(0)	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	Undefined								Undefined							
D	Undefined								Undefined							
E	Undefined								U	FUCOMPP	Undefined					
F	Undefined								Undefined							

DB

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FILD TOS <- 32-bit int		Undefined	
1, 5, or 9	FIST ST(0) -> 32-bit int		FISTP ST(0) -> 32-bit int	
2, 6, or A	Undefined		FLD TOS <- 10 byte real	
3, 7, or B	Undefined		FSTP ST(0) -> 10 byte real	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C									Undefined							
D									Undefined							
E	FENI*	FDISI*	FCLEX	FINIT	FSETPM*	U			Undefined							
F									Undefined							

* The CX-83587 treats these instructions as FNOP's.

DC

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FADD ST(0) + 64-bit real		FMUL ST(0) * 64-bit real	
1, 5, or 9	FCOM ST(0) =? 64-bit real		FCOMP ST(0) =? 64-bit real	
2, 6, or A	FSUB ST(0) - 64-bit real		FSUBR 64-bit real - ST(0)	
3, 7, or B	FDIV ST(0) / 64-bit real		FDIVR 64-bit real / ST(0)	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	FADD ST(0) = ST(0) + ST(i)								FMUL ST(0) = ST(0) * ST(i)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D	Undefined								Undefined							
E	FSUBR ST(i) = ST(0) - ST(i)								FSUB ST(i) = ST(i) - ST(0)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
F	FDIVR ST(i) = ST(0) / ST(i)								FDIV ST(i) = ST(i) / ST(0)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

DD

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FLD TOS <- 64-bit real		Undefined	
1, 5, or 9	FST ST(0) -> 64-bit real		FSTP ST(0) -> 64-bit real	
2, 6, or A	FRSTOR		Undefined	
3, 7, or B	FSAVE		FSTSW	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	FFREE ST(i)								Undefined							
i=	0	1	2	3	4	5	6	7								
D	FST ST(i)								FSTP ST(i)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
E	FUCOM ST(i)								FUCOMP ST(i)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
F	Undefined								Undefined							

DE

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FIADD ST(0) + 16-bit Int		FIMUL ST(0) * 16-bit Int	
1, 5, or 9	FICOM ST(0) =? 16-bit Int		FICOMP ST(0) =? 16-bit Int	
2, 6, or A	FISUB ST(0) - 16-bit Int		FISUBR 16-bit Int - ST(0)	
3, 7, or B	FIDIV ST(0) / 16-bit Int		FIDIVR 16-bit Int / ST(0)	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	FADDP ST(i) = ST(0) + ST(i)								FMULP ST(i) = ST(i) * ST(0)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
D	Undefined								U	FCOMPP	Undefined					
E	FSUBRP ST(i) = ST(0) - ST(i)								FSUBP ST(i) = ST(i) - ST(0)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
F	FDIVRP ST(i) = ST(0) / ST(i)								FDIVP ST(i) = ST(i) / ST(0)							
i=	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

DF

Low nibble

High nibble	0...	7	8...	F
0, 4, or 8	FILD TOS <- 16-bit Int		Undefined	
1, 5, or 9	FIST ST(0) -> 16-bit Int		FISTP ST(0) -> 16-bit Int	
2, 6, or A	FBLD TOS <- 10 byte BCD		FILD TOS <- 64-bit Int	
3, 7, or B	FBSTP ST(0) -> 10 byte BCD		FISTP ST(0) -> 64-bit Int	

Low nibble

High nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	Undefined								Undefined							
D	Undefined								Undefined							
E	FSTSWAX	Undefined							Undefined							
F	Undefined								Undefined							

Cyrix™

Advancing the Standards

P.O. Box 850118, Richardson, TX 75085-0118, (214) 234-8387

Printed in the USA

$9.95

Order Number: L2005-002

Ultra Low Power Operation
of the
Cyrix 83S87 & 82S87

83S87:

Ultra low power (typically 150 µW with CMOS input levels) can be achieved with the 83S87 by stopping the CLK2 input to the device.

