MOTOROLA
M68000 FAMILY
Programmer's Reference Manual

(Includes CPU32 Instructions)

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

LIST OF FIGURES

LIST OF FIGURES (Concluded)

LIST OF TABLES

LIST OF TABLES (Continued)

SECTION 5
FLOATING POINT INSTRUCTIONS

This section contains information about the floating-point instructions for the MC68881, MC68882, and MC68040. In this section, all references to the MC68040 do not include the MC68LC040 and MC68EC040. Each instruction is described in detail, and the instruction descriptions are arranged in alphabetical order by instruction mnemonic.

All floating-point instructions apply to the MC68881 and MC68882 processors. The MC68040 directly supports part of the floating-point instructions through hardware. It indirectly supports the remainder by providing special traps and/or stack frames for the unimplemented instructions and data types. The following identification is noted under the instruction title for the MC68040:

Directly Supported—(MC6888X, MC68040)

Software Supported—(MC6888X, MC68040FPSW)

For all MC68040 floating-point instructions, the coprocessor ID field must be 001.

Table 5-1 lists the floating-point instructions directly supported by the MC68040, and Table 5-2 lists the floating-point instructions indirectly supported.

Table 5-1. Directly Supported Floating-Point Instructions

Mnemonic	Description
FABS	Floating-Point Absolute Value
FADD	Floating-Point Add
FBcc	Floating-Point Branch Conditionally
FCMP	Floating-Point Compare
FDBcc	Floating-Point Test Condition, Decrement, and Branch
FDIV	Floating-Point Divide
FMOVE	Move Floating-Point Data Register
FMOVE	Move Floating-Point System Control Register
FMOVEM	Move Multiple Floating-Point System Data Register
FMOVEM	Move Multiple Floating-Point Control Data Register
FMUL	Floating-Point Multiply
FNEG	Floating-Point Negate
FNOP	No Operation
FRESTORE*	Restore Internal Floating-Point State*
FSAVE*	Save Internal Floating-Point State*
FScc	Set According to Floating-Point Condition
FSORT	Floating-Point Square Root
FSUB	Floating-Point Subtract
FSGLDIV	Floating-Point Single-Precision Divide
FSFLMUL	Floating-Point Single-Precision Multiply
FTRAPcc	Trap on Floating-Point Condition
FTST	Test Floating-Point Operand

*These are privileged instructions; refer to **Section 6 Supervisor (Privaleged) Instructions** for detailed information.

Table 5-2. Indirectly Supported Floating-Point Instructions

Mnemonic	Description
FACOS	Floating-Point Arc Cosine
FASIN	Floating-Point Arc Sine
FATAN	Floating-Point Arc Tangent
FATANH	Floating-Point Hyperbolic Arc Tangent
FCOS	Floating-Point Cosine
FCOSH	Floating-Point Hyperbolic Cosine
FETOX	Floating-Point e^x
FETOXM1	Floating-Point $e^x - 1$
FGETEXP	Floating-Point Get Exponent
FGETMAN	Floating-Point Get Mantissa
FINT	Floating-Point Integer Part
FINTRZ	Floating-Point Integer Part, Round-to- Zero
FLOG10	Floating-Point Log10
FLOG2	Floating-Point Log2
FLOGN	Floating-Point Loge
FLOGNP1	Floating-Point $\log_e (x + 1)$
FMOD	Floating-Point Modulo Remainder
FMOVECR	Floating-Point Move Constant ROM
FREM	Floating-Point IEEE Remainder
FSCALE	Floating-Point Scale Exponent
FSIN	Floating-Point Sine
FSINCOS	Floating-Point Simultaneous Sine and Cosine
FSINH	Floating-Point Hyperbolic Sine
FTAN	Floating-Point Tangent
FTANH	Floating-Point Hyperbolic Tangent
FTENTOX	Floating-Point 10^x
FTWOTOX	Floating-Point 2^x

FABS Floating-Point Absolute Value (MC6888X, MC68040) FABS

Operation: Absolute Value of Source → FPn

Assembler Syntax:

FABS. < fmt >	< ea > ,FPn
FABS.X	FPm,FPn
FABS.X	FPn
*FrABS. < fmt >	< ea > ,FPn
*FrABS.X	FPm,FPn
*FrABS.X	Pn

where r is rounding precision, S or D

*Supported by MC68040 only.

Attributes: Format = (Byte, Word, Long, Single, Quad, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and stores the absolute value of that number in the destination floating-point data register.

FABS will round the result to the precision selected in the floating-point control register. FSABS and FDABS will round the result to single or double precision, respectively, regardless of the rounding precision selected in the floating-point control register.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Absolute Value	Absolute Value	Absolute Value

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information

FABS Floating-Point Absolute Value (MC6888X, MC68040) FABS

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**
OPERR	Cleared
OVFL	Cleared
UNFL	If the source is an extended-precision denormalized number, refer to exception processing in the appropriate user's manual; cleared otherwise.
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in exception processing; refer to the appropriate user's manual.

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
1	1	1	1	COPROCESSOR ID	0	0	0	EFFECTIVE ADDRESS MODE	EFFECTIVE ADDRESS REGISTER
0	R/M	0	SOURCE SPECIFIER		DESTINATION REGISTER			OPMODE	

FABS Floating-Point Absolute Value (MC6888X, MC68040) FABS

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field specifies the location of the source operand. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FABS Floating-Point Absolute Value (MC6888X, MC68040) FABS

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:

000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

*This encoding will cause an unimplemented data type exception in the MC68040 to allow emulation in software.

Destination Register field—Specifies the destination floating- point data register.

Opmode field—Specifies the instruction and rounding precision.

0011000	FABS	Rounding precision specified by the floating-point control register.
1011000	FSABS	Single-precision rounding specified.
1011100	FDABS	Double-precision rounding specified.

FACOS

Arc Cosine
(MC6888X, M68040FPSP)

FACOS

Operation: Arc Cosine of Source → FPn

Assembler Syntax:
FACOS. < fmt > < ea > ,FPn
FACOS.X FPm,FPn
FACOS.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the arc cosine of that number. Stores the result in the destination floating-point data register. This function is not defined for source operands outside of the range [– 1... + 1]; if the source is not in the correct range, a NAN is returned as the result and the OPERR bit is set in the floating- point status register. If the source is in the correct range, the result is in the range of [0...π].

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Arc Cosine	+ π/2	NAN

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is infinity, > + 1 or < – 1; cleared otherwise.
OVFL	Cleared
UNFL	Cleared
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FACOS

Arc Cosine
(MC6888X, M68040FPSP)

FACOS

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	1	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FACOS Arc Cosine (MC6888X, M68040FPSP) FACOS

Source Specifier field—Specifies the source register or data format.

If R/M = 0, specifies the source floating-point data register.

If R/M = 1, specifies the source data format:

000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is then written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FADD

Floating-Point Add
(MC6888X, MC68040)

FADD

Operation: Source + FPn → FPn

Assembler Syntax:
FADD. < fmt > < ea > ,FPn
FADD.X FPm,FPn
*FrADD. < fmt > < ea > ,FPn
*FrADD.X FPm,FPn
where r is rounding precision, S or D

*Supported by MC68040 only.

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and adds that number to the number contained in the destination floating-point data register. Stores the result in the destination floating-point data register.

FADD will round the result to the precision selected in the floating-point control register. FSADD and FDADD will round the result to single or double-precision, respectively, regardless of the rounding precision selected in the floating-point control register.

Operation Table:

DESTINATION		SOURCE[1]					
		+	In Range	–	+	Zero	–	+	Infinity	–
In Range	+	Add			Add			+ inf		– inf
	–									
Zero	+	Add			+ 0.0		0.0[2]	+ inf		– inf
	–				0.0[2]		– 0.0			
Infinity	+	+ inf			+ inf			+ inf		NAN[3]
	–	– inf			– inf			NAN‡		– inf

1. If either operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Returns + 0.0 in rounding modes RN, RZ, and RP; returns – 0.0 in RM.
3. Sets the OPERR bit in the floating-point status register exception byte.

FADD

Floating-Point Add
(MC6888X, MC68040)

FADD

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing.**	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers.**
	OPERR	Set if the source and the destination are opposite-signed infinities; cleared otherwise.
	OVFL	Refer to exception processing in the appropriate user's manual.
	UNFL	Refer to exception processing in the appropriate user's manual.
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in exception processing in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
1	1	1	1	COPROCESSOR ID	0	0	0	EFFECTIVE ADDRESS MODE	REGISTER
0	R/M	0	SOURCE SPECIFIER		DESTINATION REGISTER			OPMODE	

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands. If R/M = 0, this field is unused and should be all zeros.

FADD

Floating-Point Add
(MC6888X, MC68040)

FADD

If R/M = 1, specifies the location of the source operand location. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding will cause an unimplemented data type exception to allow emulation in software.

Destination Register field—Specifies the destination floating- point data register.

Opmode field—Specifies the instruction and rounding precision.

0100010	FADD	Rounding precision specified by the floating-point control register.
1100010	FSADD	Single-precision rounding specified.
1100110	FDADD	Double-precision rounding specified.

FASIN

Arc Sine
(MC6888X, M68040FPSP)

FASIN

Operation: Arc Sine of the Source → FPn

Assembler Syntax:

FASIN. < fmt >	< ea >	,FPn
FASIN.X	FPm	,FPn
FASIN.X	FPn	

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the arc sine of the number. Stores the result in the destination floating-point data register. This function is not defined for source operands outside of the range [– 1... + 1]; if the source is not in the correct range, a NAN is returned as the result and the OPERR bit is set in the floating- point status register. If the source is in the correct range, the result is in the range of [– π/2... + π/2].

Operation Table:

DESTINATION	SOURCE[1]		
	+ In Range –	+ Zero –	+ Infinity –
Result	Arc Sine	+ 0.0 – 0.0	NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

FASIN

Arc Sine
(MC6888X, M68040FPSP)

FASIN

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is infinity, > + 1 or < – 1; cleared otherwise
OVFL	Cleared
UNFL	Can be set for an underflow condition.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FASIN Arc Sine (MC6888X, M68040FPSP) FASIN

Effective Address field—Determines the addressing mode for external operands.

If R/M = 0, this field is unused and should be all zeros.

If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is then written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FATAN — Arc Tangent (MC6888X, M68040FPSP) — FATAN

Operation: Arc Tangent of Source → FPn

Assembler Syntax:

FATAN. < fmt >	< ea > ,FPn	
FATAN.X	FPm,FPn	
FATAN.X	FPm,FPnz	

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the arc tangent of that number. Stores the result in the destination floating-point data register. The result is in the range of [– π/2... + π/2].

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Arc Tangent	+ 0.0 – 0.0	+ π/2 –π/2

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Cleared
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FATAN

Arc Tangent
(MC6888X, M68040FPSP)

FATAN

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FATAN Arc Tangent (MC6888X, M68040FPSP) FATAN

Source Specifier field—Specifies the source register or data format.

If R/M = 0, specifies the source floating-point data register.

If R/M = 1, specifies the source data format:

000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is then written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FATANH Hyperbolic Arc Tangent (MC6888X, M68040FPSP) FATANH

Operation: Hyperbolic Arc Tangent of Source → FPn

Assembler Syntax:
FATANH. < fmt > < ea > ,FPn
FATANH.X FPm,FPn
FATANH.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the hyperbolic arc tangent of that value. Stores the result in the destination floating-point data register. This function is not defined for source operands outside of the range (– 1... + 1); and the result is equal to – infinity or + infinity if the source is equal to + 1 or – 1, respectively. If the source is outside of the range [– 1... + 1], a NAN is returned as the result, and the OPERR bit is set in the floating-point status register.

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Hyperbolic Arc Tangent			+ 0.0		– 0.0	NAN[2]		

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

FATANH Hyperbolic Arc Tangent (MC6888X, M68040FPSP) FATANH

Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the source is > + 1 or < – 1; cleared otherwise.
	OVFL	Cleared
	UNFL	Refer to underflow in the appropriate user's manual.
	DZ	Set if the source is equal to + 1 or – 1; cleared otherwise.
	INEX2	Refer to inexact result in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FATANH Hyperbolic Arc Tangent (MC6888X, M68040FPSP) FATANH

Effective Address field—Determines the addressing mode for external operands.

If R/M = 0, this field is unused and should be all zeros.

If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is then written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FBcc Floating-Point Branch Conditionally (MC6888X, MC68040) FBcc

Operation: If Condition True
Then PC + d_n → PC

Assembler Syntax: FBcc. < size > , < label >

Attributes: Size = (Word, Long)

Description: If the specified floating-point condition is met, program execution continues at the location (PC) + displacement. The displacement is a twos-complement integer that counts the relative distance in bytes. The value of the program counter used to calculate the destination address is the address of the branch instruction plus two. If the displacement size is word, then a 16- bit displacement is stored in the word immediately following the instruction operation word. If the displacement size is long word, then a 32-bit displacement is stored in the two words immediately following the instruction operation word. The conditional specifier cc selects any one of the 32 floating- point conditional tests as described in **3.6.2 Conditional Testing**.

Floating-Point Status Register:

Condition Codes:	Not affected.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Set if the NAN condition code is set and the condition selected is an IEEE nonaware test.
	SNAN	Not Affected.
	OPERR	Not Affected.
	OVF	Not Affected.
	UNFL	Not Affected.
	DZ	Not Affected.
	INEX2	Not Affected.
	INEX1	Not Affected.
Accrued Exception Byte:	The IOP bit is set if the BSUN bit is set in the exception byte. No other bit is affected.	

FBcc Floating-Point Branch Conditionally (MC6888X, MC68040) FBcc

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3 2 1 0
1	1	1	1	COPROCESSOR ID	0	1	SIZE	CONDITIONAL PREDICATE
16-BIT DISPLACEMENT OR MOST SIGNIFICANT WORD OF 32-BITDISPLACEMENT								
LEAST SIGNIFICANT WORD OF 32-BIT DISPLACEMENT (IF NEEDED)								

Instruction Fields:

Size field—Specifies the size of the signed displacement.
If Format = 0, then the displacement is 16 bits and is sign- extended before use.
If Format = 1, then the displacement is 32 bits.

Conditional Predicate field—Specifies one of 32 conditional tests as defined in **Table 3-23 Floating-Point Conditional Tests.**

NOTE

When a BSUN exception occurs, the main processor takes a preinstruction exception. If the exception handler returns without modifying the image of the program counter on the stack frame (to point to the instruction following the FBcc), then it must clear the cause of the exception (by clearing the NAN bit or disabling the BSUN trap), or the exception will occur again immediately upon return to the routine that caused the exception.

FCMP Floating-Point Compare (MC6888X, MC68040) FCMP

Operation: FPn – Source

Assembler Syntax: FCMP. < fmt > < ea > ,FPn
FCMP.X FPm,FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and subtracts the operand from the destination floating- point data register. The result of the subtraction is not retained, but it is used to set the floating-point condition codes as described in **3.6.2 Conditional Testing**.

Operation Table: The entries in this operation table differ from those of the tables describing most of the floating-point instructions. For each combination of input operand types, the condition code bits that may be set are indicated. If the name of a condition code bit is given and is not enclosed in brackets, then it is always set. If the name of a condition code bit is enclosed in brackets, then that bit is either set or cleared, as appropriate. If the name of a condition code bit is not given, then that bit is always cleared by the operation. The infinity bit is always cleared by the FCMP instruction since it is not used by any of the conditional predicate equations. Note that the NAN bit is not shown since NANs are always handled in the same manner (as described in **1.6.5 Not-A-Numbers**).

DESTINATION		SOURCE					
		+	In Range	–	+	Zero	–	+	Infinity	–
In Range	+	{NZ}	none	none	none	none	N	none	none
	–	N	{NZ}	N	N	N	N	none	none
Zero	+	N	none	Z	Z	N	none		
	–	N	none	NZ	NZ	N	none		
Infinity	+	none	none	none	none	Z	none		
	–	N	N	N	N	N	NZ		

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

FCMP Floating-Point Compare (MC6888X, MC68040) FCMP

Floating-Point Status Register:

Condition Codes: Affected as described in the preceding operation table.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Cleared
UNFL	Cleared
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in exception processing in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	1	1	0	0

FCMP Floating-Point Compare (MC6888X, MC68040) FCMP

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand location. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding in the MC68040 will cause an unimplemented data type exception to allow emulation in software.

Destination Register field—Specifies the destination floating- point data register.

FCOS Cosine FCOS

(MC6888X, M68040FPSP)

Operation: Cosine of Source → FPn

Assembler Syntax: FCOS. < fmt > < ea > ,FPn
FCOS.X FPm,FPn FCOS.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the cosine of that number. Stores the result in the destination floating-point data register. This function is not defined for source operands of ± infinity. If the source operand is not in the range of [– 2π... + 2π], then the argument is reduced to within that range before the cosine is calculated. However, large arguments may lose accuracy during reduction, and very large arguments (greater than approximately 10^{20}) lose all accuracy. The result is in the range of [– 1... + 1].

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Cosine			+ 1.0			NAN[2]		

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

FCOS

Cosine
(MC6888X, M68040FPSP)

FCOS

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the source operand is ± infinity; cleared otherwise.
	OVFL	Cleared
	UNFL	Cleared
	DZ	Cleared
	INEX2	Refer to inexact result in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	1	1	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FCOS Cosine (MC6888X, M68040FPSP) FCOS

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should contain zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FCOSH — Hyperbolic Cosine (MC6888X, M68040FPSP) — FCOSH

Operation: Hyperbolic Cosine of Source → FPn

Assembler Syntax: FCOSH. < fmt > < ea > ,FPn
FCOSH.X FPm,FPn FCOSH.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the hyperbolic cosine of that number. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Hyperbolic Cosine	+ 1.0	+ inf

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Cleared
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FCOSH Hyperbolic Cosine (MC6888X, M68040FPSP) FCOSH

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	1	1	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FCOSH Hyperbolic Cosine (MC6888X, M68040FPSP) FCOSH

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FDBcc — Floating-Point Test Condition, Decrement, and Branch (MC6888X, MC68040) — FDBcc

Operation:

If Condition True
  Then No Operation
Else Dn – 1 → Dn
  If Dn ≠ – 1
    Then PC + d_n → PC
  Else Execute Next Instruction

Assembler Syntax: FDBcc Dn, < label >

Attributes: Unsized

Description: This instruction is a looping primitive of three parameters: a floating-point condition, a counter (data register), and a 16-bit displacement. The instruction first tests the condition to determine if the termination condition for the loop has been met, and if so, execution continues with the next instruction in the instruction stream. If the termination condition is not true, the low-order 16 bits of the counter register are decremented by one. If the result is – 1, the count is exhausted, and execution continues with the next instruction. If the result is not equal to – 1, execution continues at the location specified by the current value of the program counter plus the sign-extended 16-bit displacement. The value of the program counter used in the branch address calculation is the address of the displacement word.

The conditional specifier cc selects any one of the 32 floating- point conditional tests as described in **3.6.2 Conditional Testing**.

Floating-Point Status Register:

Condition Codes:	Not affected.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Set if the NAN condition code is set and the condition selected is an IEEE nonaware test.
	SNAN	Not Affected.
	OPERR	Not Affected.
	OVFL	Not Affected.
	UNFL	Not Affected.
	DZ	Not Affected.
	NEX2	Not Affected.
	INEX1	Not Affected.
Accrued Exception Byte:	The IOP bit is set if the BSUN bit is set in the exception byte. No other bit is affected.	

FDBcc

Floating-Point Test Condition, Decrement, and Branch (MC6888X, MC68040)

FDBcc

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	0	0	1	COUNT REGISTER		
0	0	0	0	0	0	0	0	0	0	CONDITIONAL PREDICATE					
16-BIT DISPLACEMENT															

Instruction Fields:

Count Register field—Specifies data register that is used as the counter.

Conditional Predicate field—Specifies one of the 32 floating-point conditional tests as described in **3.6.2 Conditional Testing**.

Displacement field—Specifies the branch distance (from the address of the instruction plus two) to the destination in bytes.

NOTE

The terminating condition is like that defined by the UNTIL loop constructs of high-level languages. For example: FDBOLT can be stated as "decrement and branch until ordered less than".

There are two basic ways of entering a loop: at the beginning or by branching to the trailing FDBcc instruction. If a loop structure terminated with FDBcc is entered at the beginning, the control counter must be one less than the number of loop executions desired. This count is useful for indexed addressing modes and dynamically specified bit operations. However, when entering a loop by branching directly to the trailing FDBcc instruction, the count should equal the loop execution count. In this case, if the counter is zero when the loop is entered, the FDBcc instruction does not branch, causing a complete bypass of the main loop.

When a BSUN exception occurs, a preinstruction exception is taken by the main processor. If the exception handler returns without modifying the image of the program counter on the stack frame (to point to the instruction following the FDBcc), then it must clear the cause of the exception (by clearing the NAN bit or disabling the BSUN trap), or the exception will occur again immediately upon return to the routine that caused the exception.

FDIV — Floating-Point Divide (MC6888X, MC68040) — FDIV

Operation: FPn ÷ Source → FPn

Assembler Syntax:
FDIV. < fmt > < ea > ,FPn
FDIV.X FPm,FPn
*FrDIV. < fmt > < ea > ,FPn
*FrDIV.X FPm,FPn
where r is rounding precision, S or D

*Supported by MC68040 only

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and divides that number into the number in the destination floating-point data register. Stores the result in the destination floating-point data register.

FDIV will round the result to the precision selected in the floating-point control register. FSDIV and FDDIV will round the result to single or double precision, respectively, regardless of the rounding precision selected in the floating-point control register.

Operation Table:

DESTINATION		SOURCE[1]					
		+	In Range	–	+	Zero	–	+	Infinity	–
In Range	+	Divide			+ inf[2]	– inf[2]		+ 0.0	– 0.0	
	–				– inf[2]	+ inf[2]		– 0.0	+ 0.0	
Zero	+	+ 0.0		+ 0.0	NAN[3]			+ 0.0	– 0.0	
	–	– 0.0		+ 0.0				– 0.0	+ 0.0	
Infinity	+	+ inf		– inf	+ inf	– inf		NAN‡		
	–	– inf		+ inf	– inf	+ inf				

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the DZ bit in the floating-point status register exception byte.
3. Sets the OPERR bit in the floating-point status register exception byte.

FDIV

Floating-Point Divide
(MC6888X, MC68040)

FDIV

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set for 0 ÷ 0 or infinity ÷ infinity; cleared otherwise.
	OVFL	Refer to exception processing in the appropriate user's manual.
	UNFL	Refer to exception processing in the appropriate user's manual.
	DZ	Set if the source is zero and the destination is in range; cleared otherwise.
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in exception processing in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			OPMODE					

FDIV Floating-Point Divide (MC6888X, MC68040) FDIV

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand location. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding in the MC68040 will cause an unimplemented data type exception to allow emulation in software.

FDIV Floating-Point Divide (MC6888X, MC68040) FDIV

Destination Register field—Specifies the destination floating- point data register.

Opmode field—Specifies the instruction and rounding precision.

0100000	FDIV	Rounding precision specified by the floating- point control register.
1100000	FSDIV	Single-precision rounding specified.
1100100	FDDIV	Double-precision rounding specified.

FETOX e^x FETOX

(MC6888X, M68040FPSP)

Operation: $e^{Source} \rightarrow FPn$

Assembler Syntax: FETOX. < fmt > < ea > ,FPn
Syntax: FETOX.X FPm,FPn
Syntax: FETOX.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates e to the power of that number. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	e^x	+ 1.0	+ inf + 0.0

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FETOX e^x FETOX

(MC6888X, M68040FPSP)

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FETOX e^x FETOX

(MC6888X, M68040FPSP)

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier Field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FETOXM1 $e^x - 1$ FETOXM1

(MC6888X, M68040FPSP)

Operation: $e^{Source} - 1 \rightarrow FPn$

Assembler Syntax:
FETOXM1. < fmt > < ea > ,FPn
FETOXM1.X FPm,FPn
FETOXM1.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates e to the power of that number. Subtracts one from the value and stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	$e^x - 1$	+ 0.0 – 0.0	+ inf – 1.0

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FETOXM1 $e^x - 1$ FETOXM1

(MC6888X, M68040FPSP)

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FETOXM1 $e^x - 1$ FETOXM1

(MC6888X, M68040FPSP)

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier Field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FGETEXP Get Exponent (MC6888X, M68040FPSP) FGETEXP

Operation: Exponent of Source → FPn

Assembler Syntax:
FGETEXP. < fmt > < ea > ,FPn
FGETEXP.X FPm,FPn
FGETEXP.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and extracts the binary exponent. Removes the exponent bias, converts the exponent to an extended-precision floating- point number, and stores the result in the destination floating- point data register.

Operation Table:

DESTINATION	SOURCE[1]		
	+ In Range –	+ Zero –	+ Infinity –
Result	Exponent	+ 0.0 – 0.0	NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is ± infinity; cleared otherwise.
OVFL	Cleared
UNFL	Cleared
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FGETEXP Get Exponent (MC6888X, M68040FPSP) FGETEXP

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	1	1	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FGETEXP Get Exponent (MC6888X, M68040FPSP) FGETEXP

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FGETMAN Get Mantissa (MC6888X, M68040FPSP) FGETMAN

Operation: Mantissa of Source → FPn

Assembler Syntax:
FGETMAN. < fmt > < ea > ,FPn
FGETMAN.X FPm,FPn
FGETMAN.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and extracts the mantissa. Converts the mantissa to an extended-precision value and stores the result in the destination floating-point data register. The result is in the range [1.0...2.0] with the sign of the source mantissa, zero, or a NAN.

Operation Table:

DESTINATION	SOURCE[1]		
	+ In Range –	+ Zero –	+ Infinity –
Result	Mantissa	+ 0.0 – 0.0	NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is ± infinity; cleared otherwise.
OVFL	Cleared
UNFL	Cleared
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FGETMAN Get Mantissa (MC6888X, M68040FPSP) FGETMAN

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	1	1	1	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FGETMAN Get Mantissa (MC6888X, M68040FPSP) FGETMAN

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FINT

Integer Part
(MC6888X, M68040FPSP)

FINT

Operation: Integer Part of Source → FPn

Assembler Syntax:
FINT. < fmt > < ea > ,FPn
FINT.X FPm,FPn
FINT.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary), extracts the integer part, and converts it to an extended-precision floating-point number. Stores the result in the destination floating-point data register. The integer part is extracted by rounding the extended-precision number to an integer using the current rounding mode selected in the floating-point control register mode control byte. Thus, the integer part returned is the number that is to the left of the radix point when the exponent is zero, after rounding. For example, the integer part of 137.57 is 137.0 for the round-to-zero and round-to- negative infinity modes and 138.0 for the round-to-nearest and round-to-positive infinity modes. Note that the result of this operation is a floating-point number.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Integer	+ 0.0 – 0.0	+ inf – inf

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

FINT

Integer Part
(MC6888X, M68040FPSP)

FINT

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Cleared
	OVFL	Cleared
	UNFL	Cleared
	DZ	Cleared
	INEX2	Refer to inexact result in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	0	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FINT Integer Part (MC6888X, M68040FPSP) FINT

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FINTRZ Integer Part, Round-to-Zero (MC6888X, M68040FPSP) FINTRZ

Operation: Integer Part of Source → FPn

Assembler Syntax:
FINTRZ. < fmt > < ea > ,FPn
FINTRZ.X FPm,FPn
FINTRZ.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and extracts the integer part and converts it to an extended-precision floating-point number. Stores the result in the destination floating-point data register. The integer part is extracted by rounding the extended-precision number to an integer using the round-to-zero mode, regardless of the rounding mode selected in the floating-point control register mode control byte (making it useful for FORTRAN assignments). Thus, the integer part returned is the number that is to the left of the radix point when the exponent is zero. For example, the integer part of 137.57 is 137.0; the integer part of 0.1245 x 102 is 12.0. Note that the result of this operation is a floating-point number.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Integer, Forced Round-to- Zero	+ 0.0 – 0.0	+ inf – inf

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

FINTRZ Integer Part, Round-to-Zero (MC6888X, M68040FPSP) FINTRZ

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Cleared
UNFL	Cleared
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	0	1	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FINTRZ Integer Part, Round-to-Zero (MC6888X, M68040FPSP) FINTRZ

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If RM = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FLOG10 — Log_{10} (MC6888X, M68040FPSP) — FLOG10

Operation: Log_{10} of Source → FPn

Assembler Syntax:
FLOG10. < fmt > < ea > ,FPn
FLOG10.X FPm,FPn
FLOG10.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Convert the source operand to extended precision (if necessary) and calculates the logarithm of that number using base 10 arithmetic. Stores the result in the destination floating-point data register. This function is not defined for input values less than zero.

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Log_{10}		NAN[2]		– inf[3]		+ inf		NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.
3. Sets the DZ bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source operand is < 0; cleared otherwise.
OVFL	Cleared
UNFL	Cleared
DZ	Set if the source is ± 0; cleared otherwise
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FLOG10 — Log_{10} (MC6888X, M68040FPSP) — FLOG10

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	1	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FLOG10 Log_{10} (MC6888X, M68040FPSP) FLOG10

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FLOG2 — Log_2 (MC6888X, M68040FPSP) — FLOG2

Operation: Log_2 of Source → FPn

Assembler Syntax:
FLOG2. < fmt > < ea > ,FPn
FLOG2.X FPm,FPn
FLOG2.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the logarithm of that number using base two arithmetic. Stores the result in the destination floating- point data register. This function is not defined for input values less than zero.

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Log_2		NAN[2]		– inf[3]		+ inf		NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.
3. Sets the DZ bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is < 0; cleared otherwise
OVFL	Cleared
UNFL	Cleared
DZ	Set if the source is ± 0; cleared otherwise
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FLOG2 — Log_2 (MC6888X, M68040FPSP) — FLOG2

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	0	1	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FLOG2 — Log_2 (MC6888X, M68040FPSP) — FLOG2

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FLOGN Log_e (MC6888X, M68040FPSP) FLOGN

Operation: Log_e of Source → FPn

Assembler Syntax:
FLOGN. < fmt > < ea > ,FPn
FLOGN.X FPm,FPn
FLOGN.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the natural logarithm of that number. Stores the result in the destination floating-point data register. This function is not defined for input values less than zero.

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	ln(x)		NAN[2]		– inf[3]		+ inf		NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.
3. Sets the DZ bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source operand is < 0; cleared otherwise.
OVFL	Cleared
UNFL	Cleared
DZ	Set if the source is ± 0; cleared otherwise
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FLOGN

Log_e
(MC6888X, M68040FPSP)

FLOGN

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FLOGN Log_e (MC6888X, M68040FPSP) FLOGN

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FLOGNP1 — $Log_e (x + 1)$ (MC6888X, M68040FPSP) — FLOGNP1

Operation: Log_e of (Source + 1) → FPn

Assembler Syntax:
FLOGNP1. < fmt > < ea > ,FPn
FLOGNP1.X FPm,FPn
FLOGNP1.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary), adds one to that value, and calculates the natural logarithm of that intermediate result. Stores the result in the destination floating-point data register. This function is not defined for input values less than – 1.

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	ln(x + 1)		$ln(x + 1)^2$	+ 0.0		– 0.0	+ inf		NAN[2]3

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. If the source is – 1, sets the DZ bit in the floating-point status register exception byte and returns a NAN. If the source is < – 1, sets the OPERR bit in the floating-point status register exception byte and returns a NAN.
3. Sets the OPERR bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

FLOGNP1 $Log_e (x + 1)$ (MC6888X, M68040FPSP) FLOGNP1

Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the source operand is < – 1; cleared otherwise.
	OVFL	Cleared
	UNFL	Refer to underflow in the appropriate user's manual.
	DZ	Set if the source operand is – 1; cleared otherwise
	INEX2	Refer to inexact result in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FLOGNP1 $Log_e (x + 1)$ (MC6888X, M68040FPSP) FLOGNP1

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FMOD Modulo Remainder (MC6888X, M68040FPSP) FMOD

Operation: Modulo Remainder of (FPn ÷ Source) → FPn

Assembler Syntax: FMOD. < fmt > < ea > ,FPn
FMOD.X FPm,FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the modulo remainder of the number in the destination floating-point data register, using the source operand as the modulus. Stores the result in the destination floating-point data register and stores the sign and seven least significant bits of the quotient in the floating-point status register quotient byte (the quotient is the result of FPn ÷ Source). The modulo remainder function is defined as:

FPn – (Source x N)

where N = INT(FPn ÷ Source) in the round-to-zero mode.