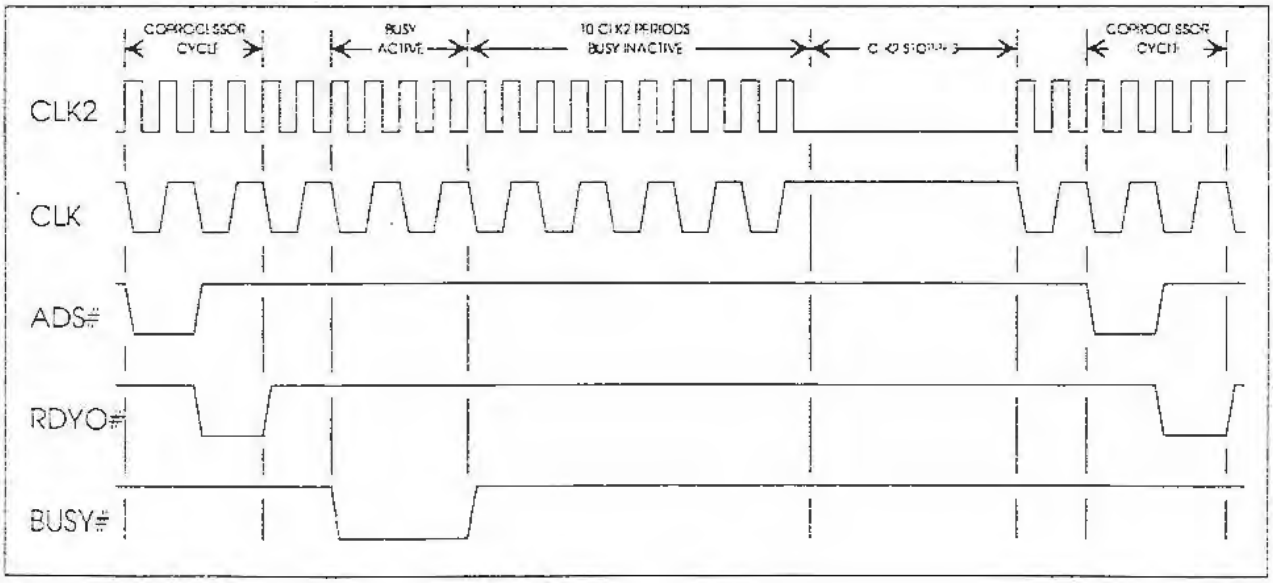
The Cyrix Recommended Stop-the-Clock Procedure:
Cyrix recommends that both the CPU and Coprocessor use the same CLK2 input signal. By doing so, the system designer is insured that the CPU and Coprocessor remain synchronized when starting and stopping the CLK2 signal.

Before stopping CLK2, two conditions must be met: (1) the current 386SX bus cycle must not be in the process of accessing the 83S87 and (2) BUSY# must be inactive for at least 10 CLK2 periods prior to stopping CLK2. Failure to stop CLK2 while the 83S87 is idle, can result in a system READY# hang after CLK2 is restarted. The FWAIT instruction forces instruction execution to wait until BUSY# is inactive for 10 CLK2 periods. This instruction should be executed in the interrupt routine used to stop the CLK2 signal.

When CLK2 is restarted, the 386SX and the 83S87 must restart in the same CLK phase as they were stopped. Failure to restart CLK2 properly can result in a system READY# hang.

The following figure illustrates the 83S87 restrictions for stopping the CLK2 input:

NOTE: The CLK2 input may be stopped in any phase. The figure below represents only one example of this.

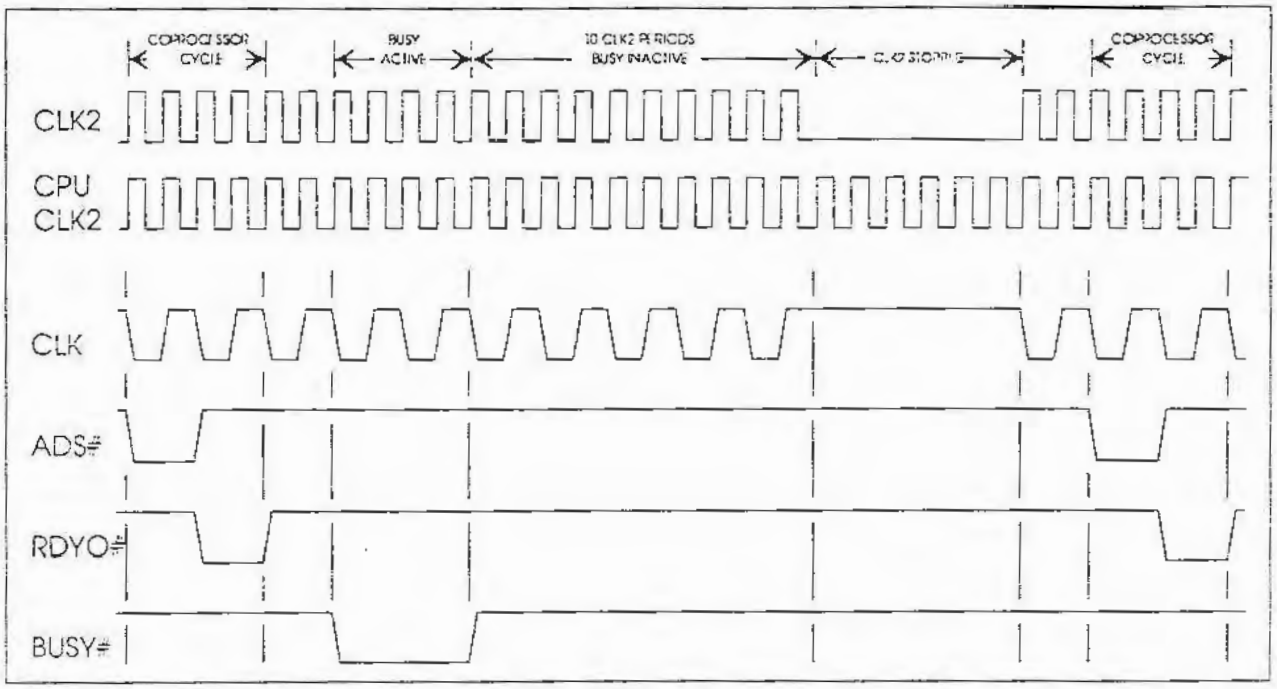


Alternate Stop-the-Clock Procedure:
Cyrix does not recommend that the Coprocessor CLK2 input be stopped independently of the CPU CLK2 input. However, if a designer wishes to pursue this alternative the steps below must be followed at a minimum.

1. Before stopping CLK2, two conditions must be met: (1) the current 386SX bus cycle must not be in the process of accessing the 83S87 and (2) BUSY# must be inactive for at least 10 CLK2 periods prior to stopping CLK2. Failure to stop CLK2 while the 83S87 is idle, can result in a system READY# hang after CLK2 is restarted. The FWAIT instruction forces instruction execution to wait until BUSY# is inactive for 10 CLK2 periods. This instruction should be executed in the interrupt routine used to stop the CLK2 signal.

2. When CLK2 is restarted, the 83S87 must be restarted in the same CLK phase as it was stopped. Therefore, the 83S87 must be restarted in such a way to ensure CLK2 is synchronized with the 386SX. Failure to restart CLK2 properly can result in a system READY# hang or Coprocessor inoperability.

The following figure illustrates the 83S87 restrictions for stopping the CLK2 input while the CPU CLK2 input remains running:



82S87:
The 82S87's low power is achieved automatically by going into idle mode after an instruction is finished executing (BUSY# not active). The coprocessor automatically restarts its clocks when the WRITE# input goes active.