The FMOD function is not defined for a source operand equal to zero or for a destination operand equal to infinity. Note that this function is not the same as the FREM instruction, which uses the round-to-nearest mode and thus returns the remainder that is required by the IEEE *Specification for Binary Floating-Point Arithmetic*.

Operation Table:

DESTINATION		SOURCE[1]		
		+ In Range –	+ Zero# –	+ Infinity –
In Range	+ –	Modulo Remainder	NAN[2]	FPn[3]
Zero	+ –	+ 0.0 – 0.0	NAN[2]	+ 0.0 – 0.0
Infinity	+ –	NAN[2]	NAN[2]	NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.
3. Returns the value of FPn before the operation. However, the result is processed by the normal instruction termination procedure to round it as required. Thus, an overflow and/or inexact result may occur if the rounding precision has been changed to a smaller size since the FPn value was loaded

FMOD Modulo Remainder (MC6888X, M68040FPSP) FMOD

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Loaded with the sign and least significant seven bits of the quotient (FPn ÷ Source). The sign of the quotient is the exclusive-OR of the sign bits of the source and destination operands.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is zero or the destination is infinity; cleared otherwise.
OVFL	Cleared
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, in the appropriate user's manual for inexact result on decimal input; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	1	1	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FMOD Modulo Remainder (MC6888X, M68040FPSP) FMOD

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register.

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Operation: Source → Destination

Assembler Syntax:
FMOVE. < fmt > < ea > ,FPn
FMOVE. < fmt > FPm, < ea >
FMOVE.P FPm, < ea > {Dn}
FMOVE.P FPm, < ea > {k}
*FrMOVE. < fmt > < ea > ,FPn
where r is rounding precision, S or D

*Supported by MC68040 only

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Moves the contents of the source operand to the destination operand. Although the primary function of this instruction is data movement, it is also considered an arithmetic instruction since conversions from the source operand format to the destination operand format are performed implicitly during the move operation. Also, the source operand is rounded according to the selected rounding precision and mode.

Unlike the MOVE instruction, the FMOVE instruction does not support a memory-to-memory format. For such transfers, it is much faster to utilize the MOVE instruction to transfer the floating- point data than to use the FMOVE instruction. The FMOVE instruction only supports memory-to-register, register-to- register, and register-to-memory operations (in this context, memory may refer to an integer data register if the data format is byte, word, long, or single). The memory-to-register and register- to-register operation uses a command word encoding distinctly different from that used by the register-to-memory operation; these two operation classes are described separately.

Memory-to-Register and Register-to-Register Operation: Converts the source operand to an extended-precision floating-point number (if necessary) and stores it in the destination floating-point data register. MOVE will round the result to the precision selected in the floating-point control register. FSMOVE and FDMOVE will round the result to single or double precision, respectively, regardless of the rounding precision selected in the floating-point control register. Depending on the source data format and the rounding precision, some operations may produce an inexact result. In the following table, combinations that can produce an inexact result are marked with a dot (·), but all other combinations produce an exact result.

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Rounding Precision	Source Format						
	B	W	L	S	D	X	P
Single			•		•	•	•
Double						•	•
Extended							•

Floating-Point Status Register (< ea > to Register):

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers.**
OPERR	Cleared
OVFL	Cleared
UNFL	Refer to exception processing in the appropriate user's manual if the source is an extended-precision denormalized number; cleared otherwise.
DZ	Cleared
INEX2	Refer to exception processing in the appropriate user's manual if < fmt > is L,D, or X; cleared otherwise.
INEX1	Refer to exception processing in the appropriate user's manual if < fmt > is P; cleared otherwise.

Accrued Exception Byte: Affected as described in exception processing in the appropriate user's manual.

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Instruction Format:

< EA > TO REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding in the MC68040 will cause an unimplemented data type exception to allow emulation in software.

Destination Register field—Specifies the destination floating- point data register.

Opmode field—Specifies the instruction and rounding precision.

0000000	FMOVE	Rounding precision specified by the floating-point control register.
1000000	FSMOVE	Single-precision rounding specified.
1000100	FDMOVE	Double-precision rounding specified.

Register-to-Memory Operation: Rounds the source operand to the size of the specified destination format and stores it at the destination effective address. If the format of the destination is packed decimal, a third operand is required to specify the format of the resultant string. This operand, called the k-factor, is a 7-bit signed integer (twos complement) and may be specified as an immediate value or in an integer data register. If a data register contains the k-factor, only the least significant seven bits are used, and the rest of the register is ignored.

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Floating-Point Status Register (Register-to-Memory):

Condition Codes:	Not affected.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
< fmt > is B, W, or L	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the source operand is infinity or if the destination size is exceeded after conversion and rounding; cleared otherwise.
	OVFL	Cleared
	UNFL	Cleared
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	Cleared
< fmt > is S, D, or X	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**
	OVFL	Refer to exception processing in the appropriate user's manual.
	UNFL	Refer to exception processing in the appropriate user's manual.
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	Cleared
< fmt > is P	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the k-factor > + 17 or the magnitude of the decimal exponent exceeds three digits; cleared otherwise.
	OVFL	Cleared
	UNFL	Cleared
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	Cleared
Accrued Exception Byte:	Affected as described in exception processing in the appropriate user's manual.	

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Instruction Format:

REGISTER—TO-MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	COPROCESSOR ID			1	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	1	1	DESTINATION FORMAT			SOURCE REGISTER				K-FACTOR (IF REQUIRED)					

Instruction Fields:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

*Only if < fmt > is byte, word, long, or single.

Destination Format field—Specifies the data format of the destination operand:

000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real with Static k-Factor (P{#k})*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
111 — Packed-Decimal Real with Dynamic k-Factor (P{Dn})*

*This encoding will cause an unimplemented data type exception in the MC68040 to allow emulation in software.

FMOVE Move Floating-Point Data Register (MC6888X, MC68040) FMOVE

Source Register field—Specifies the source floating-point data register.

k-Factor field—If the destination format is packed decimal, used to specify the format of the decimal string. For any other destination format, this field should be set to all zeros. For a static k-factor, this field is encoded with a twos-complement integer where the value defines the format as follows:

– 64 to 0—Indicates the number of significant digits to the right of the decimal point (FORTRAN "F" format).
+ 1 to + 17—Indicates the number of significant digits in the mantissa (FORTRAN "E" format).
+ 18 to + 63—Sets the OPERR bit in the floating-point status register exception byte and treated as + 17.

The format of this field for a dynamic k-factor is:

r r r 0 0 0 0

where "rrr" is the number of the main processor data register that contains the k-factor value.

The following table gives several examples of how the k-factor value affects the format of the decimal string that is produced by the floating-point coprocessor. The format of the string that is generated is independent of the source of the k-factor (static or dynamic).

k- Factor	Source Operand Value	Destination String
– 5	+ 12345.678765	+ 1.234567877E + 4
– 3	+ 12345.678765	+ 1.2345679E + 4
– 1	+ 12345.678765	+ 1.23457E + 4
0	+ 12345.678765	+ 1.2346E + 4
+ 1	+ 12345.678765	+ 1.E + 4
+ 3	+ 12345.678765	+ 1.23E + 4
+ 5	+ 12345.678765	+ 1.2346E + 4

FMOVE

Move Floating-Point System Control Register (MC6888X, MC68040)

FMOVE

Operation: Source → Destination

Assembler Syntax: FMOVE.L < ea > ,FPCR
FMOVE.L FPCR, < ea >

Attributes: Size = (Long)

Description: Moves the contents of a floating-point system control register (floating-point control register, floating-point status register, or floating-point instruction address register) to or from an effective address. A 32-bit transfer is always performed, even though the system control register may not have 32 implemented bits. Unimplemented bits of a control register are read as zeros and are ignored during writes (must be zero for compatibility with future devices). For the MC68881, this instruction does not cause pending exceptions (other than protocol violations) to be reported. Furthermore, a write to the floating-point control register exception enable byte or the floating-point status register exception status byte cannot generate a new exception, regardless of the value written.

Floating-Point Status Register: Changed only if the destination is the floating-point status register, in which case all bits are modified to reflect the value of the source operand.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
1	0	dr	REGISTER SELECT			0	0	0	0	0	0	0	0	0	0

FMOVE

Move Floating-Point System Control Register (MC6888X, MC68040)

FMOVE

Instruction Fields:

Effective Address field—(Memory-to-Register) All addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	# < data >	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d_{16},An)	101	reg. number:An	(d_{16},PC)	111	010
(d_8,An,Xn)	110	reg. number:An	(d_8,PC,Xn)	111	011
(bd,An,Xn)	110	reg. number:An	(bd,PC,Xn)	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

*Only if the source register is the floating-point instruction address register.

Effective Address field—(Register-to-Memory) Only alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An*	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	# < data >	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d_{16},An)	101	reg. number:An	(d_{16},PC)	—	—
(d_8,An,Xn)	110	reg. number:An	(d_8,PC,Xn)	—	—
(bd,An,Xn)	110	reg. number:An	(bd,PC,Xn)	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

*Only if the source register is the floating-point instruction address register.

FMOVE Move Floating-Point System Control Register (MC6888X, MC68040) FMOVE

dr field—Specifies the direction of the data transfer.
0 — From < ea > to the specified system control register.
1 — From the specified system control register to < ea > .

Register Select field—Specifies the system control register to be moved:
100 Floating-Point Control Register
010 Floating-Point Status Register
001 Floating-Point Instruction Address Register

FMOVECR Move Constant ROM (MC6888X, M68040FPSP) FMOVECR

Operation: ROM Constant → FPn

Assembler Syntax: FMOVECR.X # < ccc > ,FPn

Attributes: Format = (Extended)

Description: Fetches an extended-precision constant from the floating- point coprocessor on-chip ROM, rounds the mantissa to the precision specified in the floating-point control register mode control byte, and stores it in the destination floating-point data register. The constant is specified by a predefined offset into the constant ROM. The values of the constants contained in the ROM are shown in the offset table at the end of this description.

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing.**	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Cleared
	OPERR	Cleared
	OVFL	Cleared
	UNFL	Cleared
	DZ	Cleared
	INEX2	Refer to inexact result in the appropriate user's manual.
	INEX1	Cleared
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	DESTINATION REGISTER			ROM OFFSET						

FMOVECR Move Constant ROM (MC6888X, M68040FPSP) FMOVECR

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Destination Register field—Specifies the destination floating- point data register.

ROM Offset field—Specifies the offset into the floating-point coprocessor on-chip constant ROM where the desired constant is located. The offsets for the available constants are as follows:

Offset	Constant
$00	π
$0B	$Log_{10}(2)$
$0C	e
$0D	$Log_2(e)$
$0E	$Log_{10}(e)$
$0F	0.0
$30	1n(2)
$31	1n(10)
$32	100
$33	10^1
$34	10^2
$35	10^4
$36	10^8
$37	10^{16}
$38	10^{32}
$39	10^{64}
$3A	10^{128}
$3B	10^{256}
$3C	10^{512}
$3D	10^{1024}
$3E	10^{2048}
$3F	10^{4096}

The on-chip ROM contains other constants useful only to the on- chip microcode routines. The values contained at offsets other than those defined above are reserved for the use of Motorola and may be different on various mask sets of the floating-point coprocessor. These undefined values yield the value 0.0 in the M68040FPSP.

FMOVEM Move Multiple Floating-Point Data Registers (MC6888X, MC68040) FMOVEM

Operation: Register List → Destination
Source → Register List

Assembler Syntax: FMOVEM.X < list > , < ea >
FMOVEM.X Dn, < ea >
FMOVEM.X < ea > , < list > FMOVEM.X < ea > ,Dn

Attributes: Format = (Extended)

Description: Moves one or more extended-precision numbers to or from a list of floating-point data registers. No conversion or rounding is performed during this operation, and the floating-point status register is not affected by the instruction. For the MC68881, this instruction does not cause pending exceptions (other than protocol violations) to be reported. Furthermore, a write to the floating- point control register exception enable byte or the floating-point status register exception status byte connot generate a new exception, despite the value written.

Any combination of the eight floating-point data registers can be transferred, with the selected registers specified by a user- supplied mask. This mask is an 8-bit number, where each bit corresponds to one register; if a bit is set in the mask, that register is moved. The register select mask may be specified as a static value contained in the instruction or a dynamic value in the least significant eight bits of an integer data register (the remaining bits of the register are ignored).

FMOVEM allows three types of addressing modes: the control modes, the predecrement mode, or the postincrement mode. If the effective address is one of the control addressing modes, the registers are transferred between the processor and memory starting at the specified address and up through higher addresses. The order of the transfer is from FP0 – FP7.

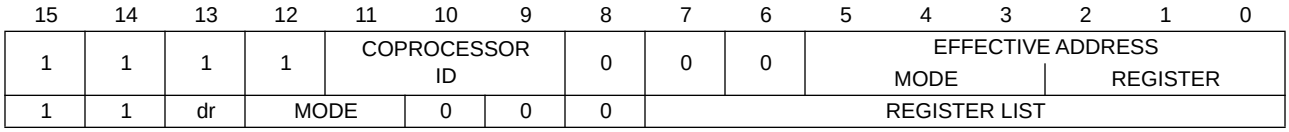
FMOVEM Move Multiple Floating-Point Data Registers (MC6888X, MC68040) FMOVEM

If the effective address is the predecrement mode, only a register- to-memory operation is allowed. The registers are stored starting at the address contained in the address register and down through lower addresses. Before each register is stored, the address register is decremented by 12 (the size of an extended-precision number in memory) and the floating-point data register is then stored at the resultant address. When the operation is complete, the address register points to the image of the last floating- point data register stored. The order of the transfer is from FP7 – FP0.

If the effective address is the postincrement mode, only a memory- to-register operation is allowed. The registers are loaded starting at the specified address and up through higher addresses. After each register is stored, the address register is incremented by 12 (the size of an extended-precision number in memory). When the operation is complete, the address register points to the byte immediately following the image of the last floating-point data register loaded. The order of the transfer is the same as for the control addressing modes: FP0 – FP7.

Floating-Point Status Register: Not Affected. Note that the FMOVEM instruction provides the only mechanism for moving a floating- point data item between the floating-point unit and memory without performing any data conversions or affecting the condition code and exception status bits.

Instruction Format:



FMOVEM Move Multiple Floating-Point Data Registers (MC6888X, MC68040) FMOVEM

Instruction Fields:

Effective Address field—(Memory-to-Register) Only control addressing modes or the postincrement addressing mode can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

Effective Address field—(Register-to-Memory) Only control alterable addressing modes or the predecrement addressing mode can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

FMOVEM Move Multiple Floating-Point Data Registers (MC6888X, MC68040) FMOVEM

dr field—Specifies the direction of the transfer.
0 — Move the listed registers from memory to the floating-point unit.
1 — Move the listed registers from the floating-point unit to memory.

Mode field—Specifies the type of the register list and addressing mode.
00 — Static register list, predecrement addressing mode.
01 — Dynamic register list, predecrement addressing mode.
10 — Static register list, postincrement or control addressing mode.
11 — Dynamic register list, postincrement or control addressing mode.

Register List field:

Static list—contains the register select mask. If a register is to be moved, the corresponding bit in the mask is set as shown below; otherwise it is clear.

Dynamic list—contains the integer data register number, rrr, as listed in the following table:

List Type	Register List Format							
Static, – (An)	FP7	FP6	FP5	FP4	FP3	FP2	FP1	FP0
Static, (An) + , or Control	FP0	FP1	FP2	FP3	FP4	FP5	FP6	FP7
Dynamic	0	r	r	r	0	0	0	0

The format of the dynamic list mask is the same as for the static list and is contained in the least significant eight bits of the specified main processor data register.

Programming Note: This instruction provides a very useful feature, dynamic register list specification, that can significantly enhance system performance. If the calling conventions used for procedure calls utilize the dynamic register list feature, the number of floating-point data registers saved and restored can be reduced.

To utilize the dynamic register specification feature of the FMOVEM instruction, both the calling and the called procedures must be written to communicate information about register usage. When one procedure calls another, a register mask must be passed to the called procedure to indicate which registers must not be altered upon return to the calling procedure. The called procedure then saves only those registers that are modified and are already in use. Several techniques can be used to utilize this mechanism, and an example follows.

FMOVEM Move Multiple Floating-Point Data Registers (MC6888X, MC68040) FMOVEM

In this example, a convention is defined by which each called procedure is passed a word mask in D7 that identifies all floating-point registers in use by the calling procedure. Bits 15 – 8 identify the registers in the order FP0 – FP7, and bits 7 – 0 identify the registers in the order FP7 – FP0 (the two masks are required due to the different transfer order used by the predecrement and postincrement addressing modes). The code used by the calling procedure consists of simply moving the mask (which is generated at compile time) for the floating-point data registers currently in use into D7:

Calling procedure...

```
MOVE.W    #ACTIVE,D7      Load the list of FP registers that are
                          in use.
BSR       PROC_2
```

The entry code for all other procedures computes two masks. The first mask identifies the registers in use by the calling procedure that are used by the called procedure (and therefore saved and restored by the called procedure). The second mask identifies the registers in use by the calling procedure that are used by the called procedure (and therefore not saved on entry). The appropriate registers are then stored along with the two masks:

Called procedure...

```
MOVE.W    D7,D6           Copy the list of active registers.
AND.W     #WILL_USE,D7    Generate the list of doubly-used
                          registers.
FMOVEM    D7, – (A7)      Save those registers.
MOVE.W    D7, – (A7)      Save the register list.
EOR.W     D7,D6           Generate the list of not saved active
                          registers.
MOVE.W    D6, – (A7)      Save it for later use.
```

If the second procedure calls a third procedure, a register mask is passed to the third procedure that indicates which registers must not be altered by the third procedure. This mask identifies any registers in the list from the first procedure that were not saved by the second procedure, plus any registers used by the second procedure that must not be altered by the third procedure.

FMOVEM Move Multiple Floating-Point Data Registers (MC6888X, MC68040) FMOVEM

An example of the calculation of this mask is as follows:

Nested calling sequence...

```
MOVE.W    UNSAVED (A7),D7  Load the list of active registers not
                           saved at entry.
OR.W      #WILL_USE,D7     Combine with those active at this time
BSR       PROC_3
```

Upon return from a procedure, the restoration of the necessary registers follows the same convention, and the register mask generated during the save operation on entry is used to restore the required floating-point data registers:

Return to caller...

```
ADDQ.L    #2,A7            Discard the list of registers not saved.
MOVE.B    (A7) + ,D7       Get the saved register list (pop word,
                           use byte).
FMOVEM    (A7) + ,D7       Restore the registers.
     *
     *
     *
RTS                        Return to the calling routine.
```

FMOVEM Move Multiple Floating-Point Control Registers (MC6888X, MC68040) FMOVEM

Operation: Register List → Destination
Source → Register List

Assembler Syntax: FMOVEM.L < list > , < ea >
FMOVEM.L < ea > , < list >

Attributes: Size = (Long)

Description: Moves one or more 32-bit values into or out of the specified system control registers. Any combination of the three system control registers may be specified. The registers are always moved in the same order, regardless of the addressing mode used; the floating-point control register is moved first, followed by the floating-point status register, and the floating-point instruction address register is moved last. If a register is not selected for the transfer, the relative order of the transfer of the other registers is the same. The first register is transferred between the floating-point unit and the specified address, with successive registers located up through higher addresses.

For the MC68881, this instruction does not cause pending exceptions (other than protocol violations) to be reported. Furthermore, a write to the floating-point control register exception enable byte or the floating-point status register exception status byte connot generate a new exception, despite the value written.

When more than one register is moved, the memory or memory- alterable addressing modes can be used as shown in the addressing mode tables. If the addressing mode is predecrement, the address register is first decremented by the total size of the register images to be moved (i.e., four times the number of registers), and then the registers are transferred starting at the resultant address. For the postincrement addressing mode, the selected registers are transferred to or from the specified address, and then the address register is incremented by the total size of the register images transferred. If a single system control register is selected, the data register direct addressing mode may be used; if the only register selected is the floating-point instruction address register, then the address register direct addressing mode is allowed. Note that if a single register is selected, the opcode generated is the same as for the FMOVE single system control register instruction.

FMOVEM Move Multiple Floating-Point Control Registers (MC6888X, MC68040) FMOVEM

Floating-Point Status Register: Changed only if thedestinationlist includes the floating-point status register in which case all bits are modified to reflect the value of the source register image.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
1	0	dr	REGISTER LIST			0	0	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Determines the addressing mode for the operation.

Memory-to-Register—Only control addressing modes or the postincrement addressing mode can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An**	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if a single floating-point instruction address register, floating-point status register, or floating-point control register is selected.

**Only if the floating-point instruction address register is the single register selected.

FMOVEM Move Multiple Floating-Point Control Registers (MC6888X, MC68040) FMOVEM

Register-to-Memory—Only control alterable addressing modes or the predecrement addressing mode can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An**	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

*Only if a single floating-point control register is selected.
**Only if the floating-point instruction address register is the single register selected.

dr field—Specifies the direction of the transfer.
0 — Move the listed registers from memory to the floating-point unit.
1 — Move the listed registers from the floating-point unit to memory.

Register List field—Contains the register select mask. If a register is to be moved, the corresponding bit in the list is set; otherwise, it is clear. At least one register must be specified.

Bit Number	Register
12	Floating-Point Control Register
11	Floating-Point Status Register
10	Floating-Point Instruction Address Register

FMUL — Floating-Point Multiply (MC6888X, MC68040) — FMUL

Operation: Source x FPn → FPn

Assembler Syntax:
FMUL. < fmt > < ea > ,FPn
FMUL.X FPm,FPn
*FrMUL < fmt > < ea > ,FPn
*FrMUL.X FPm,FPn
where r is rounding precision, S or D
*Supported by MC68040 only

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and multiplies that number by the number in the destination floating-point data register. Stores the result in the destination floating-point data register.

FMUL will round the result to the precision selected in the floating-point control register. FSMUL and FDMUL will round the result to single or double precision, respectively, regardless of the rounding precision selected in the floating-point control register.

Operation Table:

DESTINATION		SOURCE[1]					
		+	In Range	–	+	Zero	–	+	Infinity	–
In Range	+	Multiply			+ 0.0		– 0.0	+ inf		– inf
	–				– 0.0		+ 0.0	– inf		+ inf
Zero	+	+ 0.0		– 0.0	+ 0.0		– 0.0	NAN[2]		
	–	– 0.0		+ 0.0	– 0.0		+ 0.0			
Infinity	+	+ inf		– inf	NAN[2]			+ inf		– inf
	–	– inf		+ inf				– inf		+ inf

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

FMUL

Floating-Point Multiply
(MC6888X, MC68040)

FMUL

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set for 0 x infinity; cleared otherwise.
	OVFL	Refer to exception processing in the appropriate user's manual.
	UNFL	Refer to exception processing in the appropriate user's manual.
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in exception processing in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
1	1	1	1	COPROCESSOR ID	0	0	0	EFFECTIVE ADDRESS MODE	EFFECTIVE ADDRESS REGISTER
0	R/M	0	SOURCE SPECIFIER		DESTINATION REGISTER			OPMODE	

FMUL Floating-Point Multiply (MC6888X, MC68040) FMUL

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand location. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding will cause an unimplemented data type exception in the MC68040 to allow emulation in software.

FMUL Floating-Point Multiply (MC6888X, MC68040) FMUL

Destination Register field—Specifies the destination floating- point data register.

Opmode field—Specifies the instruction and rounding precision.

0100011	FMUL	Rounding precision specified by the floating-point control register.
1100011	FSMUL	Single-precision rounding specified.
1100111	FDMUL	Double-precision rounding specified.

FNEG

Floating-Point Negate
(MC6888X, MC68040)

FNEG

Operation: – (Source) → FPn

Assembler Syntax:
FNEG. < fmt > < ea > ,FPn
FNEG.X FPm,FPn
FNEG.X FPn
*FrNEG. < fmt > < ea > ,FPn
*FrNEG.X FPm,FPn
*FrNEG.X FPn
where r is rounding precision, S or D
*Supported by MC68040 only

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and inverts the sign of the mantissa. Stores the result in the destination floating-point data register.

FNEG will round the result to the precision selected in the floating-point control register. FSNEG and FDNEG will round the result to single or double precision, respectively, regardless of the rounding precision selected in the floating-point control register.

Operation Table:

DESTINATION	SOURCE					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Negate			– 0.0		+ 0.0	– inf		+ inf

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

FNEG

Floating-Point Negate
(MC6888X, MC68040)

FNEG

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Cleared
	OVFL	Cleared
	UNFL	If source is an extended-precision denormalized number, refer to exception processing in the appropriate user's manual; cleared otherwise.
	DZ	Cleared
	INEX2	Cleared
	INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in exception processing in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
1	1	1	1	COPROCESSOR ID	0	0	0	EFFECTIVE ADDRESS MODE	EFFECTIVE ADDRESS REGISTER
0	R/M	0	SOURCE SPECIFIER		DESTINATION REGISTER			OPMODE	

FNEG Floating-Point Negate (MC6888X, MC68040) FNEG

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding will cause an unimplemented data type exception to allow emulation in software.

FNEG Floating-Point Negate (MC6888X, MC68040) FNEG

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

Opmode field—Specifies the instruction and rounding precision.

0011010	FNEG	Rounding precision specified by the floating-point control register.
1011010	FSNEG	Single-precision rounding specified.
1011110	FDNEG	Double-precision rounding specified.

FNOP

No Operation
(MC6888X, MC68040)

FNOP

Operation: None

Assembler Syntax: FNOP

Attributes: Unsized

Description: This instruction does not perform any explicit operation. However, it is useful to force synchronization of the floating- point unit with an integer unit or to force processing of pending exceptions. For most floating-point instructions, the integer unit is allowed to continue with the execution of the next instruction once the floating-point unit has any operands needed for an operation, thus supporting concurrent execution of floating-point and integer instructions. The FNOP instruction synchronizes the floating-point unit and the integer unit by causing the integer unit to wait until all previous floating-point instructions have completed. Execution of FNOP also forces any exceptions pending from the execution of a previous floating-point instruction to be processed as a preinstruction exception.

The MC68882 may not wait to begin execution of another floating- point instruction until it has completed execution of the current instruction. The FNOP instruction synchronizes the coprocessor and microprocessor unit by causing the microprocessor unit to wait until the current instruction (or both instructions) have completed.

The FNOP instruction also forces the processing of exceptions pending from the execution of previous instructions. This is also inherent in the way that the floating-point coprocessor utilizes the M68000 family coprocessor interface. Once the floating-point coprocessor has received the input operand for an arithmetic instruction, it always releases the main processor to execute the next instruction (regardless of whether or not concurrent execution is prevented for the instruction due to tracing) without reporting the exception during the execution of that instruction. Then, when the main processor attempts to initiate the execution of the next floating-point coprocessor instruction, a preinstruction exception may be reported to initiate exception processing for an exception that occurred during a previous instruction. By using the FNOP instruction, the user can force any pending exceptions to be processed without performing any other operations.

Floating-Point Status Register: Not Affected.

FNOP

No Operation
(MC6888X, MC68040)

FNOP

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

NOTE

FNOP uses the same opcode as the FBcc.W < label > instruction, with cc = F (nontrapping false) and < label > = + 2 (which results in a displacement of 0).

FREM — IEEE Remainder (MC6888X, M68040FPSP) — FREM

Operation: IEEE Remainder of (FPn ÷ Source) → FPn

Assembler Syntax: FREM. < fmt > < ea > ,FPn
FREM.X FPm,FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the modulo remainder of the number in the destination floating-point data register, using the source operand as the modulus. Stores the result in the destination floating-point data register and stores the sign and seven least significant bits of the quotient in the floating-point status register quotient byte (the quotient is the result of FPn ÷ Source). The IEEE remainder function is defined as:

$$FPn - (Source \times N)$$

where N = INT (FPn ÷ Source) in the round-to-nearest mode.

The FREM function is not defined for a source operand equal to zero or for a destination operand equal to infinity. Note that this function is not the same as the FMOD instruction, which uses the round-to-zero mode and thus returns a remainder that is different from the remainder required by the IEEE *Specification for Binary Floating-Point Arithmetic*.

Operation Table:

DESTINATION		SOURCE[1]		
		+ In Range –	+ Zero# –	+ Infinity –
In Range	+ –	IEEE Remainder	NAN[2]	FPn[2]
Zero	+ –	+ 0.0 – 0.0	NAN[2]	+ 0.0 – 0.0
Infinity	+ –	NAN[2]	NAN[2]	NAN†[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.
3. Returns the value of FPn before the operation. However, the result is processed by the normal instruction termination procedure to round it as required. Thus, an overflow and/or inexact result may occur if the rounding precision has been changed to a smaller size since the FPn value was loaded.

FREM — IEEE Remainder (MC6888X, M68040FPSP) — FREM

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Loaded with the sign and least significant seven bits of the qotient (FPn ÷ Source). The sign of the quotient is the exclusive-OR of the sign bits of the source and destination operands.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is zero or the destination is infinity; cleared otherwise.
OVFL	Cleared
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FREM IEEE Remainder (MC6888X, M68040FPSP) FREM

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register.

FSCALE

Scale Exponent
(MC6888X, M68040FPSP)

FSCALE

Operation: FPn x INT(2Source) → FPn

Assembler Syntax: FSCALE. < fmt > < ea > ,FPn
FSCALE.X FPm,FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to an integer (if necessary) and adds that integer to the destination exponent. Stores the result in the destination floating-point data register. This function has the effect of multiplying the destination by 2^{Source}, but is much faster than a multiply operation when the source is an integer value.

The floating-point coprocessor assumes that the scale factor is an integer value before the operation is executed. If not, the value is chopped (i.e., rounded using the round-to-zero mode) to an integer before it is added to the exponent. When the absolute value of the source operand is $\geq 2^{14}$, an overflow or underflow always results.

Operation Table:

DESTINATION	SOURCE[1]		
	+ In Range –	+ Zero –	+ Infinity –
In Range + –	Scale Exponent	FPn[2]	NAN[3]
Zero + –	+ 0.0 – 0.0	+ 0.0 – 0.0	NAN[3]
Infinity + –	+ inf – inf	+ inf – inf	NAN[3]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Returns the value of FPn before the operation. However, the result is processed by the normal instruction termination procedure to round it as required. Thus, an overflow and/or inexact result may occur if the rounding precision has been changed to a smaller size since the FPn value was loaded.
3. Sets the OPERR bit in the floating-point status register exception byte.

FSCALE Scale Exponent (MC6888X, M68040FPSP) FSCALE

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source operand is ± infinity; cleared otherwise.
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	1	0	0	1	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FSCALE Scale Exponent (MC6888X, M68040FPSP) FSCALE

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register.

FScc Set According to Floating-Point Condition (MC6888X, MC68040) FScc

Operation:
If (Condition True)
Then 1s → Destination
Else 0s → Destination

Assembler Syntax:
FScc. < size > < ea >

Attributes:
Size = (Byte)

Description: If the specified floating-point condition is true, sets the byte integer operand at the destination to TRUE (all ones); otherwise, sets the byte to FALSE (all zeros). The conditional specifier cc may select any one of the 32 floating-point conditional tests as described in **Table 3-23 Floating-Point Conditional Tests**.

Floating-Point Status Register:

Condition Codes:	Not affected.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Set if the NAN condition code is set and the condition selected is an IEEE nonaware test.
	SNAN	Not Affected.
	OPERR	Not Affected.
	OVFL	Not Affected.
	UNFL	Not Affected.
	DZ	Not Affected.
	INEX2	Not Affected.
	INEX1	Not Affected.
Accrued Exception Byte:	The IOP bit is set if the BSUN bit is set in the exception byte. No other bit is affected.	