NOTE: The Cyrix 82S87 will not operate in a design which assumes a synchronous bus.

Preliminary Electrical Specifications for CX-83S87-25
(Addendum to 83S87 User's Manual, Chapter 7)

7.0 Electrical Specifications

7.1 Absolute Maximum Ratings

The following table lists absolute maximum ratings for the CX-83S87-25 device. Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and do not imply that operations under any conditions other than those listed under "Recommended Operating Conditions" is possible. Exposure to conditions beyond the "Absolute Maximum Ratings" will (1) reduce device reliability and (2) result in premature failure even when there is no immediately apparent sign of failure. Prolonged exposure to conditions at or near "Absolute Maximum Ratings" may also result in reduced useful life and reliability.

Parameter	Min.	Max.	Units	Notes
Case Temperature	0	+100	°C	Power Applied
Storage Temperature	-65	+150	°C	No Bias
Supply Voltage, Vcc	-0.5	+6.0	Volts	With respect to Vss
Voltage On Any Pin	-0.5	Vcc+0.5	Volts	With respect to Vss
Power Dissipation		1.8	Watts	Tc = room temp, Vcc = 5.25 volts, CLK2 = 50 MHz
Input Clamp Current, Iik		10	mA	Vi<Vss or Vi>Vcc
Output Clamp Current, Iok		25	mA	Vo<Vss or Vo>Vcc

7.2 Recommended Operating Conditions

Symbol	Parameter	Min.	Max.	Units	Notes
Tc	Case Temperature	0	+85	°C	Power Applied
Vcc	Supply Voltage	+4.75	+5.25	Volts	With respect to Vss
Vih	High Level Input Voltage	2.0	Vcc+0.3	Volts	
Vil	Low Level Input Voltage	-0.3	0.8	Volts	
Ioh	High Level Output Current		-1.0	mA	Voh=Voh(min)
Iol	Low Level Output Current		+4.0	mA	Vol=Vol(max)
Iik	Input Clamp Current		+/-10	mA	Vi<Vss or Vi>Vcc
Iok	Ouput Clamp Current		+/-25	mA	Vo<Vss or Vo>Vcc

7.3 DC Electrical Characteristics

Symbol	Parameter	Min.	Max.	Units	Notes
Vcl	Clock Input Low	0.0	0.8	Volts	
Vch	Clock Input High	3.7	Vcc	Volts	
Vol	Output Low Voltage		+0.45	Volts	Iol=4.0 mA
Voh	Output High Voltage	2.4		Volts	Ioh=-1.0 mA
Icci	Supply Current - Idle		12	mA	CLK2=50 MHz
Iccr	Supply Current - Continuous Execution		350	mA	CLK2=50 MHz
Ili	Input Leakage Current		+/-15	µA	0<Vin<Vcc
Ilo	I/O Leakage Current		+/-15	µA	0.45<Vo<Vcc
Cin	Input Capacitance		10	pF	fc=1 MHz
Co	I/O Capacitance		12	pF	fc=1 MHz
Cclk	Clock Capacitance		15	pF	fc=1 MHz

7.4 Switching Characteristics

The following table summarizes the timing requirements and characteristics of the CX-83S87-25 device. All timings are measured at 1.5 volts unless otherwise specified. Figure numbers refer to the timing diagrams shown in Chapter 7 of the 83S87 User's Manual.

Symbol	Pin	Parameter	Min.	Max.	Units	Figure	Notes
t1	CPUCLK2	Period	20	500	nS	7.1	2.0V
t2a	CPUCLK2	High Time	7		nS	7.1	2.0V
t2b	CPUCLK2	High Time	4		nS	7.1	3.7V
t3a	CPUCLK2	Low Time	7		nS	7.1	2.0V
t3b	CPUCLK2	Low Time	5		nS	7.1	0.8V
t4	CPUCLK2	Fall Time		7	nS	7.1	3.7V to 0.8V
t5	CPUCLK2	Rise Time		7	nS	7.1	0.8V to 3.7V
t7	READYO#	Output Delay	3	19	nS	7.2	Cl=75 pF
t7	PEREQ	Output Delay	4	24	nS	7.2	Cl=75 pF
t7	BUSY#	Output Delay	4	24	nS	7.2	Cl=75 pF
t7	ERROR#	Output Delay	4	24	nS	7.2	Cl=75 pF
t8	D(31-0)	Output Delay	0	27	nS	7.3	Cl=70 pF
t10	D(31-0)	Setup Time	8		nS	7.3	
t11	D(31-0)	Hold Time	11		nS	7.3	
t12	D(31-0)	Float Time	5	24	nS	7.3	Cl=120 pF
t13	READYO#	Float Time	1	35	nS	7.5	Cl=75 pF
t13	PEREQ	Float Time	1	35	nS	7.5	Cl=75 pF
t13	BUSY#	Float Time	1	35	nS	7.5	Cl=75 pF
t13	ERROR#	Float Time	1	35	nS	7.5	Cl=75 pF
t14	ADS#	Setup Time	14		nS	7.3	
t15	ADS#	Hold Time	4		nS	7.3	
t16	READY#	Setup Time	8		nS	7.3	
t16	CMD0#,	Setup Time	12		nS	7.3	
t16	NPS1#	Setup Time	12		nS	7.3	
t16	NPS2	Setup Time	12		nS	7.3	
t16	W/R#	Setup Time	12		nS	7.3	
t16	STEN	Setup Time	12		nS	7.3	
t17	READY#	Hold Time	4		nS	7.3	
t17	CMD0#	Hold Time	0		nS	7.3	
t17	NPS1#	Hold Time	0		nS	7.3	
t17	NPS2	Hold Time	0		nS	7.3	
t17	W/R#	Hold Time	0		nS	7.3	
t17	STEN	Hold Time	2		nS	7.3	
t18	RESETIN	Setup Time	10		nS	7.4	
t19	RESETIN	Hold Time	3		nS	7.4	

Preliminary Electrical Specifications for CX-83D87-40
(Addendum to 83D87 User's Manual, Chapter 7)

7.0 Electrical Specifications

7.1 Absolute Maximum Ratings

The following table lists absolute maximum ratings for the CX-83D87-40 device. Stresses beyond those listed under "Absolute Maximum Ratings" may couse permanent damage to the device. These ore stress ratings only and do not imply that operations under any conditions other than those listed under "Recommended Operating Conditions" is possible. Exposure to conditions beyond the "Absolute Maximum Ratings" will (1) reduce device reliability and (2) result in premature failure even when there is no immediately apparent sign of failure. Prolonged exposure to conditions at or near "Absolute Maximum Ratings" may also result in reduced useful life and reliability.

Parameter	Min.	Max.	Units	Notes
Case Temperature	0	+100	◦C	Power Applied
Storage Temperature	-65	+150	◦C	No Bias
Supply Voltage, Vcc	-0.5	+6.0	Volts	With respect to Vss
Voltage On Any Pin	-0.5	Vcc+0.5	Volts	With respect to Vss
Power Dissipation		2.5	Watts	Tc = room temp, Vcc = 5.25 volts, CLK2 = 80 MHz
Input Clamp Current, Iik		10	mA	Vi<Vss or Vi>Vcc
Output Clamp Current, Iok		25	mA	Vo<Vss or Vo>Vcc

7.2 Recommended Operating Conditions

Symbol	Parameter	Min.	Max.	Units	Notes
Tc	Case Temperature	0	+85	◦C	Power Applied
Vcc	Supply Voltage	+4.75	+5.25	Volts	With respect to Vss
Vih	High Level Input Voltage	2.0	Vcc+0.3	Volts	
Vil	Low Level Input Voltage	-0.3	0.8	Volts	
Ioh	High Level Output Current		-1.0	mA	Voh=Voh(min)
Iol	Low Level Output Current		+4.0	mA	Vol=Vol(max)
Iik	Input Clamp Current		+/-10	mA	Vi<Vss or Vi>Vcc
Iok	Ouput Clamp Current		+/-25	mA	Vo<Vss or Vo>Vcc