FScc Set According to Floating-Point Condition (MC6888X, MC68040) FScc

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	0	0	0	CONDITIONAL PREDICATE					

Instruction Fields:

Effective Address field—Specifies the addressing mode for the byte integer operand. Only data alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn,od)	—	—

Conditional Predicate field—Specifies one of 32 conditional tests as defined in **3.6.2 Conditional Testing**.

NOTE

When a BSUN exception occurs, a preinstruction exception is taken. If the exception handler returns without modifying the image of the program counter on the stack frame (to point to the instruction following the FScc), then it must clear the cause of the exception (by clearing the NAN bit or disabling the BSUN trap) or the exception occurs again immediately upon return to the routine that caused the exception.

FSGLDIV Single-Precision Divide (MC6888X, MC68040) FSGLDIV

Operation: FPn ÷ Source → FPn

Assembler Syntax: FSGLDIV. < fmt > < ea > ,FPn
FSGLDIV.X FPm,FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and divides that number into the number in the destination floating-point data register. Stores the result in the destination floating-point data register, rounded to single precision (despite the current rounding precision). This function is undefined for 0 ÷ 0 and infinity ÷ infinity.

Both the source and destination operands are assumed to be representable in the single-precision format. If either operand requires more than 24 bits of mantissa to be accurately represented, the extraneous mantissa bits are trancated prior to the division, hence the accuracy of the result is not guaranteed. Furthermore, the result exponent may exceed the range of single precision, regardless of the rounding precision selected in the floating-point control register mode control byte. Refer to **3.6.1 Underflow, Round, Overflow** for more information.

The accuracy of the result is not affected by the number of mantissa bits required to represent each input operand since the input operands just change to extended precision. The result mantissa is rounded to single precision, and the result exponent is rounded to extended precision, despite the rounding precision selected in the floating-point control register.

Operation Table:

DESTINATION		SOURCE[3,1]					
		+	In Range	–	+	Zero	–	+	Infinity	–
In Range	+	Divide (Single Precision)			+ inf[2]		– inf[2]	+ 0.0		– 0.0
	–				– inf[2]		+ inf[2]	– 0.0		+ 0.0
Zero	+	+ 0.0		– 0.0	NAN[3]			+ 0.0		– 0.0
	–	– 0.0		+ 0.0				– 0.0		+ 0.0
Infinity	+	+ inf		– inf	+ inf		– inf	NAN[3]		
	–	– inf		+ inf	– inf		+ inf			

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the DZ bit in the floating-point status register exception byte.
3. Sets the OPERR bit in the floating-point status register exception byte.

FSGLDIV Single-Precision Divide (MC6888X, MC68040) FSGLDIV

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set for 0 ÷ 0 or infinity ÷ infinity.
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Set if the source is zero and the destination is in range; cleared otherwise.
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to the appropriate user's manual for inexact result on decimal input; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	0	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FSGLDIV Single-Precision Divide (MC6888X, MC68040) FSGLDIV

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register.

FSGLMUL Single-Precision Multiply (MC6888X, MC68040) FSGLMUL

Operation: Source x FPn → FPn

Assembler Syntax: FSGLMUL. < fmt > < ea > ,FPn
FSGLMUL.X FPm,FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and multiplies that number by the number in the destination floating-point data register. Stores the result in the destination floating-point data register, rounded to single precision (regardless of the current rounding precision).

Both the source and destination operands are assumed to be representable in the single-precision format. If either operand requires more than 24 bits of mantissa to be accurately represented, the extraneous mantissa bits are truncated prior to the multiplication; hence, the accuracy of the result is not guaranteed. Furthermore, the result exponent may exceed the range of single precision, regardless of the rounding precision selected in the floating-point control register mode control byte. Refer to **3.6.1 Underflow, Round, Overflow** for more information.

Operation Table:

DESTINATION		SOURCE[1]					
		+ In Range	–	+ Zero	–	+ Infinity	–
In Range	+	Multiply (Single Precision)		+ 0.0	– 0.0	+ inf	– inf
	–			– 0.0	+ 0.0	– inf	+ inf
Zero	+	+ 0.0	– 0.0	+ 0.0	– 0.0	NAN[2]	
	–	– 0.0	+ 0.0	– 0.0	+ 0.0		
Infinity	+	+ inf	– inf	NAN		+ inf	– inf
	–	– inf	+ inf			– inf	+ inf

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

NOTE

The input operand mantissas truncate to single precision before the multiply operation. The result mantissa rounds to single precision despite the rounding precision selected in the floating-point control register.

FSGLMUL Single-Precision Multiply (MC6888X, MC68040) FSGLMUL

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing.**

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if one operand is zero and the other is infinity; cleared otherwise.
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	1	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FSGLMUL Single-Precision Multiply (MC6888X, MC68040) FSGLMUL

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register.

FSIN Sine (MC6888X, M68040FPSP) FSIN

Operation: Sine of Source → FPn

Assembler Syntax:
FSIN. < fmt > < ea > ,FPn
FSIN.X FPm,FPn
FSIN.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the sine of that number. Stores the result in the destination floating-point data register. This function is not defined for source operands of ± infinity. If the source operand is not in the range of [– 2π... + 2π], the argument is reduced to within that range before the sine is calculated. However, large arguments may lose accuracy during reduction, and very large arguments (greater than approximately 10^{20}) lose all accuracy. The result is in the range of [– 1... + 1].

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Sine			+ 0.0		– 0.0	NAN[2]		

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

FSIN

Sine
(MC6888X, M68040FPSP)

FSIN

Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the source is ± infinity; cleared otherwise.
	OVFL	Cleared
	UNFL	Refer to underflow in the appropriate user's manual.
	DZ	Cleared
	INEX2	Refer to inexact result in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FSIN Sine (MC6888X, M68040FPSP) FSIN

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, then the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FSINCOS Simultaneous Sine and Cosine (MC6888X, M68040FPSP) FSINCOS

Operation: Sine of Source → FPs
Cosine of Source → FPc

Assembler Syntax: FSINCOS. < fmt > < ea > ,FPc,FPs
FSINCOS.X FPm,FPc,FPs

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates both the sine and the cosine of that number. Calculates both functions simultaneously; thus, this instruction is significantly faster than performing separate FSIN and FCOS instructions. Loads the sine and cosine results into the destination floating-point data register. Sets the condition code bits according to the sine result. If FPs and FPc are specified to be the same register, the cosine result is first loaded into the register and then is overwritten with the sine result. This function is not defined for source operands of ± infinity.

If the source operand is not in the range of [– 2π... + 2π], the argument is reduced to within that range before the sine and cosine are calculated. However, large arguments may lose accuracy during reduction, and very large arguments (greater than approximately 10^{20}) lose all accuracy. The results are in the range of [– 1... + 1].

Operation Table:

DESTINATION	SOURCE[1]		
	+ In Range –	+ Zero –	+ Infinity –
FPs	Sine	+ 0.0 – 0.0	NAN[2]
FPc	Cosine	+ 1.0	NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

FSINCOS Simultaneous Sine and Cosine (MC6888X, M68040FPSP) FSINCOS

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing** (for the sine result).

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is ± infinity; cleared otherwise.
OVFL	Cleared
UNFL	Set if a sine underflow occurs, in which case the cosine result is 1. Cosine cannot underflow. Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER, FPs			0	1	1	0	DESTINATION REGISTER FPc	

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FSINCOS Simultaneous Sine and Cosine (MC6888X, M68040FPSP) FSINCOS

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register, FPc field—Specifies the destination floating- point data register, FPc. The cosine result is stored in this register.

FSINCOS Simultaneous Sine and Cosine (MC6888X, M68040FPSP) FSINCOS

Destination Register, FPs field—Specifies the destination floating- point data register, FPs. The sine result is stored in this register. If FPc and FPs specify the same floating-point data register, the sine result is stored in the register, and the cosine result is discarded.

If R/M = 0 and the source register field is equal to either of the destination register fields, the input operand is taken from the specified floating-point data register, and the appropriate result is written into the same register.

FSINH Hyperbolic Sine (MC6888X, M68040FPSP) FSINH

Operation: Hyperbolic Sine of Source → FPn

Assembler Syntax:
FSINH. < fmt > < ea > ,FPn
FSINH.X FPm,FPn
FSINH.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the hyperbolic sine of that number. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Hyperbolic Sine	+ 0.0 – 0.0	+ inf – inf

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FSINH Hyperbolic Sine (MC6888X, M68040FPSP) FSINH

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	0	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FSINH Hyperbolic Sine (MC6888X, M68040FPSP) FSINH

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FSQRT Floating-Point Square Root (MC6888X, MC68040) FSQRT

Operation: Square Root of Source → FPn

Assembler Syntax:
FSQRT. < fmt > < ea > ,FPn
FSQRT.X FPm,FPn
FSQRT.X FPn
*FrSQRT. < fmt > < ea > ,FPn
*FrSQRT FPm,FPn
*FrSQRT FPn
where r is rounding precision, S or D

*Supported by MC68040 only

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the square root of that number. Stores the result in the destination floating-point data register. This function is not defined for negative operands.

FSQRT will round the result to the precision selected in the floating-point control register. FSFSQRT and FDFSQRT will round the result to single or double precision, respectively, regardless of the rounding precision selected in the floating-point control register.Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	$\sqrt{x}$		NAN[2]	+ 0.0		– 0.0	+ inf		NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

FSQRT Floating-Point Square Root (MC6888X, MC68040) FSQRT

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if the source operand is not zero and is negative; cleared otherwise.
	OVFL	Cleared
	UNFL	Cleared
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in exception processing in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			OPMODE					

FSQRT Floating-Point Square Root (MC6888X, MC68040) FSQRT

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register		Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn		(xxx).W	111	000
An	—	—		(xxx).L	111	001
(An)	010	reg. number:An		# < data >	111	100
(An) +	011	reg. number:An				
– (An)	100	reg. number:An				
(d_{16},An)	101	reg. number:An		(d_{16},PC)	111	010
(d_8,An,Xn)	110	reg. number:An		(d_8,PC,Xn)	111	011
(bd,An,Xn)	110	reg. number:An		(bd,PC,Xn)	111	011
([bd,An,Xn],od)	110	reg. number:An		([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An		([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding will cause an unimplemented data type exception in the MC68040 to allow emulation in software.

FSQRT Floating-Point Square Root (MC6888X, MC68040) FSQRT

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

Opmode field—Specifies the instruction and rounding precision.

0000100	FSQRT	Rounding precision specified by the floating-point control register.
1000001	FSSQRT	Single-precision rounding specified.
1000101	FDSQRT	Double-precision rounding specified.

FSUB

Floating-Point Subtract
(MC6888X, MC68040)

FSUB

Operation: FPn – Source → FPn

Assembler Syntax:
FSUB. < fmt > < ea > ,FPn
FSUB.X FPm,FPn
*FrSUB. < fmt > < ea > ,FPn
*FrSUB.X FPm,FPn
where r is rounding precision, S or D

*Supported by MC68040 only

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and subtracts that number from the number in the destination floating-point data register. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION		SOURCE[1] + In Range	SOURCE[1] – In Range	SOURCE[1] + Zero	SOURCE[1] – Zero	SOURCE[1] + Infinity	SOURCE[1] – Infinity
In Range	+	Subtract		Subtract		– inf	+ inf
	–						
Zero	+	Subtract		+ 0.0[2]	+ 0.0	– inf	+ inf
	–			+ 0.0	+ 0.0[2]		
Infinity	+	+ inf		+ inf		NAN[2]	– inf
	–	– inf		– inf		– inf	NAN[2]

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Returns + 0.0 in rounding modes RN, RZ, and RP; returns – 0.0 in RM.
3. Sets the OPERR bit in the floating-point status register exception byte.

FSUB

Floating-Point Subtract
(MC6888X, MC68040)

FSUB

Floating-Point Status Register:

Condition Codes:	Affected as described in **3.6.2 Conditional Testing**.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Cleared
	SNAN	Refer to **1.6.5 Not-A-Numbers**.
	OPERR	Set if both the source and destination are like-signed infinities; cleared otherwise.
	OVFL	Refer to exception processing in the appropriate user's manual.
	UNFL	Refer to exception processing in the appropriate user's manual.
	DZ	Cleared
	INEX2	Refer to exception processing in the appropriate user's manual.
	INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.
Accrued Exception Byte:	Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.	

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			OPMODE					

FSUB Floating-Point Subtract (MC6888X, MC68040) FSUB

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register		Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn		(xxx).W	111	000
An	—	—		(xxx).L	111	001
(An)	010	reg. number:An		# < data >	111	100
(An) +	011	reg. number:An				
– (An)	100	reg. number:An				
(d_{16},An)	101	reg. number:An		(d_{16},PC)	111	010
(d_8,An,Xn)	110	reg. number:An		(d_8,PC,Xn)	111	011
(bd,An,Xn)	110	reg. number:An		(bd,PC,Xn)	111	011
([bd,An,Xn],od)	110	reg. number:An		([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An		([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding will cause an unimplemented data type exception in the MC68040 to allow emulation in software.

FSUB Floating-Point Subtract (MC6888X, MC68040) FSUB

Destination Register field—Specifies the destination floating- point data register.

Opmode field—Specifies the instruction and rounding precision.

0101000	FSUB	Rounding precision specified by the floating- point control register.
1101000	FSSUB	Single-precision rounding specified.
1101100	FDSUB	Double-precision rounding specified.

FTAN — Tangent (MC6888X/004SW) — FTAN

Operation: Tangent of Source → FPn

Assembler Syntax:
FTAN. < fmt > < ea > ,FPn
FTAN.X FPm,FPn
FTAN.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the tangent of that number. Stores the result in the destination floating-point data register. This function is not defined for source operands of ± infinity. If the source operand is not in the range of [– π/2... + π/2], the argument is reduced to within that range before the tangent is calculated. However, large arguments may lose accuracy during reduction, and very large arguments (greater than approximately 10^{20}) lose all accuracy.

Operation Table:

DESTINATION	SOURCE[1]					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	Tangent			+ 0.0		– 0.0	NAN[2]		

NOTES:
1. If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.
2. Sets the OPERR bit in the floating-point status register exception byte.

FTAN

Tangent
(MC6888X/004SW)

FTAN

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Set if the source is ± infinity; cleared otherwise.
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	1	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

FTAN Tangent (MC6888X/004SW) FTAN

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FTANH Hyperbolic Tangent (MC6888X, M68040FPSP) FTANH

Operation: Hyperbolic Tangent of Source → FPn

Assembler Syntax:
FTANH. < fmt > < ea > ,FPn
FTANH.X FPm,FPn
FTANH.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates the hyperbolic tangent of that number. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE		
	+ In Range –	+ Zero –	+ Infinity –
Result	Hyperbolic Tangent	+ 0.0 – 0.0	+ 1.0 – 1.0

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Cleared
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FTANH Hyperbolic Tangent (MC6888X, M68040FPSP) FTANH

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	0	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

FTANH Hyperbolic Tangent (MC6888X, M68040FPSP) FTANH

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FTENTOX 10^X FTENTOX

(MC6888X, M68040FPSP)

Operation: $10^{Source} \rightarrow FPn$

Assembler Syntax:
FTENTOX. < fmt > < ea > ,FPn
FTENTOX.X FPm,FPn
FTENTOX.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates 10 to the power of that number. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE					
	+	In Range	–	+ Zero	–	+ Infinity –
Result	10^X			+ 1.0		+ inf + 0.0

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to the appropriate user's manual inexact result on decimal input; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FTENTOX 10^X FTENTOX

(MC6888X, M68040FPSP)

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	0

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FTENTOX 10^X FTENTOX

(MC6888X, M68040FPSP)

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

FTRAPcc Trap on Floating-Point Condition (MC6888X, MC68040) FTRAPcc

Operation: If Condition True
Then TRAP

Assembler Syntax: FTRAPcc
FTRAPcc.W # < data >
FTRAPcc.L # < data >

Attributes: Size = (Word, Long)

Description: If the selected condition is true, the processor initiates exception processing. A vector number is generated to reference the TRAPcc exception vector. The stacked program counter points to the next instruction. If the selected condition is not true, there is no operation performed and execution continues with the next instruction in sequence. The immediate data operand is placed in the word(s) following the conditional predicate word and is available for user definition for use within the trap handler.

The conditional specifier cc selects one of the 32 conditional tests defined in **3.6.2 Conditional Testing**.

Floating-Point Status Register:

Condition Codes:	Not affected.	
Quotient Byte:	Not affected.	
Exception Byte:	BSUN	Set if the NAN condition code is set and the condition selected is an IEEE nonaware test.
	SNAN	Not Affected.
	OPERR	Not Affected.
	OVFL	Not Affected.
	UNFL	Not Affected.
	DZ	Not Affected.
	INEX2	Not Affected.
	INEX1	Not Affected.
Accrued Exception Byte:	The IOP bit is set if the BSUN bit is set in the exception byte; no other bit is affected.	

FTRAPcc Trap on Floating-Point Condition FTRAPcc
(MC6888X, MC68040)

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	1	1	1	MODE		
0	0	0	0	0	0	0	0	0	0	CONDITIONAL PREDICATE					
16-BIT OPERAND OR MOST SIGNIFICANT WORD OF 32-BIT OPERAND (IFNEEDED)															
LEAST SIGNIFICANT WORD OR 32-BIT OPERAND (IF NEEDED)															

Instruction Fields:

Mode field—Specifies the form of the instruction.
- 010 — The instruction is followed by a word operand.
- 011 — The instruction is followed by a long-word operand.
- 100 — The instruction has no operand.

Conditional Predicate field—Specifies one of 32 conditional tests as described in **3.6.2 Conditional Testing**.

Operand field—Contains an optional word or long-word operand that is user defined.

NOTE

When a BSUN exception occurs, a preinstruction exception is taken by the main processor. If the exception handler returns without modifying the image of the program counter on the stack frame (to point to the instruction following the FTRAPcc), it must clear the cause of the exception (by clearing the NAN bit or disabling the BSUN trap), or the exception occurs again immediately upon return to the routine that caused the exception.

FTST Test Floating-Point Operand (MC6888X, MC68040) FTST

Operation: Condition Codes for Operand → FPCC

Assembler Syntax: FTST. < fmt > < ea >
FTST.X FPm

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and sets the condition code bits according to the data type of the result.

Operation Table: The contents of this table differfromtheother operation tables. A letter in an entry of this table indicates that the designated condition code bit is always set by the FTST operation. All unspecified condition code bits are cleared during the operation.

DESTINATION	SOURCE					
	+	In Range	–	+	Zero	–	+	Infinity	–
Result	none		N	Z		NZ	I		NI

NOTE: If the source operand is a NAN, set the NAN condition code bit. If the source operand is an SNAN, set the SNAN bit in the floating-point status register exception byte

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Cleared
UNFL	Cleared
DZ	Cleared
INEX2	Cleared
INEX1	If < fmt > is packed, refer to exception processing in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in exception processing in the appropriate user's manual.

FTST Test Floating-Point Operand (MC6888X, MC68040) FTST

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	1	1	1	0	1	0

Instruction Fields:

Effective Address field—Determines the addressing mode for external operands.
If R/M = 0, this field is unused and should be all zeros.
If R/M = 1, specifies the location of the source operand. Only data addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FTST Test Floating-Point Operand (MC6888X, MC68040) FTST

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)*
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)
*This encoding will cause an unimplemented data type exception in the MC68040 to allow emulation in software.

Destination Register field—Since the floating-point unit uses a common command word format for all of the arithmetic instructions (including FTST), this field is treated in the same manner for FTST as for the other arithmetic instructions, even though the destination register is not modified. This field should be set to zero to maintain compatibility with future devices; however, the floating-point unit does not signal an illegal instruction trap if it is not zero.

FTWOTOX 2^X FTWOTOX

(MC6888X, M68040FPSP)

Operation: $2^{Source} \rightarrow FPn$

Assembler Syntax:
FTWOTOX. < fmt > < ea > ,FPn
FTWOTOX.X FPm,FPn
FTWOTOX.X FPn

Attributes: Format = (Byte, Word, Long, Single, Double, Extended, Packed)

Description: Converts the source operand to extended precision (if necessary) and calculates two to the power of that number. Stores the result in the destination floating-point data register.

Operation Table:

DESTINATION	SOURCE					
	+	In Range	–	+ Zero	–	+ Infinity –
Result	2^x			+ 1.0		+ inf + 0.0

NOTE: If the source operand is a NAN, refer to **1.6.5 Not-A-Numbers** for more information.

Floating-Point Status Register:

Condition Codes: Affected as described in **3.6.2 Conditional Testing**.

Quotient Byte: Not affected.

Exception Byte:

BSUN	Cleared
SNAN	Refer to **1.6.5 Not-A-Numbers**.
OPERR	Cleared
OVFL	Refer to overflow in the appropriate user's manual.
UNFL	Refer to underflow in the appropriate user's manual.
DZ	Cleared
INEX2	Refer to inexact result in the appropriate user's manual.
INEX1	If < fmt > is packed, refer to inexact result on decimal input in the appropriate user's manual; cleared otherwise.

Accrued Exception Byte: Affected as described in IEEE exception and trap compatibility in the appropriate user's manual.

FTWOTOX 2^X FTWOTOX

(MC6888X, M68040FPSP)

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	0	1

Instruction Fields:

Coprocessor ID field—Specifies which coprocessor in the system is to execute this instruction. Motorola assemblers default to ID = 1 for the floating-point coprocessor.

Effective Address field—Determines the addressing mode for external operands.

If R/M = 0, this field is unused and should be all zeros.

If R/M = 1, this field is encoded with an M68000 family addressing mode as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn],od)	110	reg. number:An
([bd,An],Xn,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn],od)	111	011
([bd,PC],Xn,od)	111	011

*Only if < fmt > is byte, word, long, or single.

FTWOTOX 2^X FTWOTOX

(MC6888X, M68040FPSP)

R/M field—Specifies the source operand address mode.
0 — The operation is register to register.
1 — The operation is < ea > to register.

Source Specifier field—Specifies the source register or data format.
If R/M = 0, specifies the source floating-point data register.
If R/M = 1, specifies the source data format:
000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

Destination Register field—Specifies the destination floating- point data register. If R/M = 0 and the source and destination fields are equal, the input operand is taken from the specified floating-point data register, and the result is written into the same register. If the single register syntax is used, Motorola assemblers set the source and destination fields to the same value.

SECTION 6
SUPERVISOR (PRIVILEGED) INSTRUCTIONS

This section contains information about the supervisor privileged instructions for the M68000 family. Each instruction is described in detail, and the instruction descriptions are arranged in alphabetical order by instruction mnemonic.

Any differences within the M68000 family of instructions are identified in the instruction. If an instruction only applies to a certain processor or processors, the processor(s) that the instruction pertains to is identified under the title of the instruction. For example:

Invalidate Cache Lines
(MC68040)

All references to the MC68000, MC68020, and MC68030 include references to the corresponding embedded controllers, MC68EC000, MC68EC020, and MC68EC030. All references to the MC68040 include the MC68LC040 and MC68EC040. This applies throughout this section unless otherwise specified.

If the instruction applies to all the M68000 family but a processor or processors may use a different instruction field, instruction format, etc., the differences will be identified within the paragraph. For example:

MC68020, MC68030 and MC68040 only

(bd,An,Xn)*	110	reg. number: An	(bd,PC,Xn)*	—	—

*Can be used with CPU32 processo

The following instructions are listed separately for each processor due to the many differences involved within the instruction:

PFLUSH	Flush ATC Entries
PMOVE	Move PMMU Register
PTEST	Test Logical Address

Appendix A Processor Instruction Summary provides a listing of all processors and the instructions that apply to them for quick reference.

ANDI to SR

AND Immediate to the Status Register (M68000 Family)

ANDI to SR

Operation: If Supervisor State
Then Source L SR → SR
ELSE TRAP

Assembler Syntax: ANDI # < data > ,SR

Attributes: size = (word)

Description: Performs an AND operation of the immediate operand with the contents of the status register and stores the result in the status register. All implemented bits of the status register are affected.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X—Cleared if bit 4 of immediate operand is zero; unchanged otherwise.

N—Cleared if bit 3 of immediate operand is zero; unchanged otherwise.

Z—Cleared if bit 2 of immediate operand is zero; unchanged otherwise.

V—Cleared if bit 1 of immediate operand is zero; unchanged otherwise.

C—Cleared if bit 0 of immediate operand is zero; unchanged otherwise.

Instruction Format:

5	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0
16-BIT WORD DATA															

CINV — Invalidate Cache Lines (MC68040, MC68LC040) — CINV

Operation:
If Supervisor State
Then Invalidate Selected Cache Lines
ELSE TRAP

Assembler Syntax:
CINVL < caches > ,(An)
CINVP < caches > ,(An)
CINVA < caches >

Where < caches > specifies the instruction cache, data cache, both caches, or neither cache.

Attributes: Unsized

Description: Invalidates selected cache lines. The data cache, instruction cache, both caches, or neither cache can be specified. Any dirty data in data cache lines that invalidate are lost; the CPUSH instruction must be used when dirty data may be contained in the data cache.

Specific cache lines can be selected in three ways:

1. CINVL invalidates the cache line (if any) matching the physical address in the specified address register.
2. CINVP invalidates the cache lines (if any) matching the physical memory page in the specified address register. For example, if 4K-byte page sizes are selected and An contains $12345000, all cache lines matching page $12345000 invalidate.
3. CINVA invalidates all cache entries.

Condition Codes:

Not affected.

CINV — Invalidate Cache Lines (MC68040, MC68LC040) — CINV

Instruction Format:

15	14	13	12	11	10	9	8	7 6	5	4 3	2 1 0
1	1	1	1	0	1	0	0	CACHE	0	SCOPE	REGISTER

Instruction Fields:

Cache field—Specifies the Cache.
- 00—No Operation
- 01—Data Cache
- 10—Instruction Cache
- 11—Data and Instruction Caches

Scope field—Specifies the Scope of the Operation.
- 00—Illegal (causes illegal instruction trap)
- 01—Line
- 10—Page
- 11—All

Register field—Specifies the address register for line and page operations. For line operations, the low-order bits 3–0 of the address are don‘t cares. Bits 11–0 or 12–0 of the address are don‘t care for 4K-byte or 8K-byte page operations, respectively.

cpRESTORE Coprocessor Restore Functions (MC68020, MC68030) cpRESTORE

Operation:
If Supervisor State
Then Restore Internal State of Coprocessor
ELSE TRAP

Assembler Syntax: cpRESTORE < ea >

Attributes: Unsized

Description: Restores the internal state of a coprocessor usually after it has been saved by a preceding cpSAVE instruction.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
1	1	1	1	COPROCESSOR ID	1	0	1	EFFECTIVE ADDRESS MODE	EFFECTIVE ADDRESS REGISTER

cpRESTORE Coprocessor Restore Functions (MC68020, MC68030) cpRESTORE

Instruction Fields:

Coprocessor ID field—Identifies the coprocessor that is to be restored. Coprocessor ID of 000 results in an F-line exception for the MC68030.

Effective Address field—Specifies the location where the internal state of the coprocessor is located. Only postincrement or control addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
— (An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn], od)	110	reg. number:An
([bd,An],Xn,od)	110	reg.number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn], od)	111	011
([bd,PC],Xn, od	111	011

NOTE

If the format word returned by the coprocessor indicates “come again”, pending interrupts are not serviced.

cpSAVE Coprocessor Save Function (MC68020, MC68030) cpSAVE

Operation: If Supervisor State
Then Save Internal State of Coprocessor
ELSE TRAP

Assembler Syntax: cpSAVE < ea >

Attributes: Unsized

Description: Saves the internal state of a coprocessor in a format that can be restored by a cpRESTORE instruction.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			1	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:

Coprocessor ID field—Identifies the coprocessor for this operation. Coprocessor ID of 000 results in an F-line exception for the MC68030.

Effective Address field—Specifies the location where the internal state of the coprocessor is to be saved. Only predecrement or control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
— (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn], od)	110	reg. number:An
([bd,An],Xn, od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn],od)	—	—
([bd,PC],Xn, od)	—	—

CPUSH Push and Invalidate Cache Lines (MC68040, MC68LC040) CPUSH

Operation:

If Supervisor State
  Then If Data Cache
    Then Push Selected Dirty Data Cache Lines
    Invalidate Selected Cache Lines
  ELSE TRAP

Assembler Syntax:

CPUSHL < caches > ,(An)
CPUSHP < caches > ,(An)
CPUSHA < caches >

Where < caches > specifies the instruction cache, data cache, both caches, or neither cache.

Attributes: Unsized

Description: Pushes and then invalidates selected cache lines. The DATA cache, instruction cache, both caches, or neither cache can be specified. When the data cache is specified, the selected data cache lines are first pushed to memory (if they contain dirty DATA) and then invalidated. Selected instruction cache lines are invalidated.

Specific cache lines can be selected in three ways:

1. CPUSHL pushes and invalidates the cache line (if any) matching the physical address in the specified address register.
2. CPUSHP pushes and invalidates the cache lines (if any) matching the physical memory page in the specified address register. For example, if 4K-byte page sizes are selected and An contains $12345000, all cache lines matching page $12345000 are selected.
3. CPUSHA pushes and invalidates all cache entries.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7 6	5	4 3	2 1 0
1	1	1	1	0	1	0	0	CACHE	1	SCOPE	REGISTER

CPUSH Push and Invalidate Cache Lines (MC68040, MC68LC040) CPUSH

Instruction Fields:

Cache field—Specifies the Cache.
- 00—No Operation
- 01—Data Cache
- 10—Instruction Cache
- 11—Data and Instruction Caches

Scope field—Specifies the Scope of the Operation.
- 00—Illegal (causes illegal instruction trap)
- 01—Line
- 10—Page
- 11—All

Register field—Specifies the address register for line and page operations. For line operations, the low-order bits 3–0 of the address are don‘t care. Bits 11–0 or 12–0 of the address are don‘t care for 4K-byte or 8K-byte page operations, respectively.

EORI to SR

Exclusive-OR Immediate to the Status Register (M68000 Family)

EORI to SR

Operation: If Supervisor State
Then Source ⊕ SR → SR
ELSE TRAP

Assembler Syntax: EORI # < data > ,SR

Attributes: Size = (Word)

Description: Performs an exclusive-OR operation on the contents of the status register using the immediate operand and stores the result in the status register. All implemented bits of the status register are affected.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X—Changed if bit 4 of immediate operand is one; unchanged otherwise.

N—Changed if bit 3 of immediate operand is one; unchanged otherwise.

Z—Changed if bit 2 of immediate operand is one; unchanged otherwise.

V—Changed if bit 1 of immediate operand is one; unchanged otherwise.

C—Changed if bit 0 of immediate operand is one; unchanged otherwise.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	0
16-BIT WORD DATA															

FRESTORE Restore Internal Floating-Point State (MC68881, MC68882, MC68040 only) FRESTORE

Operation:
If in Supervisor State
Then FPU State Frame → Internal State
ELSE TRAP

Assembler Syntax: FRESTORE < ea >

Attributes: Unsized

Description: Aborts the execution of any floating-point operation in progress and loads a new floating-point unit internal state from the state frame located at the effective address. The first word at the specified address is the format word of the state frame. It specifies the size of the frame and the revision number of the floating-point unit that created it. A format word is invalid if it does not recognize the size of the frame or the revision number does not match the revision of the floating-point unit. If the format word is invalid, FRESTORE aborts, and a format exception is generated. If the format word is valid, the appropriate state frame is loaded, starting at the specified location and proceeding through higher addresses.

The FRESTORE instruction does not normally affect the programmer's model registers of the floating-point coprocessor, except for the NULL state size, as described below. It is only for restoring the user invisible portion of the machine. The FRESTORE instruction is used with the FMOVEM instruction to perform a full context restoration of the floating-point unit, including the floating- point data registers and system control registers. To accomplish a complete restoration, the FMOVEM instructions are first executed to load the programmer's model, followed by the FRESTORE instruction to load the internal state and continue any previously suspended operation.