7.3 DC Electrical Characteristics

Symbol	Parameter	Min.	Max.	Units	Notes
Vcl	Clock Input Low	0.0	0.8	Volts	
Vch	Clock Input High	3.7	Vcc	Volts	
Vol	Output Low Voltage		+0.45	Volts	Iol=4.0 mA
Voh	Output High Voltage	2.4		Volts	Ioh=-1.0 mA
Iccr	Supply Current - Continuous Execution		475	mA	CLK2=80 MHz
Ili	Input Leakage Current		+/-15	µA	0<Vin<Vcc
Ilo	I/O Leakage Current		+/-15	µA	0.45<Vo<Vcc
Cin	Input Capacitance		10	pF	fc=1 MHz
Co	I/O Capacitance		12	pF	fc=1 MHz
Cclk	Clock Capacitance		15	pF	fc=1 MHz

7.4 Switching Characteristics

The following table summarizes the timing requirements and characteristics of the CX-83D87-40 device. All timings are measured at 1.5 volts unless otherwise specified. Figure numbers refer to the timing diagrams shown in Chapter 7 of the 83D87 User's Manual.

Symbol	Pin	Parameter	Min.	Max.	Units	Figure	Notes
t1	CPUCLK2	Period	12.5	500	nS	7.1	2.0V
t2a	CPUCLK2	High Time	6		nS	7.1	2.0V
t2b	CPUCLK2	High Time	3		nS	7.1	3.7V
t3a	CPUCLK2	Low Time	6		nS	7.1	2.0V
t3b	CPUCLK2	Low Time	3		nS	7.1	0.8V
t4	CPUCLK2	Fall Time		4	nS	7.1	3.7V to 0.8V
t5	CPUCLK2	Rise Time		4	nS	7.1	0.8V to 3.7V
t7	READYO#	Output Delay	3	13	nS	7.2	Cl=75 pF
t7	PEREQ	Output Delay	3	15	nS	7.2	Cl=75 pF
t7	BUSY#	Output Delay	3	15	nS	7.2	Cl=75 pF
t7	ERROR#	Output Delay	3	15	nS	7.2	Cl=75 pF
t8	D(31-0)	Output Delay	0	19	nS	7.3	Cl=50 pF
t10	D(31-0)	Setup Time	5		nS	7.3	
t11	D(31-0)	Hold Time	6		nS	7.3	
t12	D(31-0)	Float Time	2	19	nS	7.3	Cl=120 pF
t13	READYO#	Float Time	1	30	nS	7.5	Cl=75 pF
t13	PEREQ	Float Time	1	30	nS	7.5	Cl=75 pF
t13	BUSY#	Float Time	1	30	nS	7.5	Cl=75 pF
t13	ERROR#	Float Time	1	30	nS	7.5	Cl=75 pF
t14	ADS#	Setup Time	11		nS	7.3	
t15	ADS#	Hold Time	4		nS	7.3	
t16	READY#	Setup Time	7		nS	7.3	
t16	CMD0#	Setup Time	7		nS	7.3	
t16	NPS1#	Setup Time	7		nS	7.3	
t16	NPS2	Setup Time	7		nS	7.3	
t16	W/R#	Setup Time	7		nS	7.3	
t16	STEN	Setup Time	10		nS	7.3	
t17	READY#	Hold Time	3		nS	7.3	
t17	CMD0#	Hold Time	0		nS	7.3	
t17	NPS1#	Hold Time	0		nS	7.3	
t17	NPS2	Hold Time	0		nS	7.3	
t17	W/R#	Hold Time	0		nS	7.3	
t17	STEN	Hold Time	2		nS	7.3	
t18	RESETIN	Setup Time	5		nS	7.4	
t19	RESETIN	Hold Time	3		nS	7.4	