FRESTORE Restore Internal Floating-Point State (MC68881, MC68882, MC68040 only) FRESTORE

The current implementation supports the following four state frames:

NULL: This state frame is 4 bytes long, with a format word of $0000. An FRESTORE operation with this size state frame is equivalent to a hardware reset of the floating-point unit. The programmer's model is set to the reset state, with nonsignaling NANs in the floating-point data registers and zeros in the floating-point control register, floating-point status register, and floating-point instruction address register. (Thus, it is unnecessary to load the programmer's model before this operation.)

IDLE: This state frame is 4 bytes long in the MC68040, 28 ($1C) bytes long in the MC68881, and 60 ($3C) bytes long in the MC68882. An FRESTORE operation with this state frame causes the floating-point unit to be restored to the idle state, waiting for the initiation of the next instruction, with no exceptions pending. The programmer's model is not affected by loading this type of state frame.

UNIMP: This state frame is generated only by the MC68040. It is 48 ($30) bytes long. An FSAVE that generates this size frame indicates either an unimplemented floating-point instruction or only an E1 exception is pending. This frame is never generated when an unsupported data type exception is pending or an E3 exception is pending. If both E1 and E3 exceptions are pending, a BUSY frame is generated.

BUSY: This state frame is 96 ($60) bytes long in the MC68040, 184 ($B8) bytes long in the MC68881, and 216 ($D8) bytes long in the MC68882. An FRESTORE operation with this size state frame causes the floating-point unit to be restored to the busy state, executing the instructions that were suspended by a previous FSAVE operation. The programmer's model is not affected by loading this type of state frame; however, the completion of the suspended instructions after the restore is executed may modify the programmer's model.

Floating-Point Status Register: Cleared if the state size is NULL; otherwise, not affected.

FRESTORE Restore Internal Floating-Point State (MC68881, MC68882, MC68040 only) FRESTORE

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			1	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Field:

Effective Address field—Determines the addressing mode for the state frame. Only postincrement or control addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register		Addressing Mode	Mode	Register
Dn	—	—		(xxx).W	111	000
An	—	—		(xxx).L	111	001
(An)	010	reg. number:An		# < data >	—	—
(An) +	011	reg. number:An				
—(An)	—	—				
(d_{16},An)	101	reg. number:An		(d_{16},PC)	111	010
(d_8,An,Xn)	110	reg. number:An		(d_8,PC,Xn)	111	011
(bd,An,Xn)	110	reg. number:An		(bd,PC,Xn)	111	011
([bd,An,Xn], od)	110	reg. number:An		([bd,PC,Xn], od)	111	011
([bd,An],Xn, od)	110	reg. number:An		([bd,PC],Xn, od)	111	011

FSAVE Save Internal Floating-Point State (MC68881, MC68882, MC68040 only) FSAVE

Operation: If in Supervisor State
Then FPU Internal State → State Frame
ELSE TRAP

Assembler Syntax: FSAVE < ea >

Attributes: Unsized

Description: FSAVE allows the completion of any floating-point operation in progress for the MC68040. It saves the internal state of the floating-point unit in a state frame located at the effective address. After the save operation, the floating-point unit is in the idle state, waiting for the execution of the next instruction. The first word written to the state frame is the format word specifying the size of the frame and the revision number of the floating-point unit.

Any floating-point operations in progress when an FSAVE instruction is encountered can be completed before the FSAVE executes, saving an IDLE state frame. Execution of instructions already in the floating-point unit pipeline continues until completion of all instructions in the pipeline or generation of an exception by one of the instructions. An IDLE state frame is created by the FSAVE if no exceptions occurred; otherwise, a BUSY or an UNIMP stack frame is created.

FSAVE suspends the execution of any operation in progress and saves the internal state in a state frame located at the effective address for the MC68881/MC68882. After the save operation, the floating-point coprocessor is in the idle state, waiting for the execution of the next instruction. The first word written to the state frame is the format word, specifying the size of the frame and the revision number of the floating-point coprocessor. The microprocessor unit initiates the FSAVE instruction by reading the floating-point coprocessor save CIR. The floating-point coprocessor save CIR is encoded with a format word that indicates the appropriate action to be taken by the main processor. The current implementation of the floating-point coprocessor always returns one of five responses in the save CIR:

Value	Definition
$0018	Save NULL state frame
$0118	Not ready, come again
$0218	Illegal, take format exception
$XX18	Save IDLE state frame
$XXB4	Save BUSY state frame

NOTE: XX is the floating-point coprocessor version number.

FSAVE Save Internal Floating-Point State (MC68881, MC68882, MC68040 only) FSAVE

The not ready format word indicates that the floating-point coprocessor is not prepared to perform a state save and that the microprocessor unit should process interrupts, if necessary, and re-read the save CIR. The floating-point coprocessor uses this format word to cause the main processor to wait while an internal operation completes, if possible, to allow an IDLE frame rather than a BUSY frame to be saved. The illegal format word aborts an FSAVE instruction that is attempted while the floating-point coprocessor executes a previous FSAVE instruction. All other format words cause the microprocessor unit to save the indicated state frame at the specified address. For state frame details see state frames in the appropriate user's manual.

The following state frames apply to both the MC68040 and the MC68881/MC68882.

NULL: This state frame is 4 bytes long. An FSAVE instruction that generates this state frame indicates that the floating-point unit state has not been modified since the last hardware reset or FRESTORE instruction with a NULL state frame. This indicates that the programmer's model is in the reset state, with nonsignaling NANs in the floating-point data registers and zeros in the floating- point control register, floating-point status register, and floating-point instruction address register. (Thus, it is not necessary to save the programmer's model.)

IDLE: This state frame is 4 bytes long in the MC68040, 28 ($1C) bytes long in the MC68881, and 60 ($3C) bytes long in the MC68882. An FSAVE instruction that generates this state frame indicates that the floating-point unit finished in an idle condition and is without any pending exceptions waiting for the initiation of the next instruction.

UNIMP: This state frame is generated only by the MC68040. It is 48 ($30) bytes long. An FSAVE that generates this size frame indicates either an unimplemented floating-point instruction or that only an E1 exception is pending. This frame is never generated when an unsupported data type exception or an E3 exception is pending. If both E1 and E3 exceptions are pending, a BUSY frame is generated.

BUSY: This state frame is 96 ($60) bytes long in the MC68040, 184 ($B8) bytes long in the MC68881, and 216 ($D8) bytes long in the MC68882. An FSAVE instruction that generates this size state frame indicates that the floating-point unit encountered an exception while attempting to complete the execution of the previous floating-point instructions.

FSAVE Save Internal Floating-Point State (MC68881, MC68882, MC68040 only) FSAVE

The FSAVE does not save the programmer's model registers of the floating-point unit; it saves only the user invisible portion of the machine. The FSAVE instruction may be used with the FMOVEM instruction to perform a full context save of the floating-point unit that includes the floating-point data registers and system control registers. To accomplish a complete context save, first execute an FSAVE instruction to suspend the current operation and save the internal state, then execute the appropriate FMOVEM instructions to store the programmer's model.

Floating-Point Status Register: Not affected.

Instruction Format:

15	14	13	12	11 10 9	8	7	6	5 4 3	2 1 0
1	1	1	1	COPROCESSOR ID	1	0	0	EFFECTIVE ADDRESS MODE	REGISTER

Instruction Field:

Effective Address field—Determines the addressing mode for the state frame. Only predecrement or control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

MOVE from SR

Move from the Status Register (MC68EC000, MC68010, MC68020, MC68030, MC68040, CPU32)

MOVE from SR

Operation: If Supervisor State
Then SR → Destination
Else TRAP

Assembler Syntax: MOVE SR, < ea >

Attributes: Size = (Word)

Description: Moves the data in the status register to the destination location. The destination is word length. Unimplemented bits are read as zeros.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

MOVE from SR

Move from the Status Register (MC68EC000, MC68010, MC68020, MC68030, MC68040, CPU32)

MOVE from SR

Instruction Field:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—

MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

*Available for the CPU32.

NOTE

Use the MOVE from CCR instruction to access only the condition codes.

MOVE to SR

Move to the Status Register (M68000 Family)

MOVE to SR

Operation: If Supervisor State
Then Source → SR
Else TRAP

Assembler Syntax: MOVE < ea > ,SR

Attributes: Size = (Word)

Description: Moves the data in the source operand to the status register. The source operand is a word, and all implemented bits of the status register are affected.

Condition Codes:

Set according to the source operand.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

MOVE to SR

Move to the Status Register (M68000 Family)

MOVE to SR

Instruction Field:

Effective Address field—Specifies the location of the source operand. Only data addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011

MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

(bd,PC,Xn)*	111	011
([bd,PC,Xn] ,od)	111	011
([bd,PC],Xn ,od)	111	011

*Available for the CPU32.

MOVE USP

Move User Stack Pointer
(M68000 Family)

MOVE USP

Operation: If Supervisor State
Then USP → An or An → USP
Else TRAP

Assembler Syntax: MOVE USP,An
MOVE An,USP

Attributes: Size = (Long)

Description: Moves the contents of the user stack pointer to or from the specified address register.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	dr	REGISTER		

Instruction Fields:

dr field—Specifies the direction of transfer.
0—Transfer the address register to the user stack pointer.
1—Transfer the user stack pointer to the address register.

Register field—Specifies the address register for the operation.

MOVEC — Move Control Register — MOVEC

(MC68010, MC68020, MC68030, MC68040, CPU32)

Operation: If Supervisor State
Then Rc → Rn or Rn → Rc
Else TRAP

Assembler Syntax: MOVEC Rc,Rn
MOVEC Rn,Rc

Attributes: Size = (Long)

Description: Moves the contents of the specified control register (Rc) to the specified general register (Rn) or copies the contents of the specified general register to the specified control register. This is always a 32-bit transfer, even though the control register may be implemented with fewer bits. Unimplemented bits are read as zeros.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	dr
A/D	REGISTER			CONTROL REGISTER											

Instruction Fields:

dr field—Specifies the direction of the transfer.
0—Control register to general register.
1—General register to control register.

A/D field—Specifies the type of general register.
0—Data Register
1—Address Rregister

MOVEC Move Control Register (MC68010, MC68020, MC68030, MC68040, CPU32) MOVEC

Register field—Specifies the register number.

Control Register field—Specifies the control register.

Hex[1]	Control Register
MC68010/MC68020/MC68030/MC68040/CPU32	
000	Source Function Code (SFC)
001	Destination Function Code (DFC)
800	User Stack Pointer (USP)
801	Vector Base Register (VBR)
MC68020/MC68030/MC68040	
002	Cache Control Register (CACR)
802	Cache Address Register (CAAR)[2]
803	Master Stack Pointer (MSP)
804	Interrupt Stack Pointer (ISP)
MC68040/MC68LC040	
003	MMU Translation Control Register (TC)
004	Instruction Transparent Translation Register 0 (ITT0)
005	Instruction Transparent Translation Register 1 (ITT1)
006	Data Transparent Translation Register 0 (DTT0)
007	Data Transparent Translation Register 1 (DTT1)
805	MMU Status Register (MMUSR)
806	User Root Pointer (URP)
807	Supervisor Root Pointer (SRP)
MC68EC040 only	
004	Instruction Access Control Register 0 (IACR0)
005	Instruction Access Control Register 1 (IACR1)
006	Data Access Control Register 0 (DACR1)
007	Data Access Control Register 1 (DACR1)

NOTES:
1. Any other code causes an illegal instruction exception
2. For the MC68020 and MC68030 only.

MOVES Move Address Space MOVES
(MC68010, MC68020, MC68030, MC68040, CPU32)

Operation: If Supervisor State
Then Rn → Destination [DFC] or Source [SFC] → Rn
Else TRAP

Assembler Syntax: MOVES Rn, < ea >
MOVES < ea > ,Rn

Attributes: Size = (Byte, Word, Long)

Description: This instruction moves the byte, word, or long operand from the specified general register to a location within the address space specified by the destination function code (DFC) register, or it moves the byte, word, or long operand from a location within the address space specified by the source function code (SFC) register to the specified general register. If the destination is a data register, the source operand replaces the corresponding low-order bits of that data register, depending on the size of the operation. If the destination is an address register, the source operand is sign-extended to 32 bits and then loaded into that address register.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		
A/D	REGISTER			dr	0	0	0	0	0	0	0	0	0	0	0

MOVES Move Address Space (MC68010, MC68020, MC68030, MC68040, CPU32) MOVES

Instruction Fields:

Size field—Specifies the size of the operation.
00—Byte Operation
01—Word Operation
10—Long Operation

Effective Address field—Specifies the source or destination location within the alternate address space. Only memory alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—

MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

(bd,PC,Xn)*	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

*Available for the CPU32.

A/D field—Specifies the type of general register.
0—Data Register
1—Address Register

Register field—Specifies the register number.

dr field—Specifies the direction of the transfer.
0—From < ea > to general register.
1—From general register to < ea > .

MOVES Move Address Space MOVES
(MC68010, MC68020, MC68030, MC68040, CPU32)

NOTE

The value stored is undefined for either of the two following examples with the same address register as both source and destination.

```
MOVES.x An,(An) +
MOVES.x An,D(An)
```

The current implementations of the MC68010, MC68020, MC68030, and MC68040 store the incremented or decremented value of An. Check the following code sequence to determine what value is stored for each case.

```
MOVEA.L #$1000,A0
MOVES.L A0,(A0) +
MOVES.L A0,D(A0)
```

Because the MC68040 implements a merged instruction and data space, the MC68040's integer unit into data references (SFC/DFC = 5 or 1) translates MOVES accesses to the OinstructionO address spaces (SFC/DFC = 6 or 2). The data memory unit handles these translated accesses as normal data accesses. If the access fails due to an ATC fault or a physical bus error, the resulting access error stack frame contains the converted function code in the TM field for the faulted access. To maintain cache coherency, MOVES accesses to write the OinstructionO address space must be preceded by invalidation of the instruction cache line containing the referenced location.

ORI to SR — Inclusive-OR Immediate to the Status Register (M68000 Family) — ORI to SR

Operation:
If Supervisor State
Then Source V SR → SR
Else TRAP

Assembler Syntax: ORI # < data > ,SR

Attributes: Size = (Word)

Description: Performs an inclusive-OR operation of the immediate operand and the status register's contents and stores the result in the status register. All implemented bits of the status register are affected.

Condition Codes:

X	N	Z	V	C
*	*	*	*	*

X—Set if bit 4 of immediate operand is one; unchanged otherwise.

N—Set if bit 3 of immediate operand is one; unchanged otherwise.

Z—Set if bit 2 of immediate operand is one; unchanged otherwise.

V—Set if bit 1 of immediate operand is one; unchanged otherwise.

C—Set if bit 0 of immediate operand is one; unchanged otherwise.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
16—BIT WORD DATA															

PBcc — Branch on PMMU Condition (MC68851) — PBcc

Operation:

If Supervisor State
  Then If cc True
    Then (PC) + dn → PC
  Else TRAP

Assembler Syntax: PBcc. < size > < label >

Attributes: Size = (Word, Long)

Description: If the specified paged memory management unit condition is met, execution continues at location (PC) + displacement. The displacement is a twos complement integer that counts the relative distance in bytes. The value in the program counter is the address of the displacement word(s). The displacement may be either 16 or 32 bits.

The condition specifier cc indicates the following conditions:

Specifier	Description	Condition Field
BS	B set	000000
LS	L set	000010
SS	S set	000100
AS	A set	000110
WS	W set	001000
IS	I set	001010
GS	G set	001100
CS	C set	001110

Specifier	Description	Condition Field
BC	B clear	000001
LC	L clear	000011
SC	S clear	000101
AC	A clear	000111
WC	W clear	001001
IC	I clear	001011
GC	G clear	001101
CC	C clear	001111

PMMU Status Register: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	SIZE	MC68851 CONDITION					
16-BIT DISPLACEMENT OR MOST SIGNIFICANT WORD OF 32-BITDISPLACEMENT															
LEAST SIGNIFICANT WORD OF 32-BIT DISPLACEMENT (IF NEEDED)															

PBcc Branch on PMMU Condition (MC68851) PBcc

Instruction Fields:

Size field—Specifies the size of the displacement.
0—Displacement is 16 bits.
1—Displacement is 32 bits.

MC68851 Condition field—Specifies the coprocessor condition to be tested. This field is passed to the MC68851, which provides directives to the main processor for processing this instruction.

Word Displacement field—The shortest displacement form for MC68851 branches is 16 bits.

Long-Word Displacement field—Allows a displacement larger than 16 bits.

PDBcc Test, Decrement, and Branch (MC68851) PDBcc

Operation: If Supervisor State
Then If cc False
Then (DnD1 → Dn; If Dn < > D1 then (PC) + d 3 PC)
Else No Operation
Else TRAP

Assembler Syntax: PDBcc Dn, < label >

Attributes: Size = (Word)

Description: This instruction is a looping primitive of three parameters: an MC68851 condition, a counter (an MC68020 data register), and a 16-bit displacement. The instruction first tests the condition to determine if the termination condition for the loop has been met. If so, the main processor executes the next instruction in the instruction stream. If the termination condition is not true, the low-order 16 bits of the counter register are decremented by one. If the result is not D1, execution continues at the location specified by the current value of the program counter plus the sign-extended 16-bit displacement. The value of the program counter used in the branch address calculation is the address of the PDBcc instruction plus two.

The condition specifier cc indicates the following conditions:

Specifier	Description	Condition Field
BS	B set	000000
LS	L set	000010
SS	S set	000100
AS	A set	000110
WS	W set	001000
IS	I set	001010
GS	G set	001100
CS	C set	001110

Specifier	Description	Condition Field
BC	B clear	000001
LC	L clear	000011
SC	S clear	000101
AC	A clear	000111
WC	W clear	001001
IC	I clear	001011
GC	G clear	001101
CC	C clear	001111

PDBcc Test, Decrement, and Branch (MC68851) PDBcc

PMMU Status Register: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	0	0	1	COUNT REGISTER		
0	0	0	0	0	0	0	0	0	0	MC68851 CONDITION					
16-BIT DISPLACEMENT															

Instruction Fields:

Register field—Specifies the data register in the main processor to be used as the counter.

MC68851 Condition field—Specifies the MC68851 condition to be tested. This field is passed to the MC68851, which provides directives to the main processor for processing this instruction.

Displacement field—Specifies the distance of the branch in bytes.

PFLUSH Flush Entry in the ATC (MC68030 only) PFLUSH

Operation:
If Supervisor State
Then Invalidate ATC Entries for Destination Addresses
Else TRAP

Assembler Syntax:
PFLUSHA
PFLUSH FC,MASK
PFLUSH FC,MASK, < ea >

Attributes: Unsized

Description: PFLUSH invalidates address translation cache entries. The instruction has three forms. The PFLUSHA instruction invalidates all entries. When the instruction specifies a function code and mask, the instruction invalidates all entries for a selected function code(s). When the instruction also specifies an < ea > , the instruction invalidates the page descriptor for that effective address entry in each selected function code.

The mask operand contains three bits that correspond to the three function code bits. Each bit in the mask that is set to one indicates that the corresponding bit of the FC operand applies to the operation. Each bit in the mask that is zero indicates a bit of FC and of the ignored function code. For example, a mask operand of 100 causes the instruction to consider only the most significant bit of the FC operand. If the FC operand is 001, function codes 000, 001, 010, and 011 are selected.

The FC operand is specified in one of the following ways:

1. Immediate—Three bits in the command word.
2. Data Register—The three least significant bits of the data register specified in the instruction.
3. Source Function Code (SFC) Register
4. Destination Function Code (DFC) Register

Condition Codes:

Not affected.

MMU Status Register:

Not affected.

PFLUSH Flush Entry in the ATC (MC68030 only) PFLUSH

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	1	MODE			0	0	MASK					FC		

Instruction Fields:

Effective Address field—Specifies a control alterable address. The address translation cache entry for this address is invalidated. Valid addressing modes are in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

NOTE

The address field must provide the memory management unit with the effective address to be flushed from the address translation cache, not the effective address describing where the PFLUSH operand is located. For example, to flush the address translation cache entry corresponding to a logical address that is temporarily stored on top of the system stack, the instruction PFLUSH [(SP)] must be used since PFLUSH (SP) would invalidate the address translation cache entry mapping the system stack (i.e., the effective address passed to the memory management unit is the effective address of the system stack, not the effective address formed by the operand located on the top of the stack).

PFLUSH Flush Entry in the ATC (MC68030 only) PFLUSH

Mode field—Specifies the type of flush operation.
001—Flush all entries.
100—Flush by function code only.
110—Flush by function code and effective address.

Mask field—Mask for selecting function codes. Ones in the mask correspond to applicable bits; zeros are bits to be ignored. When mode is 001, mask must be 000.

FC field—Function code of entries to be flushed. If the mode field is 001, FC field must be 00000; otherwise:
10XXX — Function code is specified as bits XXX.
01DDD — Function code is specified as bits 2–0 of data register DDD.
00000 — Function code is specified as SFC register.
00001 — Function code is specified as DFC register.

PFLUSH — Flush ATC Entries (MC68040, MC68LC040) — PFLUSH

Operation:
If Supervisor State
Then Invalidate Instruction and Data ATC Entries for Destination Address
Else TRAP

Assembler Syntax: PFLUSH (An)
Syntax: PFLUSHN (An)
Syntax: PFLUSHA
Syntax: PFLUSHAN

Attributes: Unsized

Description: Invalidates address translation cache entries in both the instruction and data address translation caches. The instruction has two forms. The PFLUSHA instruction invalidates all entries. The PFLUSH (An) instruction invalidates the entry in each address translation cache which matches the logical address in An and the specified function code.

The function code for PFLUSH is specified in the destination function code register. Destination function code values of 1 or 2 will result in flushing of user address translation cache entries in both address translation caches; whereas, values of 5 or 6 will result in flushing of supervisor address translation cache entries. PFLUSH is undefined for destination function code values of 0, 3, 4, and 7 and may cause flushing of an unexpected entry.

The PFLUSHN and PFLUSHAN instructions have a global option specified and invalidate only nonglobal entries. For example, if only page descriptors for operating system code have the global bit set, these two PFLUSH variants can be used to flush only user address translation cache entries during task swaps.

Condition Codes:

Not affected.

PFLUSH — Flush ATC Entries (MC68040, MC68LC040) — PFLUSH

Instruction Format:

Postincrement Source and Destination

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	0	0	OPMODE		REGISTER		

Instruction Fields:

Opmode field—Specifies the flush operation.

Opcode	Operation	Assembler Syntax
00	Flush page entry if not global	PFLUSHN (An)
01	Flush page entry	PFLUSH (An)
10	Flush all except global entries	PFLUSHAN
11	Flush all entries	PFLUSHA

Register field—Specifies the address register containing the effective address to be flushed when flushing a page entry.

PFLUSH — Flush ATC Entries (MC68EC040) — PFLUSH

Operation:
If Supervisor State
Then No Operation
Else TRAP

Assembler Syntax:
PFLUSH (An)
PFLUSHN (An)

Attributes:
Unsized

Description: This instruction should not be executed when using an MC68EC040. The PFLUSH encoding suspends operation of the MC68EC040 for an indefinite period of time and subsequently continues with no adverse effects.

Condition Codes:

Not affected.

Instruction Format:

Postincrement Source and Destination

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	0	0	OPMODE		REGISTER		

Instruction Fields:

Opmode field—Specifies the flush operation.

Opcode	Operation	Assembler Syntax
00	Flush page entry if not global	PFLUSHN (An)
01	Flush page entry	PFLUSH (An)
10	Flush all except global entries	PFLUSHAN
11	Flush all entries	PFLUSHA

Register field—Specifies the address register containing the effective address to be flushed when flushing a page entry.

PFLUSH PFLUSHA PFLUSHS — Invalidate Entries in the ATC (MC68851)

PFLUSH PFLUSHA PFLUSHS

Operation:
If Supervisor State
Then Address Translation Cache Entries For Destination Address Are Invalidated
Else TRAP

Assembler Syntax:
PFLUSHA
PFLUSH FC,MASK
PFLUSHS FC,MASK
PFLUSH FC,MASK, < ea >
PFLUSHS FC,MASK, < ea >

Attributes: Unsigned

Description: PFLUSHA invalidates all entries in the address translation cache.

PFLUSH invalidates a set of address translation cache entries whose function code bits satisfy the relation: (address translation cache function code bits and mask) = (FC and MASK) for all entries whose task alias matches the task alias currently active when the instruction is executed. With an additional effective address argument, PFLUSH invalidates a set of address translation cache entries whose function code satisfies the relation above and whose effective address field matches the corresponding bits of the evaluated effective address argument. In both of these cases, address translation cache entries whose SG bit is set will not be invalidated unless the PFLUSHS is specified.

The function code for this operation may be specified as follows:

1. Immediate—The function code is four bits in the command word.
2. Data Register—The function code is in the lower four bits of the MC68020 data register specified in the instruction.
3. Source Function Code (SFC) Register—The function code is in the CPU SFC register. Since the SFC of the MC68020 has only three implemented bits, only function codes $0D$7 can be specified in this manner.
4. Destination Function Code (DFC) Register—The function code is in the CPU DFC register. Since the DFC of the MC68020 has only three implemented bits, only function codes $0D$7 can be specified in this manner.

PFLUSH PFLUSHA PFLUSHS — Invalidate Entries in the ATC (MC68851)

PFLUSH PFLUSHA PFLUSHS

PMMU Status Register: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	1	MODE			0	MASK						FC		

Instruction Fields:

Effective Address field—Specifies an address whose page descriptor is to be flushed from (invalidated) the address translation cache. Only control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

PFLUSH PFLUSHA PFLUSHS — Invalidate Entries in the ATC (MC68851)

PFLUSH PFLUSHA PFLUSHS

NOTE

The effective address field must provide the MC68851 with the effective address of the entry to be flushed from the address translation cache, not the effective address describing where the PFLUSH operand is located. For example, in order to flush the address translation cache entry corresponding to a logical address that is temporarily stored on the top of the system stack, the instruction PFLUSH [(SP)] must be used since PFLUSH (SP) would invalidate the address translation cache entry mapping the system stack (i.e., the effective address passed to the MC68851 is the effective address of the system stack, not the effective address formed by the operand located on the top of the stack).

Mode field—Specifies how the address translation cache is to be flushed.
- 001—Flush all entries.
- 100—Flush by function code only.
- 101—Flush by function code including shared entries.
- 110—Flush by function code and effective address.
- 111—Flush by function code and effective address including shared entries.

Mask field—Indicates which bits are significant in the function code compare. A zero indicates that the bit position is not significant; a one indicates that the bit position is significant. If mode = 001 (flush all entries), mask must be 0000.

FC field—Function code of address to be flushed. If the mode field is 001 (flush all entries), function code must be 00000; otherwise:
- 1DDDD — Function code is specified as four bits DDDD.
- 01RRR — Function code is contained in CPU data register RRR.
- 00000 — Function code is contained in CPU SFC register.
- 00001 — Function code is contained in CPU DFC register.

PFLUSHR Invalidate ATC and RPT Entries (MC68851) PFLUSHR

Operation:
If Supervisor State
Then RPT Entry (If Any) Matching Root Pointer Specified by < ea >
Corresponding Address Translation Cache Entries Are Invalidated
Else TRAP

Assembler Syntax: PFLUSHR < ea >

Attributes: Unsized

Description: The quad word pointed to by < ea > is regarded as a previously used value of the CPU root pointer register. The root pointer table entry matching this CPU root pointer register (if any) is flushed, and all address translation cache entries loaded with this value of CPU root pointer register (except for those that are globally shared) are invalidated. If no entry in the root pointer table matches the operand of this instruction, no action is taken.

If the supervisor root pointer is not in use, the operating system should not issue the PFLUSHR command to destroy a task identified by the current CPU root pointer register. It should wait until the CPU root pointer register has been loaded with the root pointer identifying the next task until using the PFLUSHR instruction. At any time, execution of the PFLUSHR instruction for the current CPU root pointer register causes the current task alias to be corrupted.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLUSHR Invalidate ATC and RPT Entries PFLUSHR
(MC68851)

Instruction Field:

Effective Address field—Specifies the address of a previous value of the CPU root pointer register register. Only memory addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn] ,od)	111	011
([bd,PC],Xn ,od)	111	011

NOTE

The effective address usage of this instruction is different than that of other PFLUSH variants.

PLOAD — Load an Entry into the ATC (MC68030 only, MC68851) — PLOAD

Operation:
If Supervisor State
Then Search Translation Table and Make Address Translation Cache Entry for Effective Address
Else TRAP

Assembler Syntax:
PLOADR FC, < ea >
PLOADW FC, < ea >

Attributes: Unsized

Description: For the MC68851, PLOAD searches the translation table for a translation of the specified effective address. If one is found, it is flushed from the address translation cache, and an entry is made as if a bus master had run a bus cycle. Used and modified bits in the table are updated as part of the table search. The MC68851 ignores the logical bus arbitration signals during the flush and load phases at the end of this instruction. This prevents the possibility of an entry temporarily disappearing from the address translation cache and causing a false table search.

This instruction will cause a paged memory management unit illegal operation exception (vector $39) if the E-bit of the translation control register is clear.

The function code for this operation may be specified to be:

1. Immediate—The function code is specified as four bits in the command word.
2. Data Register—The function code is contained in the lower four bits in the MC68020 data register specified in the instruction.
3. Source Function Code (SFC) Register—The function code is in the CPU SFC register. Since the SFC of the MC68020 has only three implemented bits, only function codes $0D$7 can be specified in this manner.
4. Destination Function Code (DFC) Register—The function code is in the CPU DFC register. Since the DFC of the MC68020 has only three implemented bits, only function codes $0D$7 can be specified in this manner.

PLOAD — Load an Entry into the ATC (MC68030 only, MC68851) — PLOAD

For the MC68030, PLOAD searches the address translation cache for the specified effective address. It also searches the translation table for the descriptor corresponding to the specified effective address. It creates a new entry as if the MC68030 had attempted to access that address. Sets the used and modified bits appropriately as part of the search. The instruction executes despite the value of the E-bit in the translation control register or the state of the MMUDIS signal.

The < function code > operand is specified in one of the following ways:

1. Immediate—Three bits in the command word.
2. Data Register—The three least significant bits of the data register specified in the instruction.
3. Source Function Code (SFC) Register
4. Destination Function Code (DFC) Register

The effective address field specifies the logical address whose translation is to be loaded.

PLOADR causes U bits in the translation tables to be updated as if a read access had occurred. PLOADW causes U and M bits in the translation tables to be updated as if a write access had occurred.

PMMU Status Register: Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	0	0	0	R/ W	0	0	0	0	FC				

PLOAD — Load an Entry into the ATC (MC68030 only, MC68851) — PLOAD

Instruction Fields:

Effective Address field—Specifies the logical address whose translation is to be loaded into the address translation cache. Only control alterable addressing modes are allowed as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

NOTE

The effective address field must provide the MC68851 with the effective address of the entry to be loaded into the address translation cache, not the effective address describing where the PLOAD operand is located. For example, to load an address translation cache entry to map a logical address that is temporarily stored on the system stack, the instruction PLOAD [(SP)] must be used since PLOAD (SP) would load an address translation cache entry mapping the system stack (i.e., the effective address passed to the MC68851 is the effective address of the system stack, not the effective address formed by the operand located on the top of the stack).

R/W field—Specifies whether the tables should be updated for a read or a write.
1—Read
0—Write

PLOAD — Load an Entry into the ATC (MC68030 only, MC68851) — PLOAD

FC field (MC68851)—Function code of address to load.
- 1DDDD — Function code is specified as four bits DDDD.
- 01RRR — Function code is contained in CPU data register RRR.
- 00000 — Function code is contained in CPU SFC register.
- 00001 — Function code is contained in CPU DFC register.

FC field (MC68030)—Function code of address corresponding to entry to be loaded.
- 10XXX — Function code is specified as bits XXX.
- 01DDD — Function code is specified as bits 2–0 of data register DDD.
- 00000 — Function code is specified as SFC register.
- 00001 — Function code is specified as DFC register.

PMOVE Move to/from MMU Registers (MC68030 only) PMOVE

Operation: If Supervisor State
Then (Source) → MRn or MRn → (Destination)

Assembler Syntax:
PMOVE MRn, < ea >
PMOVE < ea > ,MRn
PMOVEFD < ea > ,MRn

Attributes: Size = (Word, Long, Quad)

Description: Moves the contents of the source effective address to the specified memory management unit register or moves the contents of the memory management unit register to the destination effective address.

The instruction is a quad-word (8 byte) operation for the CPU root pointer and the supervisor root pointer. It is a long-word operation for the translation control register and the transparent translation registers (TT0 and TT1). It is a word operation for the MMU status register.

The PMOVEFD form of this instruction sets the FD-bit to disable flushing the address translation cache when a new value loads into the supervisor root pointer, CPU root pointer, TT0, TT1 or translation control register (but not the MMU status register).

Writing to the following registers has the indicated side effects:

CPU Root Pointer—When the FD-bit is zero, it flushes the address translation cache. If the operand value is invalid for a root pointer descriptor, the instruction takes an memory management unit configuration error exception after moving the operand to the CPU root pointer.

Supervisor Root Pointer—When the FD-bit is zero, it flushes the address translation cache. If the operand value is invalid as a root pointer descriptor, the instruction takes an memory management unit configuration error exception after moving the operand to the supervisor root pointer.

Translation Control Register—When the FD-bit is zero, it flushes the address translation cache. If the E-bit = 1, consistency checks are performed on the PS and TIx fields. If the checks fail, the instruction takes an memory management unit configuration exception after moving the operand to the translation control register. If the checks pass, the translation control register is loaded with the operand and the E-bit is cleared.

TT0, TT1—When the FD-bit is zero, it flushes the address translation cache. It enables or disables the transparent translation register according to the E-bit written. If the E-bit = 1, the transparent translation register is enabled. If the E- bit = 0, the register is disabled.

PMOVE Move to/from MMU Registers (MC68030 only) PMOVE

Condition Codes:

Not affected.

MMU Status Register:

Not affected (unless the MMU status register is specified as the destination operand).

Instruction Format:

SRP, CRP, and TC Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	0	P-REGISTER			R/ W	FD	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Specifies the memory location for the transfer. Only control alterable addressing modes can be used as in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

PMOVE Move to/from MMU Registers (MC68030 only) PMOVE

P-Register field—Specifies the memory management unit register.
000—Translation Control Register
010—Supervisor Root Pointer
011—CPU Root Pointer

R/W field—Specifies the direction of transfer.
0—Memory to memeory management unit register.
1—Memeory management unit register to memory.

FD field—Disables flushing of the address translation cache on writes to memeory management unit registers.
0—Address translation cache is flushed.
1—Address translation cache is not flushed.

Instruction Format:

MMU Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	1	0	0	0	R/ W	0	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Specifies the memory location for the transfer. Control alterable addressing modes shown for supervisor root pointer register apply.

R/W field—Specifies the direction of transfer.
0—Memory to MMU status register.
1—MMU status register to memory.

NOTE

The syntax of assemblers for the MC68851 use the symbol PMMU status register for the MMU status register.

PMOVE Move to/from MMU Registers (MC68030 only) PMOVE

Instruction Format:

TT Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	0	0	P-REGISTER			R/ W	FD	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Specifies the memory location for the transfer. Control alterable addressing modes shown for supervisor root pointer register apply.

P-Register field—Specifies the transparent translation register.
010—Transparent Translation Register 0
011—Transparent Translation Register 1

R/W field—Specifies the direction of transfer.
0—Memory to MMU status register.
1—MMU status register to memory.

FD field—Disables flushing of the address translation cache.
0—Address translation cache is flushed.
1—Address translation cache does not flush.

PMOVE Move to/from MMU Registers (MC68EC030) PMOVE

Operation: If Supervisor State
Then (Source) → MRn or MRn → (Destination)

Assembler Syntax: PMOVE MRn, < ea >
PMOVE < ea > ,MRn

Attributes: Size = (Word, Long, Quad)

Description: Moves the contents of the source effective address to an access control register or moves the contents of an access control register to the destination effective address.

The instruction is a long-word operation for the access control registers (AC0 and AC1). It is a word operation for the access control unit status register (ACUSR).

Writing to the ACx registers enables or disables the access control register according to the E-bit written. If the E-bit = 1, the access control register is enabled. If the E-bit = 0, the register is disabled

Condition Codes:

Not affected.

ACUSR:

Not affected unless the ACUSR is specified as the destination operand.

Instruction Format:

ACUSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	1	0	0	0	R/W	0	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Specifies the memory location for the transfer.

R/W field—Specifies the direction of transfer.
0—Memory to ACUSR
1—ACUSR to memory

PMOVE Move to/from MMU Registers (MC68EC030) PMOVE

NOTE

Assembler syntax for the MC68851 uses the symbol PMMU status register for the ACUSR; and for the MC68030, the symbols TT0 and TT1 for AC0 and AC1.

Instruction Format:

ACx Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	0	0	P-REGISTER			R/W	0	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Specifies the memory location for the transfer.

P-Register field—Specifies the ACx register.
001—Access Control Register 0
011—Access Control Register 1

R/W field—Specifies the direction of transfer.
0—Memory to ACUSR
1—ACUSR to memory

PMOVE Move PMMU Register (MC68851) PMOVE

Operation:
If Supervisor State
Then MC68851 Register →Destination
Or Source → MC68851 Register
Else TRAP

Assembler Syntax:
PMOVE < PMMU Register > , < ea >
PMOVE < ea > , < PMMU Register >

Attributes:
Size = (Byte, Word, Long, Double Long)

Description: The contents of the MC68851 register copies to the address specified by < ea > , or the data at < ea > copies into the MC68851 register.

The instruction is a quad-word operation for CPU root pointer, supervisor root pointer, and DMA root pointer registers. It is a long-word operation for the translation control register and a word operation for the breakpoint acknowledge control, breakpoint acknowledge data, access control, PMMU status, and PMMU cache status registers. PMOVE is a byte operation for the current access level, valid access level, and stack change control registers.

The following side effects occur when data is read into certain registers:

CPU Root Pointer—Causes the internal root pointer table to be searched for the new value. If there is no matching value, an entry in the root pointer table is selected for replacement, and all address translation cache entries associated with the replaced entry are invalidated.

Supervisor Root Pointer—Causes all entries in the address translation cache that were formed with the supervisor root pointer (even globally shared entries) to be invalidated.

DMA Root Pointer—Causes all entries in the address translation cache that were formed with the DMA root pointer (even globally shared entries) to be invalidated.

Translation Control Register—If data written to the translation control register attempts to set the E-bit and the E-bit is currently clear, a consistency check is performed on the IS, TIA, TIB, TIC, TID, and PS fields.

PMOVE Move PMMU Register (MC68851) PMOVE

PMMU Status Register: Not affected unless the PMMU status register is written to by the instruction.

Instruction Format 1:

PMOVE to/from TC, CRP, DRP, SRP, CAL, VAL, SCC, AC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	0	P-REGISTER			R/ W	0	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—for memory-to-register transfers, any addressing mode is allowed as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An*	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	111	100
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn] ,od)	111	011
([bd,PC],Xn ,od)	111	011

*PMOVE to CRP, SRP, and DMA root pointer not allowed with these modes

PMOVE Move PMMU Register (MC68851) PMOVE

For register-to-memory transfers, only alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn*	000	reg. number:Dn
An*	001	reg. number:An
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

*PMOVE to CRP, SRP, and DMA root pointer not allowed with these modes

Register field—Specifies the MC68851 register.
- 000—Translation Control Register
- 001—DMA Root Pointer
- 010—Supervisor Root Pointer
- 011—CPU Root Pointer
- 100—Current Access Level
- 101—Valid Access Level
- 110—Stack Change Control Register
- 111—Access Control Register

R/W field—Specifies the direction of transfer.
- 0—Transfer < ea > to MC68851 register.
- 1—Transfer MC68851 register to < ea > .

Instruction Format 2:

PMOVE to/from BADx, BACx

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	1	P-REGISTER			R/ W	0	0	0	0	NUM			0	0

PMOVE Move PMMU Register (MC68851) PMOVE

Instruction Fields:

Effective Address field—Same as format 1.

P-Register field—Specifies the type of MC68851 register.
100—Breakpoint Acknowledge Data
101—Breakpoint Acknowledge Control

R/W field—Specifies the direction of transfer.
0—Transfer < ea > to MC68851 register
1—Transfer MC68851 register to < ea >

Num field—Specifies the number of the BACx or BADx register to be used.

Instruction Format 3:

PMOVE to/from PSR, from PCSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	1	P-REGISTER			R/ W	0	0	0	0	0	0	0	0	0

Instruction Fields:

Effective Address field—Same as format 1.

P Register field—Specifies the MC68851 register.
000 — PMMU Status Register
001 — PMMU Cache Status Register

R/W field—Specifies direction of transfer.
0—Transfer < ea > to MC68851 register.
1—Transfer MC68851 register to < ea > (must be one to access PMMU cache status register using this format).

PRESTORE PMMU Restore Function (MC68851) PRESTORE

Operation: If Supervisor State
Then MC68851 State Frame → Internal State, Programmer Registers
Else TRAP

Assembler Syntax: PRESTORE < ea >

Attributes: Unsized, Privileged

Description: The MC68851 aborts execution of any operation in progress. New programmer registers and internal states are loaded from the state frame located at the effective address. The first word at the specified address is the format word of the state frame, specifying the size of the frame and the revision number of the MC68851 that created it. The MC68020 writes the first word to the MC68851 restore coprocessor interface register, initiating the restore operation. Then it reads the response coprocessor interface register to verify that the MC68851 recognizes the format as valid. The format is invalid if the MC68851 does not recognize the frame size or the revision number does not match. If the format is invalid, the MC68020 takes a format exception, and the MC68851 returns to the idle state with its user visible registers unchanged. However, if the format is valid, then the appropriate state frame loads, starting at the specified location and proceeding up through the higher addresses.

The PRESTORE instruction restores the nonuser visible state of the MC68851 as well as the PMMU status register, CPU root pointer, supervisor root pointer, current access level, valid access level, and stack change control registers of the user programming model. In addition, if any breakpoints are enabled, all breakpoint acknowledge control and breakpoint acknowledge data registers are restored. This instruction is the inverse of the PSAVE instruction.

The current implementation of the MC68851 supports four state frame sizes:

NULL: This state frame is 4 bytes long, with a format word of $0. A PRESTORE with this size state frame places the MC68851 in the idle state with no coprocessor or module operations in progress.

IDLE: This state frame is 36 ($24) bytes long. A PRESTORE with this size state frame causes the MC68851 to place itself in an idle state with no coprocessor operations in progress and no breakpoints enabled. A module operation may or may not be in progress. This state frame restores the minimal set of MC68851 registers.

PRESTORE PMMU Restore Function (MC68851) PRESTORE

MID-COPROCESSOR: This state frame is 44 ($2C) bytes long. A PRESTORE with this size frame restores the MC68851 to a state with a coprocessor operation in progress and no breakpoints enabled.

BREAKPOINTS ENABLED: This state frame is 76 ($4C) bytes long. A PRESTORE with this size state frame restores all breakpoint registers, along with other states. A coprocessor operation may or may not be in progress.

PMMU Status Register: Set according to restored data.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	1	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:

Effective Address field—Specifies the source location. Only control or post-increment addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011
([bd,PC,Xn] ,od)	111	011
([bd,PC],Xn ,od)	111	011

PSAVE PMMU Save Function (MC68851) PSAVE

Operation:
If Supervisor State
Then MC68851 Internal State, Programmer Registers → State Frame
Else TRAP

Assembler Syntax: PSAVE < ea >

Attributes: Unsized, Privileged

Description: The MC68851 suspends execution of any operation that it is performing and saves its internal state and some programmer registers in a state frame located at the effective address. The following registers are copied: PMMU status, control root pointer, supervisor root pointer, current access level, valid access level, and stack change control. If any breakpoint is enabled, all breakpoint acknowledge control and breakpoint acknowledge data registers are copied. After the save operation, the MC68851 is in an idle state waiting for another operation to be requested. Programmer registers are not changed.

The state frame format saved by the MC68851 depends on its state at the time of the PSAVE operation. In the current implementation, three state frames are possible:

IDLE: This state frame is 36 ($24) bytes long. A PSAVE of this size state frame indicates that the MC68851 was in an idle state with no coprocessor operations in progress and no breakpoints enabled. A module call operation may or may not have been in progress when this state frame was saved.

MID-COPROCESSOR: This state frame is 44 ($2C) bytes long. A PSAVE of this size frame indicates that the MC68851 was in a state with a coprocessor or module call operation in progress and no breakpoints enabled.

BREAKPOINTS ENABLED: This state frame is 76 ($4C) bytes long. A PSAVE of this size state frame indicates that one or more breakpoints were enabled. A coprocessor or module call operation may or may not have been in progress.

PMMU Status Register: Not affected

PSAVE PMMU Save Function (MC68851) PSAVE

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	1	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

Instruction Fields:

Effective Address field—Specifies the destination location. Only control or predecrement addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

PScc — Set on PMMU unit Condition (MC68851) — PScc

Operation:
If Supervisor State
  Then If cc True
    Then 1s → Destination
    Else 0s → Destination
  Else TRAP

Assembler Syntax: PScc < ea >

Attributes: Size = (Byte)

Description: The specified MC68851 condition code is tested. If the condition is true, the byte specified by the effective address is set to TRUE (all ones); otherwise, that byte is set to FALSE (all zeros).

The condition code specifier cc may specify the following conditions:

Specifier	Description	Condition Field
BS	B set	000000
LS	L set	000010
SS	S set	000100
AS	A set	000110
WS	W set	001000
IS	I set	001010
GS	G set	001100
CS	C set	001110

Specifier	Description	Condition Field
BC	B clear	000001
LC	L clear	000011
SC	S clear	000101
AC	A clear	000111
WC	W clear	001001
IC	I clear	001011
GC	G clear	001101
CC	C clear	001111

PMMU Status Register: Not affected

PScc — Set on PMMU Condition (MC68851) — PScc

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	0	0	0	0	0	0	0	0	MC68851 CONDITION					

Instruction Fields:

Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	000	reg. number:Dn
An	—	—
(An)	010	reg. number:An
(An) +	011	reg. number:An
—(An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

MC68851 Condition field—Specifies the coprocessor condition to be tested. This field is passed to the MC68851, which provides directives to the main processor for processing this instruction.

PTEST — Test a Logical Address (MC68030 only) — PTEST

Operation:
If Supervisor State
Then Logical Address Status → MMUSR
Else TRAP

Assembler Syntax:
PTESTR FC, < ea > ,# < level >
PTESTR FC, < ea > ,# < level > ,An
PTESTW FC, < ea > ,# < level >
PTESTW FC, < ea > ,# < level > ,An

Attributes: Unsized

Description: This instruction searches the address translation cache or the translation tables to a specified level. Searching for the translation descriptor corresponding to the < ea > field, it sets the bits of the MMU status register according to the status of the descriptor. Optionally, PTEST stores the physical address of the last table entry accessed during the search in the specified address register. The PTEST instruction searches the address translation cache or the translation tables to obtain status information, but alters neither the used or modified bits of the translation tables nor the address translation cache. When the level operand is zero, only the transparent translation of either read or write accesses causes the operations of the PTESTR and PTESTW to return different results.

The < function code > operand is specified as one of the following:

1. Immediate—Three bits in the command word.
2. Data Register—The three least significant bits of the data register specified in the instruction.
3. Source Function Code (SFC) Register
4. Destination Function Code (DFC) Register

The effective address is the address to test. The < level > operand specifies the level of the search. Level 0 specifies searching the addrass translation cache only. Levels 1–7 specify searching the translation tables only. The search ends at the specified level. A level 0 test does not return the same MMU status register values as a test at a nonzero level number.

Execution of the instruction continues to the requested level or until detecting one of the following conditions:

- Invalid Descriptor
- Limit Violation
- Bus Error Assertion (Physical Bus Error)

PTEST Test a Logical Address (MC68030 only) PTEST

The instruction accumulates status as it accesses successive table entries. When the instruction specifies an address translation cache search with an address register operand, the MC68030 takes an F-line unimplemented instruction exception.

If there is a parameter specification for a translation table search, the physical address of the last descriptor successfully fetched loads into the address register. A successfully fetched descriptor occurs only if all portions of the descriptor can be read by the MC68030 without abnormal termination of the bus cycle. If the root pointer's DT field indicates page descriptor, the returned address is $0. For a long descriptor, the address of the first long word is returned. The size of the descriptor (short or long) is not returned and must be determined from a knowledge of the translation table.

Condition Codes:

Not affected.

MMUSR:

B	L	S		W	I	M			T				N
*	*	*	0	*	*	*	0	0	0	0	0	0	*

PTEST Test a Logical Address (MC68030 only) PTEST

The MMU status register contains the results of the search. The values in the fields of the MMU status register for an address translation cache search are given in the following table:

MMUSR Bit	PTEST, Level 0	PTEST, Levels 1–7
Bus Error (B)	This bit is set if the bus error bit is set in the ATC entry for the specified logical address.	This bit is set if a bus error is encountered during the table search for the PTEST instruction.
Limit (L)	This bit is cleared.	This bit is set if an index exceeds a limit during the table search.
Supervis or Violatio n (S)	This bit is cleared.	This bit is set if the S-bit of a long (S) format table descriptor or long format page descriptor encountered during the search is set and if the FC2-bit of the function code specified by the PTEST instruction is not equal to one. The S-bit is undefined if the I-bit is set.
Write Protecte d (W)	The bit is set if the WP-bit of the ATC entry is set. It is undefined if the I-bit is set.	This bit is set if a descriptor or page descriptor is encountered with the WP-bit set during the table search. The W-bit is undefined if the I-bit is set.
Invalid (I)	This bit indicates an invalid translation. The I- bit is set if the translation for the specified logical address is not resident in the ATC or if the B-bit of the corresponding ATC entry is set.	This bit indicates an invalid translation. The I-bit is set if the DT field of a table or a page descriptor encountered during the search is set to invalid or if either the B or L bits of the MMUSR are set during the table search.
Modified (M)	This bit is set if the ATC entry corresponding to the specified address has the modified bit set. It is undefined if the I-bit is set.	This bit is set if the page descriptor for the specified address has the modified bit set. It is undefined if I-bit is set.
Transparent (T)	This bit is set if a match occurred in either (or both) of the transparent translation registers (TT0 or TT1).	This bit is set to zero.
Number of Levels (N)	This 3-bit field is set to zero.	This 3-bit field contains the actual number of tables accessed during the search.

PTEST Test a Logical Address (MC68030 only) PTEST

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
1	0	0	LEVEL			R/ W	A	REGISTER					FC		

Instruction Fields:

Effective Address field—Specifies the logical address to be tested. Only control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

Level field—Specifies the highest numbered level to be searched in the table. When this field contains 0, the A field and the register field must also be 0. The instruction takes an F-line exception when the level field is 0 and the A field is not 0.

R/W field—Specifies simulating a read or write bus cycle (no difference for MC68030 MMU).
0—Write
1—Read

A field—Specifies the address register option.
0—No address register.
1—Return the address of the last descriptor searched in the address register specified in the register field.

PTEST Test a Logical Address (MC68030 only) PTEST

Register field—Specifies an address register for the instruction. When the A field contains 0, this field must contain 0.

FC field—Function code of address to be tested.

10XXX — Function code is specified as bits XXX.
01DDD — Function code is specified as bits 2–0 of data register DDD.
00000 — Function code is specified as source function code register.
00001 — Function code is specified as destination function code register.

PTEST Test a Logical Address (MC68EC030) PTEST

Operation: If Supervisor State
Then Logical Address Status → ACUSR
Else TRAP

Assembler Syntax: PTESTR FC, < ea >
PTESTW FC, < ea >

Attributes: Unsized

Description: This instruction searches the access control registers for the address descriptor corresponding to the < ea > field and sets the bit of the access control unit status register (ACUSR) according to the status of the descriptor.

The < function code > operand is specified in one of the following ways:

1. Immediate—Three bits in the command word.
2. Data Register—The three least significant bits of the data register specified in the instruction.
3. Source Function Code (SFC) Register
4. Destination Function Code (DFC) Register

The effective address is the address to test.

Condition Codes:

Not affected.

ACUSR:

x	x	x	0	x	x	x	0	0	AC	0	0	0	x	x	x

x = May be 0 or 1.

The AC-bit is set if a match occurs in either (or both) of the access control registers.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	0	0	0	0	R/ W	0	REGISTER					FC		

PTEST — Test a Logical Address (MC68EC030) — PTEST

Instruction Fields:

Effective Address field—Specifies the logical address to be tested. Only control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

R/W field—Specifies simulating a read or write bus cycle.
0—Write
1—Read

Register field—Specifies an address register for the instruction. When the A field contains 0, this field must contain 0.

FC field—Function code of address to be tested.
10XXX — Function code is specified as bits XXX.
01DDD — Function code is specified as bits 2–0 of data register DDD.
00000 — Function code is specified as source function code register.
00001 — Function code is specified as destination function code register.

NOTE

Assembler syntax for the MC68030 is PTESTR FC, < ea > ,#0 and PTESTW FC, < ea > ,#0.

PTEST Test a Logical Address (MC68040, MC68LC040) PTEST

Operation: If Supervisor State
Then Logical Address Status → MMUSR; Entry → ATC
Else TRAP

Assembler Syntax: PTESTR (An)
PTESTW (An)

Attributes: Unsized

Description: This instruction searches the translation tables for the page descriptor corresponding to the test address in An and sets the bits of the MMU status register according to the status of the descriptors. The upper address bits of the translated physical address are also stored in the MMU status register. The PTESTR instruction simulates a read access and sets the U-bit in each descriptor during table searches; PTESTW simulates a write access and also sets the M-bit in the descriptors, the address translation cache entry, and the MMU status register.

A matching entry in the address translation cache (data or instruction) specified by the function code will be flushed by PTEST. Completion of PTEST results in the creation of a new address translation cache entry. The specification of the function code for the test address is in the destination function code (DFC) register. A PTEST instruction with a DFC value of 0, 3, 4, or 7 is undefined and will return an unknown value in the MMUSR.

Execution of the instruction continues until one of the following conditions occurs:

- Match with one of the two transparent translation registers.
- Transfer Error Assertion (physical transfer error)
- Invalid Descriptor
- Valid Page Descriptor

Condition Codes:

Not affected.

MMU Status Register:

PHYSICAL ADDRESS	B	G	U1	U0	S	CM	M		W	T	R
*	*	*	*	*	*	*	*	0	*	*	*

PTEST Test a Logical Address (MC68040, MC68LC040) PTEST

The MMUSR contains the results of the search. The values in the fields of the MMUSR for a search are:

Physical Address—This 20-bit field contains the upper bits of the translated physical address. Merging these bits with the lower bits of the logical address forms the actual physical address.

Bus Error (B)—Set if a transfer error is encountered during the table search for the PTEST instruction. If this bit is set, all other bits are zero.

Globally Shared (G)—Set if the G-bit is set in the page descriptor.

User Page Attributes (U1, U0)—Set if corresponding bits in the page descriptor are set.

Supervisor Protection (S)—Set if the S-bit in the page descriptor is set. This bit does not indicate that a violation has occurred.

Cache Mode (CM)—This 2-bit field is copied from the CM-bit in the page descriptor.

Modified (M)—Set if the M-bit is set in the page descriptor associated with the address.

Write Protect (W)—Set if the W-bit is set in any of the descriptors encountered during the table search. Setting of this bit does not indicate that a violation occurred.

Transparent Translation Register Hit (T)—Set if the PTEST address matches an instruction or data transparent translation register and the R-bit is set; all other bits are zero.

Resident (R)—Set if the PTEST address matches a transparent translation register or if the table search completes by obtaining a valid page descriptor.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	1	R/ W	0	1	REGISTER		

Instruction Fields:

R/W field—Specifies simulating a read or write bus transfer.
0—Write
1—Read

Register field—Specifies the address register containing the effective address for the instruction.

PTEST — Test a Logical Address (MC68EC040) — PTEST

Operation:
If Supervisor State
Then No Operation, Possibly Run Extraneous Bus Cycles
Else TRAP

Assembler Syntax:
PTESTR (An)
PTESTW (An)

Attributes:
Unsized

Description: This instruction must not be executed on an MC68EC040. This instruction may cause extraneous bus cycles to occur and may result in unexpected exception types.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	1	R/ W	0	1	REGISTER		

Instruction Fields:

R/W field—Specifies simulating a read or write bus transfer.
0—Write
1—Read

Register field—Specifies the address register containing the effective address for the instruction.

PTEST Get Information About Logical Address (MC68851) PTEST

Operation: If Supervisor State
Then Information About Logical Address → PSTATUS
Else TRAP

Assembler Syntax: PTESTR FC, < ea > ,# < level > ,(An)
PTESTW FC, < ea > ,# < level > ,(An)

Attributes: Unsized

Description: If the E-bit of the translation control register is set, information about the logical address specified by FC and < ea > is placed in the PMMU status register. If the E-bit of the translation control register is clear, this instruction will cause a paged memory management unit illegal operation exception (vector $39).

The function code for this operation may be specified as follows:

1. Immediate—The function code is four bits in the command word.
2. Data Register—The function code is in the lower four bits in the MC68020 data register specified in the instruction.
3. Source Function Code (SFC) Register—The function code is in the SFC register in the CPU. Since the SFC of the MC68020 has only three implemented bits, only function codes $0D$7 can be specified in this manner.
4. Destination Function Code (DFC) Register—The function code is in the DFC register in the CPU. Since the DFC of the MC68020 has only three implemented bits, only function codes $0D$7 can be specified in this manner.

The effective address field specifies the logical address to be tested.

The # < level > parameter specifies the depth to which the translation table is to be searched. A value of zero specifies a search of the address translation cache only. Values 1–7 cause the address translation cache to be ignored and specify the maximum number of descriptors to fetch.

NOTE

Finding an address translation cache entry with < level > set to zero may result in a different value in the PMMU status register than forcing a table search. Only the I, W, G, M, and C bits of the PMMU status register are always the same in both cases.

PTEST Get Information About Logical Address (MC68851) PTEST

Either PTESTR or PTESTW must be specified. These two instructions differ in the setting of the A-bit of the PMMU status register. For systems where access levels are not in use, either PTESTR or PTESTW may be used. U and M bits in the translation table are not modified by this instruction.

If there is a specified address register parameter, the physical address of the last successfully fetched descriptor is loaded into the address register. A descriptor is successfully fetched if all portions of the descriptor can be read by the MC68851 without abnormal termination of the bus cycle. If the DT field of the root pointer used indicates page descriptor, the returned address is $0.

The PTEST instruction continues searching the translation tables until reaching the requested level or until a condition occurs that makes further searching impossible (i.e., a DT field set to invalid, a limit violation, or a bus error from memory). The information in the PMMU status register reflects the accumulated values.

PMMU Status Register:

Bus Error (B)—Set if a bus error was received during a descriptor fetch, or if < level > = 0 and an entry was found in the address translation cache with its BERR bit set; cleared otherwise.

Limit (L)—Set if the limit field of a long descriptor was exceeded; cleared otherwise.

Supervisor Violation (S)—Set if a long descriptor indicated supervisor-only access and the < fc > parameter did not have bit 2 set; cleared otherwise.

Access Level Violation (A)—If PTESTR was specified, set if the RAL field of a long descriptor would deny access. If PTESTW was specified, set if a WAL or RAL field of a long descriptor would deny access; cleared otherwise.

Write Protection (W)—Set if the WP-bit of a descriptor was set or if a WAL field of a long descriptor would deny access; cleared otherwise.

Invalid (I)—Set if a valid translation was not available; cleared otherwise.

Modified (M)—If the tested address is found in the address translation cache, set to the value of the M-bit in the address translation cache. If the tested address is found in the translation table, set if the M-bit of the page descriptor is set; cleared otherwise.

PTEST Get Information About Logical Address (MC68851) PTEST

Gate (G)—If the tested address is found in the address translation cache, set to the value of the G-bit in the address translation cache. If the tested address is found in the translation table, set if the G-bit of the page descriptor is set; cleared otherwise.

Globally Shared (C)—Set if the address is globally shared; cleared otherwise.

Level Number (N)—Set to the number of levels searched. A value of zero indicates an early termination of the table search in the root pointer (DT = page descriptor) if the level specification was not zero. If the level specification was zero, N is always set to zero.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	0	LEVEL			R/ W	A-REGISTER			FC					

PTEST Get Information About Logical Address (MC68851) PTEST

Instruction Fields:

Effective Address field—Specifies the logical address about which information is requested. Only control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

NOTE

The effective address field must provide the MC68851 with the effective address of the logical address to be tested, not the effective address describing where the PTEST operand is located. For example, to test a logical address that is temporarily stored on the system stack, the instruction PTEST [(SP)] must be used since PTEST (SP) would test the mapping of the system stack (i.e., the effective address passed to the MC68851 is the effective address of the system stack, not the effective address formed by the operand located on the top of the stack).

PTEST Get Information About Logical Address (MC68851) PTEST

Level field—Specifies the depth to which the translation table should be searched.

R/W field—Specifies whether the A-bit should be updated for a read or a write.
1—Read
0—Write

A-Register field—Specifies the address register in which to load the last descriptor address.
0xxx — Do not return the last descriptor address to an address register.
1RRR — Return the last descriptor address to address register RRR.

NOTE

When the PTEST instruction specifies a level of zero, the A-register field must be 0000. Otherwise, an F-line exception is generated.

FC field—Function code of address to test.
1DDDD — Function code is specified as four bits DDDD.
01RRR — Function code is contained in CPU data register RRR.
00000 — Function code is contained in CPU source function code register.
00001 — Function code is contained in CPU destination function code register.

PTRAPcc TRAP on PMMU Condition (M68851) PTRAPcc

Operation:
If Supervisor State
Then If cc True
Then TRAP
Else TRAP

Assembler Syntax: PTRAPcc
PTRAPcc.W # < data > PTRAPcc.L # < data >

Attributes: Unsized or Size = (Word, Long)

Description: If the selected MC68851 condition is true, the processor initiates exception processing. The vector number is generated referencing the cpTRAPcc exception vector; the stacked program counter is the address of the next instruction. If the selected condition is not true, no operation is performed, and execution continues with the next instruction. The immediate data operand is placed in the next word(s) following the MC68851 condition and is available for user definition to be used within the trap handler. Following the condition word, there may be a user-defined data operand, specified as immediate data, to be used by the trap handler.

The condition specifier cc may specify the following conditions:

Specifier	Description	Condition Field
BS	B set	000000
LS	L set	000010
SS	S set	000100
AS	A set	000110
WS	W set	001000
IS	I set	001010
GS	G set	001100
CS	C set	001110

Specifier	Description	Condition Field
BC	B clear	000001
LC	L clear	000011
SC	S clear	000101
AC	A clear	000111
WC	W clear	001001
IC	I clear	001011
GC	G clear	001101
CC	C clear	001111

PMMU Status Register: Not affected

PTRAPcc TRAP on PMMU Condition (M68851) PTRAPcc

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	1	1	1	OPMODE		
0	0	0	0	0	0	0	0	0	0	MC68851 CONDITION					
16-BIT OPERAND OR MOST SIGNIFICANT WORD OF 32-BIT OPERAND (IFNEEDED)															
LEAST SIGNIFICANT WORD OF 32-BIT OPERAND (IF NEEDED)															

Instruction Fields:

Opmode field—Selects the instruction form.
- 010 — Instruction is followed by one operand word.
- 011 — Instruction is followed by two operand words.
- 100 — Instruction has no following operand words.

MC68851 Condition field—Specifies the coprocessor condition to be tested. This field is passed to the MC68851, which provides directives to the main processor for processing this instruction.

PVALID — Validate a Pointer (MC68851) — PVALID

Operation: If (Source AL Bits) → (Destination AL Bits)
Then TRAP

Assembler Syntax:
PVALID VAL, < ea >
PVALID An, < ea >

Attributes: Size = (Long)

Description: The upper bits of the source, VAL or An, compare with the upper bits of the destination, < ea > . The ALC field of the access control register defines the number of bits compared. If the upper bits of the source are numerically greater than (less privileged than) the destination, they cause a memory management access level exception. Otherwise, execution continues with the next instruction. If the MC field of the access control register = 0, then this instruction always causes a paged memory management unit access level exception.

PMMU Status Register: Not affected.

Instruction Format 1:

VAL Contains Access Level to Test Against

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

PVALID — Validate a Pointer (MC68851) — PVALID

Instruction Field:

Effective Address field—Specifies the logical address to be evaluated and compared against the valid access level register. Only control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

PVALID — Validate a Pointer (MC68851) — PVALID

Instruction Format 2:

Main Processor Register Contains Access Level to Test Against

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	0	1	0	0	0	0	0	0	0	0	REGISTER		

Instruction Fields:

Effective Address field—Specifies the logical address to be evaluated and compared against specified main processor address register. Only control alterable addressing modes can be used as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
—(An)	—	—
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An
([bd,An,Xn] ,od)	110	reg. number:An
([bd,An],Xn ,od)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	—	—
(d_8,PC,Xn)	—	—
(bd,PC,Xn)	—	—
([bd,PC,Xn] ,od)	—	—
([bd,PC],Xn ,od)	—	—

NOTE

The effective address field must provide the MC68851 with the effective address of the logical address to be validated, not the effective address describing where the PVALID operand is located. For example, to validate a logical address that is temporarily stored on the system stack, the instruction PVALID VAL,[(SP)] must be used since PVALID VAL,(SP) would validate the mapping on the system stack (i.e., the effective address passed to the MC68851 is the effective address of the system stack, not the effective address formed by the operand located on the top of the stack).

Register field—Specifies the main processor address register to be used in the compare.

RESET — Reset External Devices (M68000 Family) — RESET

Operation: If Supervisor State
Then Assert $\overline{\text{RESET}}$ ($\overline{\text{RSTO}}$, MC68040 Only) Line
Else TRAP

Assembler Syntax: RESET

Attributes: Unsized

Description: Asserts the $\overline{\text{RSTO}}$ signal for 512 (124 for MC68000, MC68EC000, MC68HC000, MC68HC001, MC68008, MC68010, and MC68302) clock periods, resetting all external devices. The processor state, other than the program counter, is unaffected, and execution continues with the next instruction.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

RTE — Return from Exception (M68000 Family) — RTE

Operation: If Supervisor State
Then (SP) → SR; SP + 2 → SP; (SP) → PC; SP + 4 → SP; Restore State and Deallocate Stack According to (SP)
Else TRAP

Assembler Syntax: RTE

Attributes: Unsized

Description: Loads the processor state information stored in the exception stack frame located at the top of the stack into the processor. The instruction examines the stack format field in the format/offset word to determine how much information must be restored.

Condition Codes:

Set according to the condition code bits in the status register value restored from the stack.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1

Format/Offset Word (in Stack Frame):

MC68010, MC68020, MC68030, MC68040, CPU32

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FORMAT				0	0	VECTOR OFFSET									

Format Field of Format/Offset Word:

Contains the format code, which implies the stack frame size (including the format/offset word). For further information, refer to **Appendix B Exception Processing Reference**.

STOP Load Status Register and Stop (M68000 Family) STOP

Operation: If Supervisor State
Then Immediate Data → SR; STOP
Else TRAP

Assembler Syntax: STOP # < data >

Attributes: Unsized

Description: Moves the immediate operand into the status register (both user and supervisor portions), advances the program counter to point to the next instruction, and stops the fetching and executing of instructions. A trace, interrupt, or reset exception causes the processor to resume instruction execution. A trace exception occurs if instruction tracing is enabled (T0 = 1, T1 = 0) when the STOP instruction begins execution. If an interrupt request is asserted with a priority higher than the priority level set by the new status register value, an interrupt exception occurs; otherwise, the interrupt request is ignored. External reset always initiates reset exception processing.

Condition Codes:

Set according to the immediate operand.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0
IMMEDIATE DATA															

Instruction Fields:

Immediate field—Specifies the data to be loaded into the status register.

SECTION 7
CPU32 INSTRUCTIONS

This section describes the instructions provided for the CPU32. The CPU32 can execute object code from an MC68000 and MC68010 and many of the instructions of the MC68020.

There are three new instructions provided for the CPU32: enter background mode (BGND), low-power stop (LPSTOP), and table lookup and interpolate (TBLS, TBLSN, TBLU, and TBLUN). Table 7-1 lists the MC68020 instructions not supported by the CPU32.

Table 7-1. MC68020 Instructions Not Supported

Mnemonic	Description
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
CALLM	CALL Module
CAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
cpBcc	Branch on Coprocessor Condition
cpDBcc	Test Coprocessor Condition Decrement and Branch
cpGEN	Coprocessor General Function
cpRESTORE	Coprocessor Restore Function
cpSAVE	Coprocessor Save Function
cpScc	Set on Coprocessor Condition
cpTRAPcc	Trap on Coprocessor Condition
RTM	Return from Module
PACK	Pack BCD
UNPK	Unpack BCD

Addressing in the CPU32 is register oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. This flexibility eliminates the need for extra instructions to store register contents in memory. Table 7- 2 lists the M68000 family addressing modes with cross-references to the MC68000, MC68010, CPU32, and MC68020. When referring to instructions in the previous sections, refer to Table 7-2 to identify the addressing modes available to the CPU32. Table 7-3 lists the instructions for the CPU32.

Table 7-2. M68000 Family Addressing Modes

Addressing Mode	Syntax	MC68000 MC68010	CPU32	MC68020
Register Indirect	Rn	X	X	X
Address Register Indirect	(An)	X	X	X
Address Register Indirect with Postincrement	(An) +	X	X	X
Address Register Indirect with Postdecrement	– (An)	X	X	X
Address Register Indirect with Displacement	(d_{16},An)	X	X	X
Address Register Indirect with Index (8-Bit Displacement)	(d_8,An,Xn)	X	X	X
Address Register Indirect with Index (Base Displacement)	(d_8,An,Xn∗SCALE)		X	X
Memory Indirect with Postincrement	([bd,An],Xn, od)			X
Memory Indirect with Preincrement	([bd,An],Xn, od)			X
Absolute Short	(xxx).W	X	X	X
Absolute Long	(xxx).L	X	X	X
Program Counter Indirect with Displacement	(d_{16},PC)	X	X	X
Program Counter Indirect with Index (8-Bit Displacement)	(d_8,PC,Xn)	X	X	X
Program Counter Indirect with Index (Base Displacement)	(d_8,PC,Xn*SC ALE)		X	X
Immediate	# < data >	X	X	X
PC Memory Indirect with Postincrement	([bd,PC],Xn, od)			X
PC Memory Indirect with Predecrement	([bd,PC],Xn, od)			X

NOTE: Xn,SIZE*SCALE—Denotes index register n (data or address), the index size (W for word, L for long word and scale factor (1, 2, 4, or 8 for no-word, long-word, or 8 for quad- word scaling, respectively).
X—Supported

Table 7-3. CPU32 Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BGND	Enter Background Mode
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bound
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
CMP2	Compare Register Against Upper and Lower Bounds
DBcc	Test Condition, Decrement, and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT, LSR	Sign-Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LPSTOP	Low Power Stop
LSL, LSR	Logical Shift Left and Right

Mnemonic	Description
MOVE	Move
MOVEA	Move Address
MOVE from CCR	Move Condition Code Register
Move from SR	Move from Status Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
PEA	Push Effective Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTR	Return and Restore Codes
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TBLS, TBLSN	Signed/Unsigned Table Lookup and Interpolate
TBLU, TBLUN	Signed/Unsigned Table Lookup and Interpolate
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap an Overflow
TST	Test Operand
UNLK	Unlink

BGND — Enter Background Mode (CPU32) — BGND

Operation:
If Background Mode Enabled
    Then Enter Background Mode
Else Format/Vector Offset → – (SSP);
    PC → – (SSP)
    SR → – (SSP)
    (Vector) → PC

Assembler Syntax: BGND

Attributes: Size = (Unsized)

Description: The processor suspends instruction execution and enters background mode if background mode is enabled. The freeze output is asserted to acknowledge entrance into background mode. Upon exiting background mode, instruction execution continues with the instruction pointed to by the current program counter. If background mode is not enabled, the processor initiates illegal instruction exception processing. The vector number is generated to reference the illegal instruction exception vector. Refer to the appropriate user's manual for detailed information on background mode.

Condition Codes:

X	N	Z	V	C
—	—	—	—	—

X — Not affected.
N — Not affected.
Z — Not affected.
V — Not affected.
C — Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	1	1	1	0	1	0

LPSTOP Low-Power Stop (CPU32) LPSTOP

Operation:
If Supervisor State
Immediate Data → SR
Interrupt Mask → External Bus Interface (EBI)
STOP
Else TRAP

Assembler Syntax: LPSTOP # < data >

Attributes: Size = (Word) Privileged

Description: The immediate operand moves into the entire status register, the program counter advances to point to the next instruction, and the processor stops fetching and executing instructions. A CPU LPSTOP broadcast cycle is executed to CPU space $3 to copy the updated interrupt mask to the external bus interface (EBI). The internal clocks are stopped.

Instruction execution resumes when a trace, interrupt, or reset exception occurs. A trace exception will occur if the trace state is on when the LPSTOP instruction is executed. If an interrupt request is asserted with a higher priority that the current priority level set by the new status register value, an interrupt exception occurs; otherwise, the interrupt request is ignored. If the bit of the immediate data corresponding to the S-bit is off, execution of the instruction will cause a privilege violation. An external reset always initiates reset exception processing.

Condition Codes:

Set according to the immediate operand.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
IMMEDIATE DATA															

Instruction Fields:

Immediate field—Specifies the data to be loaded into the status register.

TBLS TBLSN — Table Lookup and Interpolate (Signed) (CPU32)

TBLS TBLSN

Operation:

Rounded:

ENTRY(n) + {(ENTRY(n + 1) – ENTRY(n)) x Dx 7 – 0} ÷ 256 → Dx

Unrounded:

ENTRY(n) x 256 + {(ENTRY(n + 1) – ENTRY(n)) x Dx 7 – 0} → Dx

Where ENTRY(n) and ENTRY(n + 1) are either:

1. Consecutive entries in the table pointed to by the < ea > and indexed by Dx 15 – 8 π SIZE or;
2. The registers Dym, Dyn respectively.

Assembler Syntax:

TBLS. < size > < ea > ,Dx	Result rounded
TBLSN. < size > < ea > ,Dx	Result not rounded
TBLS. < size > Dym:Dyn, Dx	Result rounded
TBLSN. < size > Dym:Dyn, Dx	Result not rounded

Attributes: Size = (Byte, Word, Long)

Description: The TBLS and TBLSN instructions allow the efficient use of piecewise linear compressed data tables to model complex functions. The TBLS instruction has two modes of operation: table lookup and interpolate mode and data register interpolate mode.

For table lookup and interpolate mode, data register Dx 15 – 0 contains the independent variable X. The effective address points to the start of a signed byte, word, or long-word table containing a linearized representation of the dependent variable, Y, as a function of X. In general, the independent variable, located in the low-order word of Dx, consists of an 8-bit integer part and an 8-bit fractional part. An assumed radix point is located between bits 7 and 8. The integer part, Dx 15 – 8, is scaled by the operand size and is used as an offset into the table. The selected entry in the table is subtracted from the next consecutive entry. A fractional portion of this difference is taken by multiplying by the interpolation fraction, Dx 7 – 0 .The adjusted difference is then added to the selected table entry. The result is returned in the destination data register, Dx.

TBLS TBLSN — Table Lookup and Interpolate (Signed) (CPU32) — TBLS TBLSN

For register interpolate mode, the interpolation occurs using the Dym and Dyn registers in place of the two table entries. For this mode, only the fractional portion, Dx 7 – 0, is used in the interpolation, and the integer portion, Dx 15 – 8, is ignored. The register interpolation mode may be used with several table lookup and interpolations to model multidimensional functions.

Signed table entries range from -2^{n-1} to $2^{n-1} - 1$; whereas, unsigned table entries range from 0 to 2^{n-1} where n is 8, 16, or 32 for byte, word, and long-word tables, respectively.

Rounding of the result is optionally selected via the "R" instruction field. If R = 0 (TABLE), the fractional portion is rounded according to the round-to-nearest algorithm. The following table summerizes the rounding procedure:

Adjusted Difference Fraction	Rounding Adjustment
≤ – 1/2	– 1
> – 1/2 and < 1/2	+ 0
≥ 1/2	+ 1

The adjusted difference is then added to the selected table entry. The rounded result is returned in the destination data register, Dx. Only the portion of the register corresponding to the selected size is affected.

	31 24	23 16	15 8	7 0
BYTE	UNAFFECTED	UNAFFECTED	UNAFFECTED	RESULT
WORD	UNAFFECTED	UNAFFECTED	RESULT	RESULT
LONG	RESULT	RESULT	RESULT	RESULT

TBLS TBLSN — Table Lookup and Interpolate (Signed) (CPU32) — TBLS TBLSN

If R = 1 (TABLENR), the result is returned in register Dx without rounding. If the size is byte, the integer portion of the result is returned in Dx 15 – 8; the integer portion of a word result is stored in Dx 23 – 8; the least significant 24 bits of a long result are stored in Dx 31 – 8. Byte and word results are sign-extended to fill the entire 32-bit register.

	31 – 24	23 – 16	15 – 8	7 – 0
BYTE	SIGN-EXTENDED	SIGN-EXTENDED	RESULT	FRACTION
WORD	SIGN-EXTENDED	RESULT	RESULT	FRACTION
LONG	RESULT	RESULT	RESULT	FRACTION

NOTE

The long-word result contains only the least significant 24 bits of integer precision.

For all sizes, the 8-bit fractional portion of the result is returned to the low byte of the data register, Dx 7 – 0. User software can make use of the fractional data to reduce cumulative errors in lengthy calculations or implement rounding algorithms different from that provided by other forms of TBLS. The previously described assumed radix point places two restrictions on the programmer:

1. Tables are limited to 257 entries in length.
2. Interpolation resolution is limited to 1/256, the distance between consecutive table entries. The assumed radix point should not, however, be construed by the programmer as a requirement that the independent variable be calculated as a fractional number in the range $0 < \pi < 255$. On the contrary, X should be considered an integer in the range $0 < \pi < 65535$, realizing that the table is actually a compressed representation of a linearized function in which only every 256th value is actually stored in memory.

TBLS TBLSN — Table Lookup and Interpolate (Signed) (CPU32)

TBLS TBLSN

Condition Codes:

X	N	Z	V	C
—	*	*	*	0

X — Not affected.
N — Set if the most significant bit of the result is set; cleared otherwise.
Z — Set if the result is zero; cleared otherwise.
V — Set if the integer portion of an unrounded long result is not in the range, $-(2^{23}) \le \text{Result} \le (2^{23}) - 1$; cleared otherwise.
C — Always cleared.

Instruction Format:

TABLE LOOKUP AND INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER Dx			1	R	0	1	SIZE		0	0	0	0	0	0

DATA REGISTER INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	REGISTER Dym		
0	REGISTER Dx			1	R	0	1	SIZE		0	0	0	REGISTER Dyn		

TBLS TBLSN — Table Lookup and Interpolate (Signed) (CPU32) — TBLS TBLSN

Instruction Fields:

Effective address field (table lookup and interpolate mode only)—Specifies the destination location. Only control alterable addressing modes are allowed as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	—	—
(An) +	—	—
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011

Size Field—Specifies the size of operation.
00 — Byte Operation
01 — Word Operation
10 — Long Operation

Register field—Specifies the destination data register, Dx. On entry, the register contains the interpolation fraction and entry number.

Dym, Dyn field—If the effective address mode field is nonzero, this operand register is unused and should be zero. If the effective address mode field is zero, the surface interpolation variant of this instruction is implied, and Dyn specifies one of the two source operands.

Rounding mode field—The R-bit controls the rounding of the final result. When R = 0, the result is rounded according to the round-to-nearest algorithm. When R = 1, the result is returned unrounded.

TBLU TBLUN Table Lookup and Interpolation (Unsigned) (CPU32)

TBLU TBLUN

Operation:

Rounded:

ENTRY(n) + {(ENTRY(n + 1) – ENTRY(n)) x Dx 7 – 0} ÷ 256 → Dx

Unrounded:

ENTRY(n) x 256 + {(ENTRY(n + 1) – ENTRY(n)) x Dx 7 – 0} → Dx

Where ENTRY(n) and ENTRY(n + 1) are either:

1. Consecutive entries in the table pointed to by the < ea > and indexed by Dx 15 – 8 π SIZE or;
2. The registers Dym, Dyn respectively

Assembler Syntax:

TBLU. < size > < ea > ,Dx	Result rounded
TBLUN. < size > < ea > ,Dx	Result not rounded
TBLU. < size > Dym:Dyn, Dx	Result rounded
TBLUN. < size > Dym:Dyn, Dx	Result not rounded

Attributes: Size = (Byte, Word, Long)

Description: The TBLU and TBLUN instructions allow the efficient use of piecewise linear, compressed data tables to model complex functions. The TBLU instruction has two modes of operation: table lookup and interpolate mode and data register interpolate mode.

For table lookup and interpolate mode, data register Dx 15 – 0 contains the independent variable X. The effective address points to the start of a unsigned byte, word, or long-word table containing a linearized representation of the dependent variable, Y, as a function of X. In general, the independent variable, located in the low-order word of Dx, consists of an 8-bit integer part and an 8-bit fractional part. An assumed radix point is located between bits 7 and 8. The integer part, Dx 15 – 8, is scaled by the operand size and is used as an offset into the table. The selected entry in the table is subtracted from the next consecutive entry. A fractional portion of this difference is taken by multiplying by the interpolation fraction, Dx 7 – 0. The adjusted difference is then added to the selected table entry. The result is returned in the destination data register, Dx.

TBLU TBLUN Table Lookup and Interpolation (Unsigned) (CPU32) TBLU TBLUN

For register interpolate mode, the interpolation occurs using the Dym and Dyn registers in place of the two table entries. For this mode, only the fractional portion, Dx 7 – 0, is used in the interpolation and the integer portion, Dx 15 – 8, is ignored. The register interpolation mode may be used with several table lookup and interpolations to model multidimensional functions.

Signed table entries range from -2^{n-1} to $2^{n-1}-1$; whereas, unsigned table entries range from 0 to 2^{n-1} where n is 8, 16, or 32 for byte, word, and long-word tables, respectively. The unsigned and unrounded table results will be zero-extended instead of sign-extended.

Rounding of the result is optionally selected via the "R" instruction field. If R = 0 (TABLE), the fractional portion is rounded according to the round-to-nearest algorithm. The rounding procedure can be summarized by the following table:

Adjusted Difference Fraction	Rounding Adjustment
≥ 1/2	+ 1
< 1/2	+ 0

The adjusted difference is then added to the selected table entry. The rounded result is returned in the destination data register, Dx. Only the portion of the register corresponding to the selected size is affected.

	31 24	23 16	15 8	7 0
BYTE	UNAFFECTED	UNAFFECTED	UNAFFECTED	RESULT
WORD	UNAFFECTED	UNAFFECTED	RESULT	RESULT
LONG	RESULT	RESULT	RESULT	RESULT

If R = 1 (TBLUN), the result is returned in register Dx without rounding. If the size is byte, the integer portion of the result is returned in Dx 15 – 8; the integer portion of a word result is stored in Dx 23 – 8; the least significant 24 bits of a long result are stored in Dx 31 – 8. Byte and word results are sign-extended to fill the entire 32-bit register.

TBLU TBLUN Table Lookup and Interpolation (Unsigned) (CPU32) TBLU TBLUN

	31 24	23 16	15 8	7 0
BYTE	SIGN-EXTENDED	SIGN-EXTENDED	RESULT	FRACTION
WORD	SIGN-EXTENDED	RESULT	RESULT	FRACTION
LONG	RESULT	RESULT	RESULT	FRACTION

NOTE

The long-word result contains only the least significant 24 bits of integer precision.

For all sizes, the 8-bit fractional portion of the result is returned in the low byte of the data register, Dx 7 – 0. User software can make use of the fractional data to reduce cumulative errors in lengthy calculations or implement rounding algorithms different from that provided by other forms of TBLU. The previously described assumed radix point places two restrictions on the programmer:

1. Tables are limited to 257 entries in length.
2. Interpolation resolution is limited to 1/256, the distance between consecutive table entries. The assumed radix point should not, however, be construed by the programmer as a requirement that the independent variable be calculated as a fractional number in the range $0 \le X \le 255$. On the contrary, X should be considered to be an integer in the range $0 \le X \le 65535$, realizing that the table is actually a compressed representation of a linearized function in which only every 256th value is actually stored in memory.

Condition Codes:

X	N	Z	V	C
—	*	*	*	0

X — Not affected.
N — Set if the most significant bit of the result is set; cleared otherwise.
Z — Set if the result is zero; cleared otherwise.
V — Set if the integer portion of an unrounded long result is not in the range, $-(2^{23}) \le \text{Result} \le (2^{23}) - 1$; cleared otherwise.
C — Always cleared.

TBLU TBLUN Table Lookup and Interpolation (Unsigned) (CPU32) TBLU TBLUN

Instruction Format:

TABLE LOOKUP AND INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER Dx			0	R	0	1	SIZE	0	0	0	0	0	0	0

DATA REGISTER INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	REGISTER Dym		
0	REGISTER Dx			0	R	0	0	SIZE		0	0	0	REGISTER Dyn		

Instruction Fields:

Effective address field (table lookup and interpolate mode only)—Specifies the destination location. Only control alterable addressing modes are allowed as listed in the following table:

Addressing Mode	Mode	Register
Dn	—	—
An	—	—
(An)	010	reg. number:An
(An) +	—	—
– (An)	100	reg. number:An
(d_{16},An)	101	reg. number:An
(d_8,An,Xn)	110	reg. number:An
(bd,An,Xn)	110	reg. number:An

Addressing Mode	Mode	Register
(xxx).W	111	000
(xxx).L	111	001
# < data >	—	—
(d_{16},PC)	111	010
(d_8,PC,Xn)	111	011
(bd,PC,Xn)	111	011

Size field—Specifies the size of operation.
00 — Byte Operation
01 — Word Operation
10 — Long Operation

TBLU TBLUN Table Lookup and Interpolation (Unsigned) (CPU32) TBLU TBLUN

Register field—Specifies the destination data register, Dx. On entry, the register contains the interpolation fraction and entry number.

Dym, Dyn field—If the effective address mode field is nonzero, this operand register is unused and should be zero. If the effective address mode field is zero, the surface interpolation variant of this instruction is implied, and Dyn specifies one of the two source operands.

Rounding mode field—The R-bit controls the rounding of the final result. When R = 0, the result is rounded according to the round-to-nearest algorithm. When R = 1, the result is returned unrounded.

SECTION 8
INSTRUCTION FORMAT SUMMARY

This section contains a listing of the M68000 family instructions in binary format. It is listed in opcode order for the M68000 family instruction set.

8.1 INSTRUCTION FORMAT

The following paragraphs present a summary of the binary encoding fields.

8.1.1 Coprocessor ID Field

This field specifies which coprocessor in a system is to perform the operation. When using directly supported floating-point instructions for the MC68040, this field must be set to one.

8.1.2 Effective Address Field

This field specifies which addressing mode is to be used. For some operations, there are hardware-enforced restrictions on the available addressing modes allowed.

8.1.3 Register/Memory Field

This field is common to all arithmetic instructions. A zero in this field indicates a register-to-register operation, and a one indicates an < ea > -to-register operation.

8.1.4 Source Specifier Field

This field is common to all artihmetic instructions. The value of the register/memory (R/M) field affects this field,s definition. If R/M = 0, specifies the source floating-point data register (FPDR). If R/M = 1, specifies the source operand data format.

000 — Long-Word Integer (L)
001 — Single-Precision Real (S)
010 — Extended-Precision Real (X)
011 — Packed-Decimal Real (P)
100 — Word Integer (W)
101 — Double-Precision Real (D)
110 — Byte Integer (B)

8.1.5 Destination Register Field

This field is common to all arithmetic instructions. It specifies the FPDR that that will be the destination. The results are always stored in this register.

8.1.6 Conditional Predicate Field

This field is common to all conditional instructions and specifies the conditional test that is to be evaluated. Table 8-1 shows the binary encodings for the conditional tests.

8.1.7 Shift and Rotate Instructions

The following paragraphs define the fields used with the shift and rotate instructions.

8.1.7.1 Count Register Field. If i/r = 0, this field contains the rotate (shift) count of 1 – 8 (a zero specifies 8). If i/r = 1, this field specifies a data register that contains the rotate (shift) count. The following shift and rotate fields are encoded as follows:

dr field
0 — Rotate (shift) Right
1 — Rotate (shift) Left

i/r field
0 — Immediate Rotate (shift) Count
1 — Register Rotate (shift) Count

8.1.7.2 Register Field. This field specifies a data register to be rotated (shifted).

Table 8-1. Conditional Predicate Field Encoding

Conditional Predicate	Mnemonic	Definition
000000	F	False
000001	EQ	Equal
000010	OGT	Ordered Greater Than
000011	OGE	Ordered Greater Than or Equal
000100	OLT	Ordered Less Than
000101	OLE	Ordered Less Than or Equal
000110	OGL	Ordered Greater Than or Less Than
000111	OR	Ordered
001000	UN	Unordered
001001	UEQ	Unordered or Equal
001010	UGT	Unordered or Greater Than
001011	UGE	Unordered or Greater Than or Equal
001100	ULT	Unordered or Less Than
001101	ULE	Unordered or Less Than or Equal
001110	NE	Not Equal
001111	T	True
010000	SF	Signaling False
010001	SEQ	Signaling Equal
010010	GT	Greater Than
010011	GE	Greater Than or Equal
010100	LT	Less Than
010101	LE	Less Than or Equal
010110	GL	Greater Than or Less Than
010111	GLE	Greater Than or Less Than or Equal
011000	NGLE	Not (Greater Than or Less Than or Equal)
011001	NGL	Not (Greater Than or Less Than)
011010	NLE	Not (Less Than or Equal)
011011	NLT	Not (Less Than)
011100	NGE	Not (Greater Than or Equal)
011101	NGT	Not (Greater Than)
011110	SNE	Signaling Not Equal
011111	ST	Signaling True

8.1.8 Size Field

This field specifies the size of the operation. The encoding is as follows:

00 — Byte Operation
01 — Word Operation
10 — Long Operation

8.1.9 Opmode Field

Refer to the applicable instruction descriptions for the encoding of this field in **Section 4 Integer Instructions**, **Section 5 Floating Point Instructions**, **Section 6 Supervisor (Privaleged) Instructions**, and **Section 7 CPU32 Instructions**.

8.1.10 Address/Data Field

This field specifies the type of general register. The encoding is:

0 — Data Register
1 — Address Register

8.2 OPERATION CODE MAP

Table 8-2 lists the encoding for bits 15 – 12 and the operation performed.

Table 8-2. Operation Code Map

Bits 15 – 12	Operation
0000	Bit Manipulation/MOVEP/Immed iate
0001	Move Byte
0010	Move Long
0011	Move Word
0100	Miscellaneous
0101	ADDQ/SUBQ/Scc/DBcc/TRAPc c
0110	Bcc/BSR/BRA
0111	MOVEQ
1000	OR/DIV/SBCD
1001	SUB/SUBX
1010	(Unassigned, Reserved)
1011	CMP/EOR
1100	AND/MUL/ABCD/EXG
1101	ADD/ADDX
1110	Shift/Rotate/Bit Field
1111	Coprocessor Interface/MC68040 and CPU32 Extensions

ORI to CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	8-BIT BYTE DATA							

ORI to SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
16-BIT WORD DATA															

ORI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

ANDI to CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	8-BIT BYTE DATA							

ANDI to SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0
16-BIT WORD DATA															

ANDI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

SUBI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	SIZE		EFFECTIVE ADDRESS MODE			REGISTER		
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

RTM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	0	0	D/A	REGISTER		

CALLM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	0	0	0	0	0	0	ARGUMENT COUNT							

ADDI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	SIZE		EFFECTIVE ADDRESS MODE			REGISTER		
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

CMP2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	SIZE		0	1	1	EFFECTIVE ADDRESS MODE			REGISTER		
D/A	REGISTER			0	0	0	0	0	0	0	0	0	0	0	0

CHK2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	SIZE		0	1	1	EFFECTIVE ADDRESS MODE			REGISTER		
D/A	REGISTER			1	0	0	0	0	0	0	0	0	0	0	0

EORI to CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	BYTE DATA (8 BITS)							

EORI to SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	0
16-BIT WORD DATA															

EORI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		
16-BIT WORD DATA								16-BIT BYTE DATA							
32-BIT LONG DATA															

CMPI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		
16-BIT WORD DATA								8-BIT BYTE DATA							
32-BIT LONG DATA															

BTST

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	0	BIT NUMBER							

BCHG

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	0	BIT NUMBER							

BCLR

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	0	BIT NUMBER							

BSET

BIT NUMBER STATIC, SPECIFIED AS IMMEDIATE DATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	BIT NUMBER								

MOVES

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		
A/D	REGISTER			dr	0	0	0	0	0	0	0	0	0	0	0

CAS2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	SIZE		0	1	1	1	1	1	1	0	0
D/A1	Rn1			0	0	0	Du1			0	0	0	Dc1		
D/A2	Rn2			0	0	0	Du2			0	0	0	Dc2		

CAS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	SIZE		0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	Du			0	0	0	Dc		

BTST

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	REGISTER			1	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

BCHG

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	REGISTER			1	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

BCLR

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	REGISTER			1	1	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

BSET

BIT NUMBER DYNAMIC, SPECIFIED IN A REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	REGISTER			1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

MOVEP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	DATA REGISTER			OPMODE			0	0	1	ADDRESS REGISTER		
16-BIT DISPLACEMENT															

MOVEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	SIZE		DESTINATION REGISTER			0	0	1	SOURCE					
										MODE			REGISTER		

MOVE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	SIZE		DESTINATION						SOURCE					
				REGISTER			MODE			MODE			REGISTER		

MOVE from SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	1	1	SOURCE					
										MODE			REGISTER		

MOVE from CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

NEGX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	SIZE1		EFFECTIVE ADDRESS					
										MODE			REGISTER		

CLR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		

MOVE to CCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

NEG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		

NOT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		

MOVE to SR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

EXT, EXTB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	OPMODE			0	0	0	REGISTER		

LINK

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	0	0	0	1	REGISTER		
HIGH-ORDER DISPLACEMENT															
LOW-ORDER DISPLACEMENT															

NBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

SWAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	0	REGISTER		

BKPT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	0	0	1	VECTOR		

PEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

BGND

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	1	1	1	0	1	0

ILLEGAL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0

TAS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

TST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		

MULU

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER DI			0	SIZE	0	0	0	0	0	0	0	REGISTER Dh		

MULS

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER DI			1	SIZE	0	0	0	0	0	0	0	REGISTER Dh		

DIVU, DIVUL

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER Dq			0	SIZE	0	0	0	0	0	0	0	REGISTER Dr		

DIVS, DIVSL

LONG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER Dq			1	SIZE	0	0	0	0	0	0	0	REGISTER Dr		

TRAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	0	VECTOR			

LINK

WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	REGISTER		
WORD DISPLACEMENT															

UNLK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	REGISTER		

MOVE USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	dr	REGISTER		

RESET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1

STOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0
IMMEDIATE DATA															

RTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1

RTD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	0
16-BIT DISPLACEMENT															

RTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

TRAPV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	0

RTR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	1

MOVEC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	dr
A/D	REGISTER			CONTROL REGISTER											

JSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

JMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

MOVEM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	dr	0	0	1	SIZE	EFFECTIVE ADDRESS					
										MODE			REGISTER		
REGISTER LIST MASK															

LEA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	REGISTER			1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

CHK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	REGISTER			SIZE		0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

ADDQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	DATA			0	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		

SUBQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	DATA			1	SIZE		EFFECTIVE ADDRESS					
										MODE			REGISTER		

DBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	CONDITION				1	1	0	0	1	REGISTER		
16-BIT DISPLACEMENT															

TRAPcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	CONDITION				1	1	1	1	1	OPMODE		
OPTIONAL WORD															
OR LONG WORD															

Scc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	CONDITION				1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

BRA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	8-BIT DISPLACEMENT							
16-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $00															
32-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $FF															

BSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	8-BIT DISPLACEMENT							
16-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $00															
32-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $FF															

Bcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	CONDITION				8-BIT DISPLACEMENT							
16-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $00															
32-BIT DISPLACEMENT IF 8-BIT DISPLACEMENT = $FF															

MOVEQ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	REGISTER			0	DATA							

DIVU, DIVUL

WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER			0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

SBCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER Dy/Ay			1	0	0	0	0	R/M	REGISTER Dx/Ax		

PACK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER Dy/Ay			1	0	1	0	0	R/M	REGISTER Dx/Ax		
16-BIT EXTENSION: ADJUSTMENT															

UNPK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER Dy/Ay			1	1	0	0	0	R/M	REGISTER Dx/Ax		
16-BIT EXTENSION: ADJUSTMENT															

DIVS, DIVSL

WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER			1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

OR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

SUBX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	REGISTER Dy/Ay			1	SIZE		0	0	R/M	REGISTER Dx/Ax		

SUB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

SUBA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

CMPM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER Ax			1	SIZE		0	0	1	REGISTER Ay		

CMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

CMPA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

EOR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

MULU

WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	REGISTER			0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

ABCD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	REGISTER Rx			1	0	0	0	0	R/M	REGISTER Ry		

MULS

WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	REGISTER			1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

EXG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	REGISTER Rx			1	OPMODE					REGISTER Ry		

AND

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

ADDX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	REGISTER Rx			1	SIZE		0	0	R/M	REGISTER Ry		

ADDA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

ADD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE			REGISTER		

ASL, ASR

MEMORY SHIFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	0	dr	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

LSL, LSR

MEMORY SHIFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	dr	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

ROXL, ROXR

MEMORY ROTATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	dr	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

ROL, ROR

MEMORY ROTATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	dr	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

BFTST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	Do	OFFSET					Dw	WIDTH				

BFEXTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER			Do	OFFSET					Dw	WIDTH				

BFCHG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	Do	OFFSET					Dw	WIDTH				

BFEXTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER			Do	OFFSET					Dw	WIDTH				

BFCLR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	Do	OFFSET					Dw	WIDTH				

BFFFO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER			Do	OFFSET					Dw	WIDTH				

BFSET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	Do	OFFSET					Dw	WIDTH				

BFINS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	1	1	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	REGISTER			Do	OFFSET					Dw	WIDTH				

ASL, ASR

REGISTER SHIFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	COUNT/ REGISTER			dr	SIZE		i/r	0	0	REGISTER		

LSL, LSR

REGISTER SHIFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	COUNT/ REGISTER			dr	SIZE		i/r	0	1	REGISTER		

ROXL, ROXR

REGISTER ROTATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	COUNT/ REGISTER			dr	SIZE		i/r	1	0	REGISTER		

ROL, ROR

REGISTER ROTATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	COUNT/ REGISTER			dr	SIZE		i/r	1	1	REGISTER		

PMOVE

MC68EC030, ACX REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	P REGISTER			R/W	0	0	0	0	0	0	0	0	0

PMOVE

MC68030 ONLY, TT REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	P REGISTER			R/W	FD	0	0	0	0	0	0	0	0

PLOAD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	0	0	0	R/W	0	0	0	0	FC				

PVALID

VAL CONTAINS ACCESS LEVEL TO TEST AGAINST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

PVALID

MAIN PROCESSOR REGISTER CONTAINS ACCESS LEVEL TO TEST AGAINST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	0	1	0	0	0	0	0	0	0	0	REGISTER		

PFLUSH

MC68030 ONLY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	1	MODE			0	0	MASK			FC				

PFLUSH
PFLUSHA
PFLUSHS

MC68851

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	1	MODE			0	MASK			FC					

PMOVE

MC68851, TO/FROM TC, CRP, DRP, SRP, CAL, VAL, SCC, AND AC REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	1	0	P REGISTER			R/W	0	0	0	0	0	0	0	0	0

PMOVE

MC68030 ONLY, SRP, CRP, AND TC REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	1	0	P REGISTER			R/W	FD	0	0	0	0	0	0	0	0

PMOVE

MC68030 ONLY, MMUSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	1	1	0	0	0	R/W	0	0	0	0	0	0	0	0	0

PMOVE

MC68EC030, ACUSR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	1	1	0	0	0	R/W	0	0	0	0	0	0	0	0	0

PMOVE

MC68851, TO/FROM PSR AND PCSR REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	1	P REGISTER			R/W	0	0	0	0	0	0	0	0	0

PMOVE

MC68851, TO/FROM BADx AND BACx REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	1	1	P REGISTER			R/W	0	0	0	0	NUM			0	0

PTEST

MC68EC030

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	0	0	0	0	R/W	0	REGISTER		FC					

PTEST

MC68030 ONLY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	0	LEVEL			R/W	A	REGISTER		FC					

PTEST

MC68851

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	0	LEVEL			R/W	A REGISTER			FC					

PFLUSHR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

PScc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	0	0	0	0	0	0	0	0	0	MC68851 CONDITION					

PDBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	0	0	1	COUNT REGISTER		
0	0	0	0	0	0	0	0	0	0	MC68851 CONDITION					
16-BIT DISPLACEMENT															

PTRAPcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	1	1	1	OPMODE		
0	0	0	0	0	0	0	0	0	0	MC68851 CONDITION					
16-BIT OPERAND OR MOST SIGNIFICANT WORD OF 32-BIT OPERAND (IF NEEDED)															
LEAST SIGNIFICANT WORD OF 32-BIT OPERAND (IF NEEDED)															

PBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1	SIZE	MC68851 CONDITION					
16-BIT DISPLACEMENT OR MOST SIGNIFICANT WORD OF 32-BIT DISPLACEMENT															
LEAST SIGNIFICANT WORD OF 32-BIT DISPLACEMENT (IF NEEDED)															

PSAVE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	1	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		

PRESTORE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	1	0	1	EFFECTIVE ADDRESS					
										MODE			REGISTER		

PFLUSH

MC68EC040, POSTINCREMENT SOURCE AND DESTINATION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	0	0	OPMODE		REGISTER		

PFLUSH

MC68040/MC68LC040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	0	0	OPMODE		REGISTER		

PTEST

MC68040/MC68LC040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	1	R/W	0	1	REGISTER		

PTEST

MC68EC040

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	1	0	1	R/W	0	1	REGISTER		

CINV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	0	CACHE		0	SCOPE		REGISTER		

CPUSH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	0	CACHE		1	SCOPE		REGISTER		

MOVE16

ABSOLUTE LONG ADDRESS SOURCE OR DESTINATION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0	0	0	0	OPMODE		REGISTER Ay		
HIGH-ORDER ADDRESS															
LOW-ORDER ADDRESS															

MOVE16

POSTINCREMENT SOURCE AND DESTINATION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	1	0	0	0	1	0	0	REGISTER Ax		
1	REGISTER Ay			0	0	0	0	0	0	0	0	0	0	0	0

TBLU, TBLUN

TABLE LOOKUP AND INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	EFFECTIVE ADDRESS					
											MODE		REGISTER		
0	REGISTER Dx			0	R	0	1	SIZE	0	0	0	0	0	0	0

TBLS, TBLSN

TABLE LOOKUP AND INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	EFFECTIVE ADDRESS					
											MODE		REGISTER		
0	REGISTER Dx			1	R	0	1	SIZE		0	0	0	0	0	0

TBLU, TBLUN

DATA REGISTER INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	REGISTER Dym		
0	REGISTER Dx			0	R	0	0	SIZE		0	0	0	REGISTER Dyn		

TBLS, TBLSN

DATA REGISTER INTERPOLATE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	REGISTER Dym		
0	REGISTER Dx			1	R	0	0	SIZE		0	0	0	REGISTER Dyn		

LPSTOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
IMMEDIATE DATA															

FMOVECR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	DESTINATION REGISTER		ROM OFFSET							

FINT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	0	0	0	1

FSINH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	0	0	1	0

FINTRZ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	0	0	1	1

FLOGNP1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	0	1	1	0

FETOXM1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	1	0	0	0

FTANH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	1	0	0	1

FATAN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	1	0	1	0

FASIN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			EFFECTIVE ADDRESS REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	0	1	1	0	0

FATANH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	0	1

FSIN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	1	0

FTAN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	0	1	1	1	1

FETOX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	0	0	0	0

FTWOTOX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	0	0	0	1

FTENTOX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	0	0	1	0

FLOGN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	0	1	0	0

FLOG10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	0	1	0	1

FLOG2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	0	1	1	0

FCOSH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	1	0	0	1

FACOS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	1	1	0	0

FCOS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	0	1	1	1	0	1

FGETEXP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	1	1	0

FGETMAN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	0	1	1	1	1

FMOD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	0	1

FSGLDIV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	0

FREM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	1

FSCALE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS					
										MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER				DESTINATION REGISTER			0	1	0	0	1	0

FSGLMUL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	1	0	0	1	1	1

FSINCOS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	1	1	0	DESTINATION REGISTER, FPc		

FCMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	1	1	1	0	0	0

FTST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			0	1	1	1	0	1	0

FABS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FADD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FDIV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FMOVE

DATA REGISTER, EFFECTIVE ADDRESS TO REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FMUL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FNEG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FSQRT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FSUB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	R/M	0	SOURCE SPECIFIER			DESTINATION REGISTER			OPMODE						

FMOVE

DATA REGISTER, REGISTER TO MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
0	1	1	SOURCE SPECIFIER			DESTINATION REGISTER				K-FACTOR (IF REQUIRED)					

FMOVE

SYSTEM CONTROL REGISTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
1	0	dr	REGISTER SELECT			0	0	0	0	0	0	0	0	0	0

FMOVEM

CONTROL REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
1	0	dr	REGISTER SELECT			0	0	0	0	0	0	0	0	0	0

FMOVEM

DATA REGISTERS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
1	1	dr	MODE		0	0	0	REGISTER LIST							

cpGEN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	0	EFFECTIVE ADDRESS MODE			REGISTER		
COPROCESSOR ID-DEPENDENT COMMAND WORD															
OPTIONAL EFFECTIVE ADDRESS OR COPROCESSOR ID-DEFINED EXTENSION WORDS															

FScc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	0	0	0	0	0	0	0	0	CONDITIONAL PREDICATE					

cpScc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	EFFECTIVE ADDRESS MODE			REGISTER		
0	0	0	0	0	0	0	0	0	0	COPROCESSOR ID CONDITION					
OPTIONAL EFFECTIVE ADDRESS OR COPROCESSOR ID-DEFINED EXTENSION WORDS															

FBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	1	SIZE	CONDITIONAL PREDICATE					
16-BIT DISPLACEMENT OR MOST SIGNIFICANT WORD OF 32-BIT DISPLACEMENT															
LEAST SIGNIFICANT WORD OF 32-BIT DISPLACEMENT (IF NEEDED)															

cpBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	1	SIZE	COPROCESSOR ID CONDITION					
OPTIONAL COPROCESSOR ID-DEFINED EXTENSION WORDS															
WORD OR															
LONG-WORD DISPLACEMENT															

cpSAVE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			1	0	0	EFFECTIVE ADDRESS MODE			REGISTER		

FSAVE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			1	0	0	EFFECTIVE ADDRESS MODE			REGISTER		

cpRESTORE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			1	0	1	EFFECTIVE ADDRESS MODE			REGISTER		

FRESTORE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			1	0	1	EFFECTIVE ADDRESS MODE			REGISTER		

FDBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	0	0	1	COUNT REGISTER		
0	0	0	0	0	0	0	0	0	0	CONDITIONAL PREDICATE					
16-BIT DISPLACEMENT															

cpDBcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	0	0	1	REGISTER		
0	0	0	0	0	0	0	0	0	0	COPROCESSOR ID CONDITION					
OPTIONAL COPROCESSOR ID-DEFINED EXTENSION WORDS															
16-BIT DISPLACEMENT															

FTRAPcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	1	1	1	MODE		
0	0	0	0	0	0	0	0	0	0	CONDITIONAL PREDICATE					
16-BIT OPERAND OR MOST SIGNIFICANT WORD OF 32-BIT OPERAND (IF NEEDED)															
LEAST SIGNIFICANT WORD OR 32-BIT OPERAND (IF NEEDED)															

cpTRAPcc

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	0	1	1	1	1	OPMODE		
0	0	0	0	0	0	0	0	0	0	COPROCESSOR ID CONDITION					
OPTIONAL COPROCESSOR ID-DEFINED EXTENSION WORDS															
OPTIONAL WORD															
OR LONG-WORD OPERAND															

FNOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	COPROCESSOR ID			0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX A
PROCESSOR INSTRUCTION SUMMARY

This appendix provides a quick reference of the M68000 family instructions. The organization of this section is by processors and their addressing modes. All references to the MC68000, MC68020, and MC68030 include references to the corresponding embedded controllers, MC68EC000, MC68EC020, and MC68EC030. All references to the MC68040 include the MC68LC040 and MC68EC040. This referencing applies throughout this section unless otherwise specified. Table A-1 lists the M68000 family instructions by mnemonic and indicates which processors they apply to.

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
ABCD	X	X	X	X	X	X			X
ADD	X	X	X	X	X	X			X
ADDA	X	X	X	X	X	X			X
ADDI	X	X	X	X	X	X			X
ADDQ	X	X	X	X	X	X			X
ADDX	X	X	X	X	X	X			X
AND	X	X	X	X	X	X			X
ANDI	X	X	X	X	X	X			X
ANDI to CCR	X	X	X	X	X	X			X
ANDI to SR[1]	X	X	X	X	X	X			X
ASL, ASR	X	X	X	X	X	X			X
Bcc	X	X	X	X	X	X			X
BCHG	X	X	X	X	X	X			X
BCLR	X	X	X	X	X	X			X
BFCHG				X	X	X			
BFCLR				X	X	X			
BFEXTS				X	X	X			
BFEXTU				X	X	X			
BFFFO				X	X	X			

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference (Continued)

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
BFINS				X	X	X			
BFSET				X	X	X			
BFTST				X	X	X			
BGND									X
BKPT			X	X	X	X			X
BRA	X	X	X	X	X	X			X
BSET	X	X	X	X	X	X			X
BSR	X	X	X	X	X	X			X
BTST	X	X	X	X	X	X			X
CALLM				X					
CAS, CAS2				X	X	X			
CHK	X	X	X	X	X	X			X
CHK2				X	X	X			X
CINV[1]						X			
CLR	X	X	X	X	X	X			X
CMP	X	X	X	X	X	X			X
CMPA	X	X	X	X	X	X			X
CMPI	X	X	X	X	X	X			X
CMPM	X	X	X	X	X	X			X
CMP2				X	X	X			X
cpBcc				X	X				
cpDBcc				X	X				
cpGEN				X	X				
cpRESTORE[1]				X	X				
cpSAVE[1]				X	X				
cpScc				X	X				
cpTRAPcc				X	X				
CPUSH[1]						X			
DBcc	X	X	X	X	X	X			X
DIVS	X	X	X	X	X	X			X
DIVSL				X	X	X			X
DIVU	X	X	X	X	X	X			X
DIVUL				X	X	X			X

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference (Continued)

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
EOR	X	X	X	X	X	X			X
EORI	X	X	X	X	X	X			X
EORI to CCR	X	X	X	X	X	X			X
EORI to SR[1]	X	X	X	X	X	X			X
EXG	X	X	X	X	X	X			X
EXT	X	X	X	X	X	X			X
EXTB				X	X	X			X
FABS						X[2]	X		
FSABS, FDABS						X[2]			
FACOS						[2,3]	X		
FADD						X[2]	X		
FSADD, FDADD						X[2]			
FASIN						[2,3]	X		
FATAN						[2,3]	X		
FATANH						[2,3]	X		
FBcc						X[2]	X		
FCMP						X[2]	X		
FCOS						[2,3]	X		
FCOSH						[2,3]	X		
FDBcc						X[2]	X		
FDIV						X[2]	X		
FSDIV, FDDIV						X[2]			
FETOX						[2,3]	X		
FETOXM1						[2,3]	X		
FGETEXP						[2,3]	X		
FGETMAN						[2,3]	X		
FINT						[2,3]	X		
FINTRZ						[2,3]	X		
FLOG10						[2,3]	X		
FLOG2						[2,3]	X		
FLOGN						[2,3]	X		

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference (Continued)

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
FLOGNP1						2,3			
FMOD						2,3	X		
FMOVE						X[2]	X		
FSMOVE, FDMOVE						X[2]			
FMOVECR						2,3	X		
FMOVEM						X[2]	X		
FMUL						X[2]	X		
FSMUL, FDMUL						X[2]			
FNEG						X[2]	X		
FSNEG, FDNEG						X[2]			
FNOP						X[2]	X		
FREM						2,3	X		
FRESTORE[1]						X[2]	X		
FSAVE*						X[2]	X		
FSCALE						2,3	X		
FScc						X[2]	X		
FSGLDIV						2,3	X		
FSGLMUL						2,3	X		
FSIN						2,3	X		
FSINCOS						2,3	X		
FSINH						2,3	X		
FSQRT						X[2]	X		
FSSQRT, FDSQRT						X[2]			
FSUB						X[2]	X		
FSSUB, FDSUB						X[2]			
FTAN						2,3	X		
FTANH						2,3	X		
FTENTOX						2,3	X		
FTRAPcc						X[2]	X		
FTST						X[2]	X		

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference (Continued)

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
FTWOTOX						2,3	X		
ILLEGAL	X	X	X	X	X	X			X
JMP	X	X	X	X	X	X			X
JSR	X	X	X	X	X	X			X
LEA	X	X	X	X	X	X			X
LINK	X	X	X	X	X	X			X
LPSTOP									X
LSL,LSR	X	X	X	X	X	X			X
MOVE	X	X	X	X	X	X			X
MOVEA	X	X	X	X	X	X			X
MOVE from CCR			X	X	X	X			X
MOVE to CCR	X	X	X	X	X	X			X
MOVE from SR[1]	4	4	X	X	X	X			X
MOVE to SR[1]	X	X	X	X	X	X			X
MOVE USP[1]	X	X	X	X	X	X			X
MOVE16						X			
MOVEC[1]			X	X	X	X			X
MOVEM	X	X	X	X	X	X			X
MOVEP	X	X	X	X	X	X			X
MOVEQ	X	X	X	X	X	X			X
MOVES[1]			X	X	X	X			X
MULS	X	X	X	X	X	X			X
MULU	X	X	X	X	X	X			X
NBCD	X	X	X	X	X	X			X
NEG	X	X	X	X	X	X			X
NEGX	X	X	X	X	X	X			X
NOP	X	X	X	X	X	X			X
NOT	X	X	X	X	X	X			X
OR	X	X	X	X	X	X			X

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference (Continued)

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
ORI	X	X	X	X	X	X			X
ORI to CCR	X	X	X	X	X	X			X
ORI to SR[1]	X	X	X	X	X	X			X
PACK				X	X	X			
PBcc[1]								X	
PDBcc[1]								X	
PEA	X	X	X	X	X	X			X
PFLUSH[1]					X[5]	X		X	
PFLUSHA[1]					X[5]			X	
PFLUSHR[1]								X	
PFLUSHS[1]								X	
PLOAD[1]					X[5]			X	
PMOVE[1]					X			X	
PRESTORE[1]								X	
PSAVE[1]								X	
PScc[1]								X	
PTEST[1]					X	X		X	
PTRAPcc[1]								X	
PVALID								X	
RESET[1]	X	X	X	X	X	X			X
ROL,ROR	X	X	X	X	X	X			X
ROXL, ROXR	X	X	X	X	X	X			X
RTD			X	X	X	X			X
RTE[1]	X	X	X	X	X	X			X
RTM				X					
RTR	X	X	X	X	X	X			X
RTS	X	X	X	X	X	X			X
SBCD	X	X	X	X	X	X			X
Scc	X	X	X	X	X	X			X
STOP[1]	X	X	X	X	X	X			X
SUB	X	X	X	X	X	X			X
SUBA	X	X	X	X	X	X			X
SUBI	X	X	X	X	X	X			X
SUBQ	X	X	X	X	X	X			X
SUBX	X	X	X	X	X	X			X

Table A-1. M68000 Family Instruction Set And Processor Cross-Reference (Concluded)

Mnemonic	68000	68008	68010	68020	68030	68040	68881/ 68882	68851	CPU32
SWAP	X	X	X	X	X	X			X
TAS	X	X	X	X	X	X			X
TBLS, TBLSN									X
TBLU, TBLUN									X
TRAP	X	X	X	X	X	X			X
TRAPcc				X	X	X			X
TRAPV	X	X	X	X	X	X			X
TST	X	X	X	X	X	X			X
UNLK	X	X	X	X	X	X			X
UNPK				X	X	X			

NOTES:
1. Privileged (Supervisor) Instruction.
2. Not applicable to MC68EC040 and MC68LC040
3. These instructions are software supported on the MC68040.
4. This instruction is not privileged for the MC68000 and MC68008.
5. Not applicable to MC68EC030.

Table A-2 lists the M68000 family instructions by mnemonics, followed by the descriptive name.

Table A-2. M68000 Family Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
BGND	Enter Background Mode
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CALLM	CALL Module
CAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bounds
CINV	Invalidate Cache Entries
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
CMP2	Compare Register Against Upper and Lower Bounds
cpBcc	Branch on Coprocessor Condition
cpDBcc	Test Coprocessor Condition Decrement and Branch
cpGEN	Coprocessor General Function
cpRESTORE	Coprocessor Restore Function

Table A-2. M68000 Family Instruction Set (Continued)

Mnemonic	Description
cpSAVE	Coprocessor Save Function
cpScc	Set on Coprocessor Condition
cpTRAPcc	Trap on Coprocessor Condition
CPUSH	Push then Invalidate Cache Entries
DBcc	Test Condition, Decrement and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT, EXTB	Sign Extend
FABS	Floating-Point Absolute Value
FSFABS, FDFABS	Floating-Point Absolute Value (Single/Double Precision)
FACOS	Floating-Point Arc Cosine
FADD	Floating-Point Add
FSADD, FDADD	Floating-Point Add (Single/Double Precision)
FASIN	Floating-Point Arc Sine
FATAN	Floating-Point Arc Tangent
FATANH	Floating-Point Hyperbolic Arc Tangent
FBcc	Floating-Point Branch
FCMP	Floating-Point Compare
FCOS	Floating-Point Cosine
FCOSH	Floating-Point Hyperbolic Cosine
FDBcc	Floating-Point Decrement and Branch
FDIV	Floating-Point Divide
FSDIV, FDDIV	Floating-Point Divide (Single/Double Precision)
FETOX	Floating-Point ex
FETOXM1	Floating-Point ex - 1
FGETEXP	Floating-Point Get Exponent
FGETMAN	Floating-Point Get Mantissa
FINT	Floating-Point Integer Part
FINTRZ	Floating-Point Integer Part, Round-to-Zero
FLOG10	Floating-Point Log10
FLOG2	Floating-Point Log2
FLOGN	Floating-Point Loge
FLOGNP1	Floating-Point Loge (x + 1)
FMOD	Floating-Point Modulo Remainder
FMOVE	Move Floating-Point Register
FSMOVE,FDMOVE	Move Floating-Point Register (Single/Double Precision)
FMOVECR	Move Constant ROM
FMOVEM	Move Multiple Floating-Point Registers
FMUL	Floating-Point Multiply
FSMUL,FDMUL	Floating-Point Multiply (Single/Double Precision)
FNEG	Floating-Point Negate
FSNEG,FDNEG	Floating-Point Negate (Single/Double Precision)
FNOP	Floating-Point No Operation

Table A-2. M68000 Family Instruction Set (Continued)

Mnemonic	Description
FREM	IEEE Remainder
FRESTORE	Restore Floating-Point Internal State
FSAVE	Save Floating-Point Internal State
FSCALE	Floating-Point Scale Exponent
FScc	Floating-Point Set According to Condition
FSGLDIV	Single-Precision Divide
FSGLMUL	Single-Precision Multiply
FSIN	Sine
FSINCOS	Simultaneous Sine and Cosine
FSINH	Hyperbolic Sine
FSQRT	Floating-Point Square Root
FSSQRT,FDSQRT	Floating-Point Square Root (Single/Double Precision)
FSUB	Floating-Point Subtract
FSSUB,FDSUB	Floating-Point Subtract (Single/Double Precision)
FTAN	Tangent
FTANH	Hyperbolic Tangent
FTENTOX	Floating-Point 10x
FTRAPcc	Floating-Point Trap On Condition
FTST	Floating-Point Test
FTWOTOX	Floating-Point 2x
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LPSTOP	Low-Power Stop
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE from CCR	Move from Condition Code Register
MOVE from SR	Move from Status Register
MOVE to CCR	Move to Condition Code Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVE16	16-Byte Block Move
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement

Table A-2. M68000 Family Instruction Set (Concluded)

Mnemonic	Description
OR	Logical Inclusive-OR
ORI	Logical Inclusive-OR Immediate
ORI to CCR	Inclusive-OR Immediate to Condition Code Register
ORI to SR	Inclusive-OR Immediate to Status Register
PACK	Pack BCD
PBcc	Branch on PMMU Condition
PDBcc	Test, Decrement, and Branch on PMMU Condition
PEA	Push Effective Address
PFLUSH	Flush Entry(ies) in the ATCs
PFLUSHA	Flush Entry(ies) in the ATCs
PFLUSHR	Flush Entry(ies) in the ATCs and RPT Entries
PFLUSHS	Flush Entry(ies) in the ATCs
PLOAD	Load an Entry into the ATC
PMOVE	Move PMMU Register
PRESTORE	PMMU Restore Function
PSAVE	PMMU Save Function
PScc	Set on PMMU Condition
PTEST	Test a Logical Address
PTRAPcc	Trap on PMMU Condition
PVALID	Validate a Pointer
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTM	Return from Module
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TBLS, TBLSN	Signed Table Lookup with Interpolate
TBLU, TBLUN	Unsigned Table Lookup with Interpolate
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink
UNPK	Unpack BCD

A.1 MC68000, MC68008, MC68010 PROCESSORS

The following paragraphs provide information on the MC68000, MC68008, and MC68010 instruction set and addressing modes.

A.1.1 M68000, MC68008, and MC68010 Instruction Set

Table A-3 lists the instructions used with the MC68000 and MC68008 processors, and Table A-4 lists the instructions used with MC68010.

Table A-3. MC68000 and MC68008 Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CHK	Check Register Against Bound
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
DBcc	Test Condition, Decrement, and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine

Table A-3. MC68000 and MC68008 Instruction Set (Continued)

Mnemonic	Description
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE to CCR	Move to Condition Code Register
MOVE from SR	Move from Status Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Logical Inclusive-OR
ORI	Logical Inclusive-OR Immediate
ORI to CCR	Inclusive-OR Immediate to Condition Code Register
ORI to SR	Inclusive-OR Immediate to Status Register
PEA	Push Effective Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink

Table A-4. MC68010 Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CHK	Check Register Against Bound
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
DBcc	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine

Table A-4. MC68010 Instruction Set (Continued)

Mnemonic	Description
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE from CCR	Move from Condition Code Register
MOVE from SR	Move from Status Register
MOVE to CCR	Move to Condition Code Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Logical Inclusive-OR
ORI	Logical Inclusive-OR Immediate
ORI to CCR	Inclusive-OR Immediate to Condition Code Register
ORI to SR	Inclusive-OR Immediate to Status Register
PEA	Push Effective Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink

A.1.2 MC68000, MC68008, and MC68010 Addressing Modes

The MC68000, MC68008, and MC68010 support 14 addressing modes as shown in Table A-5.

Table A-5. MC68000, MC68008, and MC68010 Data Addressing Modes

Mode	Generation
Register Direct Addressing	
Data Register Direct	<ea> = Dn
Address Register Direct	<ea> = An
Absolute Data Addressing	
Absolute Short	<ea> = (Next Word)
Absolute Long	<ea> = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	<ea> = (PC) + d_{16}
Relative with Index and Offset	<ea> = (PC) + d_8
Register Indirect Addressing	
Register Indirect	<ea> = (An)
Postincrement Register Indirect	<ea> = (An), An –An + N
Predecrement Register Indirect	An –An–N, <ea> = (An)
Register Indirect with Offset	<ea> = (An) + d_{16}
Indexed Register Indirect with Offset	<ea> = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	<ea> = SR, USP, SSP, PC, VBR, SFC, DFC
Implied Register	

N = 1 for byte, 2 for word, and 4 for long word. If An is the stack pointer and the operand size is byte, N = 2 to keep the stack pointer on a word boundary.

A.2 MC68020 PROCESSORS

The following paragraphs provide information on the MC68020 instruction set and addressing modes.

A.2.1 MC68020 Instruction Set

Table A-6 lists the instructions used with the MC68020 processors.

Table A-6. MC68020 Instruction Set Summary

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CALLM	CALL Module
CAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bounds
CLR	Clear
CMP	Compare
CMP2	Compare Register Against Upper and Lower Bounds
CMPA	Compare Address
CMPI	Compare Immediate

Table A-6. MC68020 Instruction Set Summary (Continued)

Mnemonic	Description
CMPM	Compare Memory to Memory
cpBcc	Branch to Coprocessor Condition
cpDBcc	Test Coprocessor Condition, Decrement and Branch
cpGEN	Coprocessor General Function
cpRESTORE	Coprocessor Restore Function
cpSAVE	Coprocessor Save Function
cpScc	Set on Coprocessor Condition
cpTRACPcc	Trap on Coprocessor Condition
DBcc	Test Condition, Decrement, and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT, EXTB	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE from CCR	Move from Condition Code Register
MOVE from SR	Move from Status Register
MOVE to CCR	Move to Condition Code Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement

Table A-6. MC68020 Instruction Set Summary (Concluded)

Mnemonic	Description
OR	Logical Inclusive-OR
ORI	Logical Inclusive-OR Immediate
ORI to CCR	Inclusive-OR Immediate to Condition Code Register
ORI to SR	Inclusive-OR Immediate to Status Register
PACK	Pack BCD
PEA	Push Effective Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTM	Return from Module
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink
UNPK	Unpack BCD

A.2.2 MC68020 Addressing Modes

The MC68020 supports 18 addressing modes as shown in Table A-7.

Table A-7. MC68020 Data Addressing Modes

Addressing Modes	Syntax
Register Direct	
Address Register Direct	Dn
Address Register Direct	An
Register Indirect	
Address Register Indirect	(An)
Address Register Indirect with Postincrement	(An)+
Address Register Indirect with Predecrement	–(An)
Address Register Indirect with Displacement	(d_{16},An)
Register Indirect with Index	
Address Register Indirect with Index (8-Bit Displacement)	(d_8,An,Xn)
Address Register Indirect with Index (Base Displacement)	(bd,An,Xn)
Memory Indirect	
Memory Indirect Postindexed	([bd,An],Xn,od)
Memory Indirect Preindexed	([bd,An,Xn],od)
Program Counter Indirect with Displacement	(d_{16},PC)
Program Counter Indirect with Index	
PC Indirect with Index (8-Bit Displacement)	(d_8,PC,Xn)
PC Indirect with Index (Base Displacement)	(bd,PC,Xn)
Program Counter Memory Indirect	
PC Memory Indirect Postindexed	([bd,PC],Xn,od)
PC Memory Indirect Preindexed	([bd,PC,Xn],od)
Absolute	
Absolute Short	(xxx).W
Absolute Long	(xxx).L
Immediate	#<data>

A.3 MC68030 PROCESSORS

The following paragraphs provide information on the MC68030 instruction set and addressing modes.

A.3.1 MC68030 Instruction Set

Table A-8 lists the instructions used with the MC68030 processors.

Table A-8. MC68030 Instruction Set Summary

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bounds
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory

Table A-8. MC68030 Instruction Set Summary (Continued)

Mnemonic	Description
CMP2	Compare Register Against Upper and Lower Bounds
cpBcc	Branch on Coprocessor Condition
cpDBcc	Test Coprocessor Condition, Decrement and Branch
cpGEN	Coprocessor General Function
cpRESTORE	Coprocessor Restore Function
cpSAVE	Coprocessor Save Function
cpScc	Set on Coprocessor Condition
cpTRAPcc	Trap on Coprocessor Condition
DBcc	Test Condition, Decrement and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT, EXTB	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE from CCR	Move from Condition Code Register
MOVE to CCR	Move to Condition Code Register
MOVE from SR	Move from Status Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement

Table A-8. MC68030 Instruction Set Summary (Concluded)

Mnemonic	Description
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Logical Inclusive-OR
ORI	Logical Inclusive-OR Immediate
ORI to CCR	Inclusive-OR Immediate to Condition Code Register
ORI to SR	Inclusive-OR Immediate to Status Register
PACK	Pack BCD
PEA	Push Effective Address
PFLUSH*	Invalidate Entries in the ATC
PFLUSHA*	Invalidate all Entries in the ATC
PLOAD*	Load an Entry into the ATC
PMOVE	Move PMMU Register
PTEST	Get Information about Logical Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink
UNPK	Unpack BCD

*Not applicable to the MC68EC030

A.3.2 MC68030 Addressing Modes

The MC68030 supports 18 addressing modes as shown in Table A-9.

Table A-9. MC68030 Data Addressing Modes

Addressing Modes	Syntax
Register Direct	
Data Register Direct	Dn
Address Register Direct	An
Register Indirect	
Address Register Indirect	(An)
Address Register Indirect with Postincrement	(An)+
Address Register Indirect with Predecrement	–(An)
Address Register Indirect with Displacement	(d_{16},An)
Register Indirect with Index	
Address Register Indirect with Index (8-Bit Displacement)	(d_8,An,Xn)
Address Register Indirect with Index (Base Displacement)	(bd,An,Xn)
Memory Indirect	
Memory Indirect Postindexed	([bd,An],Xn,od)
Memory Indirect Preindexed	([bd,An,Xn],od)
Program Counter Indirect with Displacement	(d_{16},PC)
Program Counter Indirect with Index	
PC Indirect with Index (8-Bit Displacement)	(d_8,PC,Xn)
PC Indirect with Index (Base Displacement)	(bd,PC,Xn)
Program Counter Memory Indirect	
PC Memory Indirect Postindexed	([bd,PC],Xn,od)
PC Memory Indirect Preindexed	([bd,PC,Xn],od)
Absolute	
Absolute Short	(xxx).W
Absolute Long	(xxx).L
Immediate	#<data>

A.4 MC68040 PROCESSORS

The following paragraphs provide information on the MC68040 instruction set and addressing modes.

A.4.1 MC68040 Instruction Set

Table A-10 lists the instructions used with the MC68040 processor.

Table A-10. MC68040 Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ANDI to CCR	AND Immediate to Condition Code Register
ANDI to SR	AND Immediate to Status Register
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bounds
CINV	Invalidate Cache Entries
CLR	Clear
CMP	Compare
CMPA	Compare Address

Table A-10. MC68040 Instruction Set (Continued)

Mnemonic	Description
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
CMP2	Compare Register Against Upper and Lower Bounds
CPUSH	Push then Invalidate Cache Entries
DBcc	Test Condition, Decrement and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive-OR
EORI	Logical Exclusive-OR Immediate
EORI to CCR	Exclusive-OR Immediate to Condition Code Register
EORI to SR	Exclusive-OR Immediate to Status Register
EXG	Exchange Registers
EXT, EXTB	Sign Extend
FABS[1]	Floating-Point Absolute Value
FSABS, FDABS[1]	Floating-Point Absolute Value (Single/Double Precision)
FACOS[1,2]	Floating-Point Arc Cosine
FADD[1]	Floating-Point Add
FSADD, FDADD[1]	Floating-Point Add (Single/Double Precision)
FASIN[1,2]	Floating-Point Arc Sine
FATAN[1,2]	Floating-Point Arc Tangent
FATANH[1,2]	Floating-Point Hyperbolic Arc Tangent
FBcc[1]	Floating-Point Branch
FCMP[1]	Floating-Point Compare
FCOS[1,2]	Floating-Point Cosine
FCOSH[1,2]	Floating-Point Hyperbolic Cosine
FDBcc[1]	Floating-Point Decrement and Branch
FDIV[1]	Floating-Point Divide
FSDIV, FDDIV[1]	Floating-Point Divide (Single/Double Precision)
FETOX[1,2]	Floating-Point e^x
FETOXM1[1,2]	Floating-Point $e^x - 1$
FGETEXP[1,2]	Floating-Point Get Exponent
FGETMAN[1,2]	Floating-Point Get Mantissa
FINT[1,2]	Floating-Point Integer Part
FINTRZ[1,2]	Floating-Point Integer Part, Round-to-Zero
FLOG10[1,2]	Floating-Point $\log_{10}$
FLOG2[1,2]	Floating-Point $\log_2$
FLOGN[1,2]	Floating-Point $\log_e$
FLOGNP1[1,2]	Floating-Point $\log_e (x + 1)$
FMOD[1,2]	Floating-Point Modulo Remainder
FMOVE[1]	Move Floating-Point Register
FSMOVE, FDMOVE[1]	Move Floating-Point Register (Single/Double Precision)
FMOVECR[1]	Move Constant ROM
FMOVEM[1]	Move Multiple Floating-Point Registers
FMUL[1]	Floating-Point Multiply
FSMUL, FDMUL[1]	Floating-Point Multiply (Single/Double Precision)

Table A-10. MC68040 Instruction Set (Continued)

Mnemonic	Description
FNEG[1]	Floating-Point Negate
FSNEG, FDNEG[1]	Floating-Point Negate (Single/Double Precision)
FNOP[1]	Floating-Point No Operation
FREM[1,2]	IEEE Remainder
FRESTORE[1]	Restore Floating-Point Internal State
FSAVE[1]	Save Floating-Point Internal State
FSCALE[1,2]	Floating-Point Scale Exponent
FScc[1]	Floating-Point Set According to Condition
FSGLDIV[1,2]	Single-Precision Divide
FSGLMUL[1,2]	Single-Precision Multiply
FSIN[1,2]	Sine
FSINCOS[1,2]	Simultaneous Sine and Cosine
FSINH[1,2]	Hyperbolic Sine
FSQRT[1]	Floating-Point Square Root
FSSQRT, FDSQRT[1]	Floating-Point Square Root (Single/Double Precision)
FSUB[1]	Floating-Point Subtract
FSSUB, FDSUB[1]	Floating-Point Subtract (Single/Double Precision)
FTAN[1,2]	Tangent
FTANH[1,2]	Hyperbolic Tangent
FTENTOX[1,2]	Floating-Point 10^x
FTRAPcc[1,2]	Floating-Point Trap On Condition
FTST[1]	Floating-Point Test
FTWOTOX[1,2]	Floating-Point 2^x
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE from CCR	Move from Condition Code Register
MOVE to CCR	Move to Condition Code Register
MOVE from SR	Move from Status Register
MOVE to SR	Move to Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MOVE16	16-Byte Block Move
MULS	Signed Multiply
MULU	Unsigned Multiply

Table A-10. MC68040 Instruction Set (Concluded)

Mnemonic	Description
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Logical Inclusive-OR
ORI	Logical Inclusive-OR Immediate
ORI to CCR	Inclusive-OR Immediate to Condition Code Register
ORI to SR	Inclusive-OR Immediate to Status Register
PACK	Pack BCD
PEA	Push Effective Address
PFLUSH	Flush Entry(ies) in the ATCs
PFLUSHA	Flush all Entry(ies) in the ATCs
PTEST	Test a Logical Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink
UNPK	Unpack BCD

NOTES:
1. Not applicable to the MC68EC040 and MC68LC040.
2. These instructions are software supported.

A.4.2 MC68040 Addressing Modes

The MC68040 supports 18 addressing modes as shown in Table A-11.

Table A-11. MC68040 Data Addressing Modes

Addressing Modes	Syntax
Register Direct Data Register Direct Address Register Direct	 Dn An
Register Indirect Address Register Indirect Address Register Indirect with Postincrement Address Register Indirect with Predecrement Address Register Indirect with Displacement	 (An) (An) + –(An) (d_{16},An)
Register Indirect with Index Address Register Indirect with Index (8-Bit Displacement) Address Register Indirect with Index (Base Displacement)	 (d_8,An,Xn) (bd,An,Xn)
Memory Indirect Memory Indirect Postindexed Memory Indirect Preindexed	 ([bd,An],Xn,od) ([bd,An,Xn],od)
Program Counter Indirect with Displacement	(d_{16},PC)
Program Counter Indirect with Index PC Indirect with Index (8-Bit Displacement) PC Indirect with Index (Base Displacement)	 (d_8,PC,Xn) (bd,PC,Xn)
Program Counter Memory Indirect PC Memory Indirect Postindexed PC Memory Indirect Preindexed	 ([bd,PC],Xn,od) ([bd,PC,Xn],od)
Absolute Absolute Short Absolute Long	 xxx.W xxx.L
Immediate	# < data >

A.5 MC68881/MC68882 COPROCESSORS

The following paragraphs provide information on the MC68881/MC68882 instruction set and addressing modes.

A.5.1 MC68881/MC68882 Instruction Set

Table A-12 lists the instructions used with the MC68881/MC68882 coprocessors.

Table A-12. MC68881/MC68882 Instruction Set

Mnemonic	Description
FABS	Floating-Point Absolute Value
FACOS	Floating-Point Arc Cosine
FADD	Floating-Point Add
FASIN	Floating-Point Arc Sine
FATAN	Floating-Point Arc Tangent
FATANH	Floating-Point Hyperbolic Arc Tangent
FBcc	Floating-Point Branch
FCMP	Floating-Point Compare
FCOS	Floating-Point Cosine
FCOSH	Floating-Point Hyperbolic Cosine
FDBcc	Floating-Point Decrement and Branch
FDIV	Floating-Point Divide
FETOX	Floating-Point ex
FETOXM1	Floating-Point ex - 1
FGETEXP	Floating-Point Get Exponent
FGETMAN	Floating-Point Get Mantissa
FINT	Floating-Point Integer Part
FINTRZ	Floating-Point Integer Part, Round-to-Zero
FLOG10	Floating-Point Log10
FLOG2	Floating-Point Log2
FLOGN	Floating-Point Loge
FLOGNP1	Floating-Point Loge (x + 1)
FMOD	Floating-Point Modulo Remainder
FMOVE	Move Floating-Point Register
FMOVECR	Move Constant ROM
FMOVEM	Move Multiple Floating-Point Registers
FMUL	Floating-Point Multiply
FNEG	Floating-Point Negate
FNOP	Floating-Point No Operation
FREM	IEEE Remainder
FRESTORE	Restore Floating-Point Internal State
FSAVE	Save Floating-Point Internal State
FSCALE	Floating-Point Scale Exponent
FScc	Floating-Point Set According to Condition
FSGLDIV	Single-Precision Divide
FSGLMUL	Single-Precision Multiply
FSIN	Sine
FSINCOS	Simultaneous Sine and Cosine
FSINH	Hyperbolic Sine

Table A-12. MC68881/MC68882 Instruction Set

Mnemonic	Description
FSQRT	Floating-Point Square Root
FSUB	Floating-Point Subtract
FTAN	Tangent
FTANH	Hyperbolic Tangent
FTENTOX	Floating-Point 10x
FTRAPcc	Floating-Point Trap On Condition
FTST	Floating-Point Test
FTWOTOX	Floating-Point 2x

A.5.2 MC68881/MC68882 Addressing Modes

The MC68881/MC68882 does not perform address calculations. When the floating-point coprocessor instructs the processor to transfer an operand via the coprocessor interface, the processor performs the addressing mode calculation requested in the instruction.

A.6 MC68851 COPROCESSORS

The following paragraphs provide information on the MC68851 instruction set and addressing modes.

A.6.1 MC68851 Instruction Set

Table A-13 lists the instructions used with the MC68851 coprocessor.

Table A-13. MC68851 Instruction Set

Mnemonic	Description
PBcc	Branch on PMMU Condition
PDBcc	Test, Decrement, and Branch on PMMU Condition
PFLUSH	Flush Entry(ies) in the ATCs
PFLUSHA	Flush Entry(ies) in the ATCs
PFLUSHR	Flush Entry(ies) in the ATCs and RPT Entries
PFLUSHS	Flush Entry(ies) in the ATCs
PLOAD	Load an Entry into the ATC
PMOVE	Move PMMU Register
PRESTORE	PMMU Restore Function
PSAVE	PMMU Save Function
PScc	Set on PMMU Condition
PTEST	Test a Logical Address
PTRAPcc	Trap on PMMU Condition
PVALID	Validate a Pointer

A.6.2 MC68851 Addressing Modes

The MC68851 supports the same addressing modes as the MC68020 (see Table A-7).

APPENDIX B
EXCEPTION PROCESSING REFERENCE

This appendix provides a quick reference for system programmers who are already familiar with the stack frames. For more detail, please refer to the appropriate userOs manual.

B.1 EXCEPTION VECTOR ASSIGNMENTS FOR THE M68000 FAMILY

Table B-1 lists all vector assignments up to and including the MC68040 and its derivatives. Many of these vector assignments are processor specific. For instance, vector 13, the coprocessor protocol violation vector, only applies to the MC68020, MC68EC020, MC68030, and MC68EC030. Refer to the appropriate user's manual to determine which exception type is applicable to a specific processor.

Table B-1. Exception Vector Assignments for the M68000 Family

Vector Number(s)	Vector Offset (Hex)	Assignment
0	000	Reset Initial Interrupt Stack Pointer
1	004	Reset Initial Program Counter
2	008	Access Fault
3	00C	Address Error
4	010	Illegal Instruction
5	014	Integer Divide by Zero
6	018	CHK, CHK2 Instruction
7	01C	FTRAPcc, TRAPcc, TRAPV Instructions
8	020	Privilege Violation
9	024	Trace
10	028	Line 1010 Emulator (Unimplemented A- Line Opcode)
11	02C	Line 1111 Emulator (Unimplemented F-Line Opcode)
12	030	(Unassigned, Reserved)
13	034	Coprocessor Protocol Violation
14	038	Format Error
15	03C	Uninitialized Interrupt
16–23	040–05C	(Unassigned, Reserved)
24	060	Spurious Interrupt
25	064	Level 1 Interrupt Autovector
26	068	Level 2 Interrupt Autovector
27	06C	Level 3 Interrupt Autovector
28	070	Level 4 Interrupt Autovector
29	074	Level 5 Interrupt Autovector
30	078	Level 6 Interrupt Autovector
31	07C	Level 7 Interrupt Autovector
32–47	080–0BC	TRAP #0 D 15 Instruction Vectors
48	0C0	FP Branch or Set on Unordered Condition
49	0C4	FP Inexact Result
50	0C8	FP Divide by Zero
51	0CC	FP Underflow
52	0D0	FP Operand Error
53	0D4	FP Overflow
54	0D8	FP Signaling NAN
55	0DC	FP Unimplemented Data Type (Defined for MC68040)
56	0E0	MMU Configuration Error
57	0E4	MMU Illegal Operation Error
58	0E8	MMU Access Level Violation Error
59–63	0ECD0FC	(Unassigned, Reserved)
64–255	100D3FC	User Defined Vectors (192)

B.2 EXCEPTION STACK FRAMES

Figures B-1 through B-15 illustrate all exception stack frames for the M68000 family..

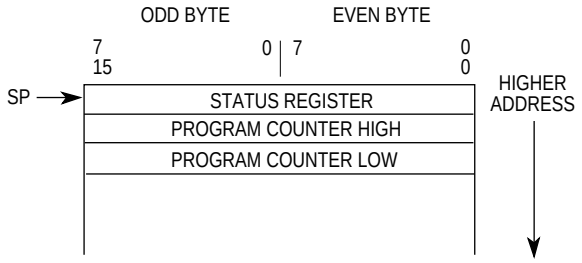


Figure B-1. MC68000 Group 1 and 2 Exception Stack Frame

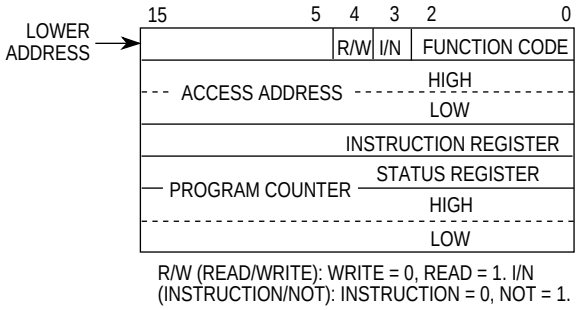


Figure B-2. MC68000 Bus or Address Error Exception Stack Frame

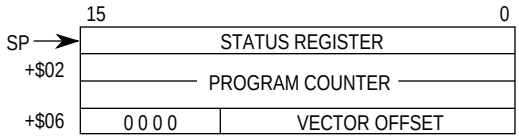


Figure B-3. Four-Word Stack Frame, Format $0

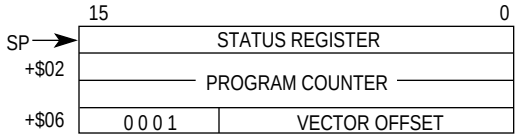


Figure B-4. Throwaway Four-Word Stack Frame, Format $1

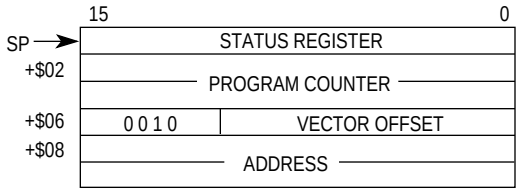


Figure B-5. Six-Word Stack Frame, Format $2

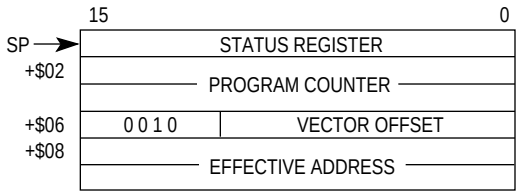


Figure B-6. MC68040 Floating-Point Post-Instruction Stack Frame, Format $3

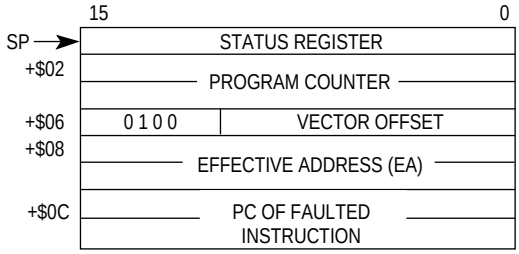


Figure B-7. MC68EC040 and MC68LC040 Floating-Point Unimplemented Stack Frame, Format $4

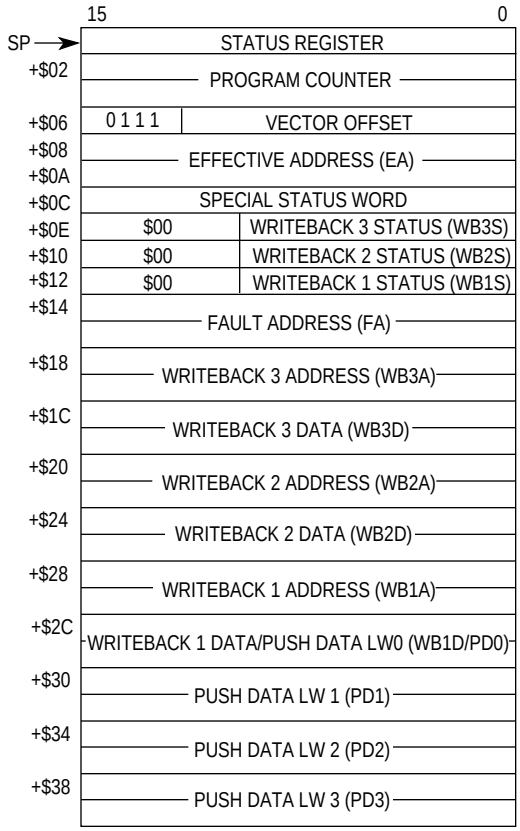
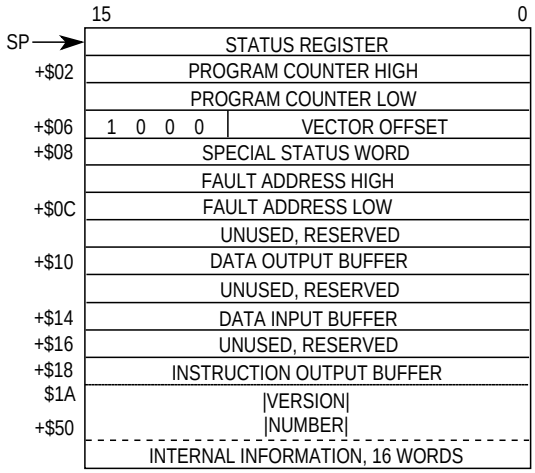


Figure B-8. MC68040 Access Error Stack Frame, Format $7



NOTE: The stack pointer decrements by 29 words, although only 26 words of information actually write to memory. Motorola reserves the three additional words for future use.

Figure B-9. MC68010 Bus and Address Error Stack Frame, Format $8

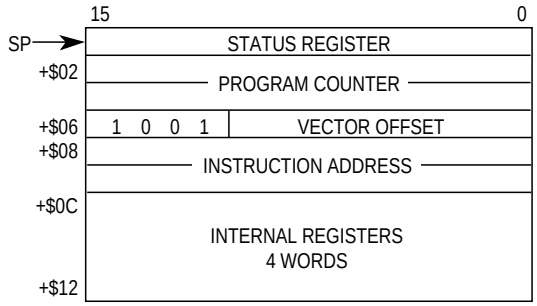


Figure B-10. MC68020 Bus and MC68030 Coprocessor Mid-Instruction Stack Frame, Format $9

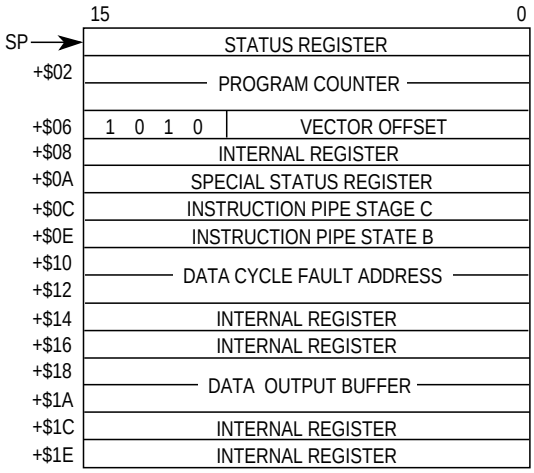


Figure B-11. MC68020 and MC68030 Short Bus Cycle Stack Frame, Format $A

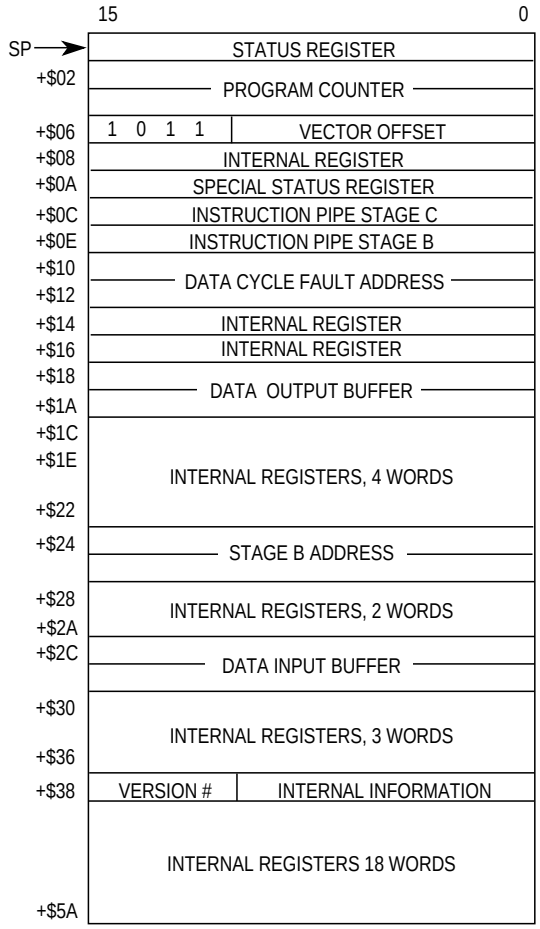


Figure B-12. MC68020 and MC68030 Long Bus Cycle Stack Frame, Format $B

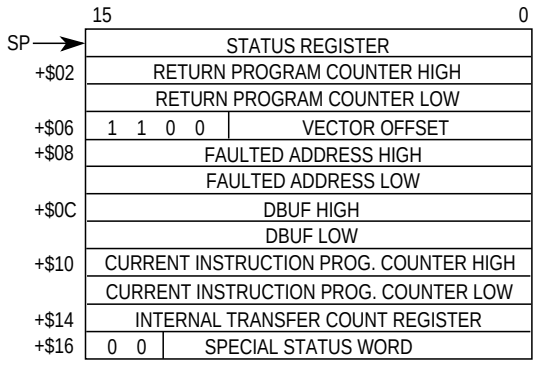


Figure B-13. CPU32 Bus Error for Prefetches and Operands Stack Frame, Format $C

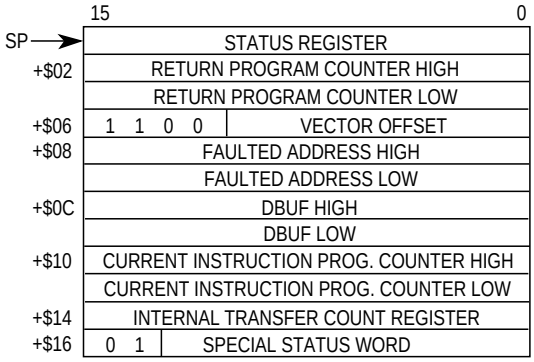


Figure B-14. CPU32 Bus Error on MOVEM Operand Stack Frame, Format $C

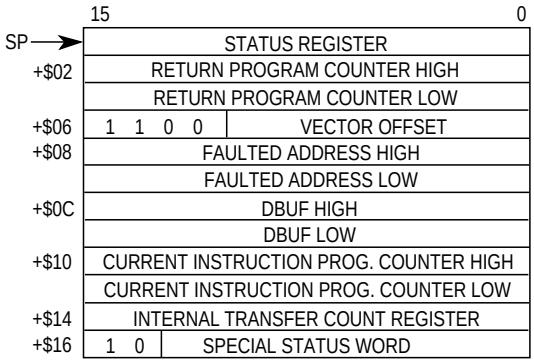


Figure B-15. CPU32 Four- and Six-Word Bus Error Stack Frame, Format $C

B.3 FLOATING-POINT STACK FRAMES

Figures B-16 through B-23 illustrate floating-point stack frames for the MC68881/MC68882 and the MC68040.

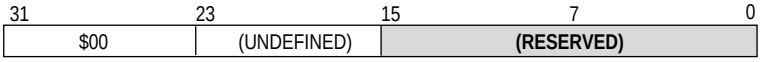


Figure B-16. MC68881/MC68882 and MC68040 Null Stack Frame

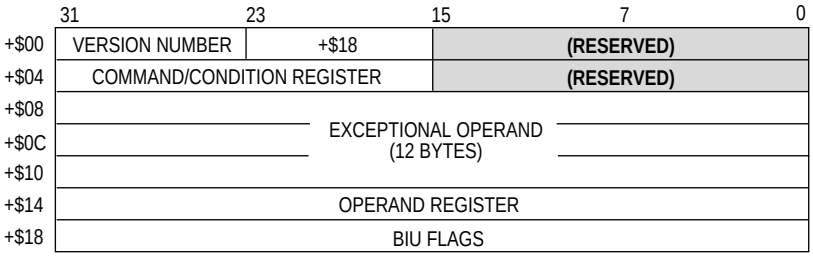


Figure B-17. MC68881 Idle Stack Frame

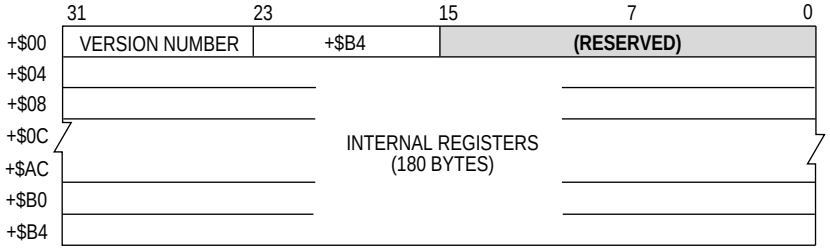


Figure B-18. MC68881 Busy Stack Frame

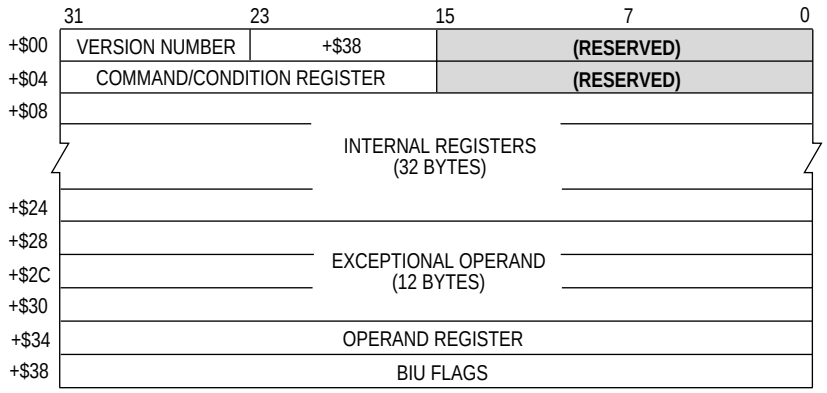


Figure B-19. MC68882 Idle Stack Frame

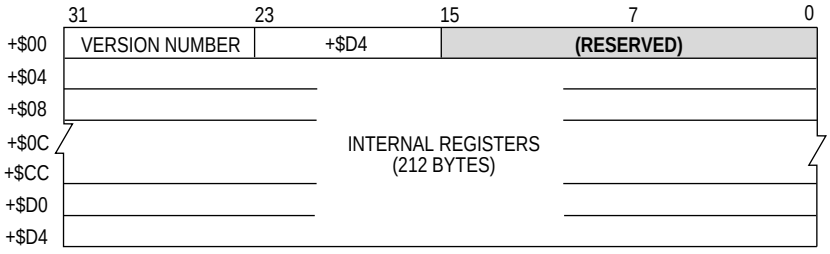


Figure B-20. MC68882 Busy Stack Frame

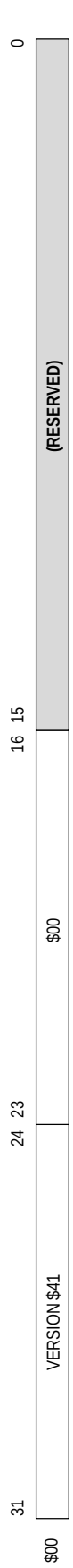


Figure B-21. MC68040 Idle Stack Frame

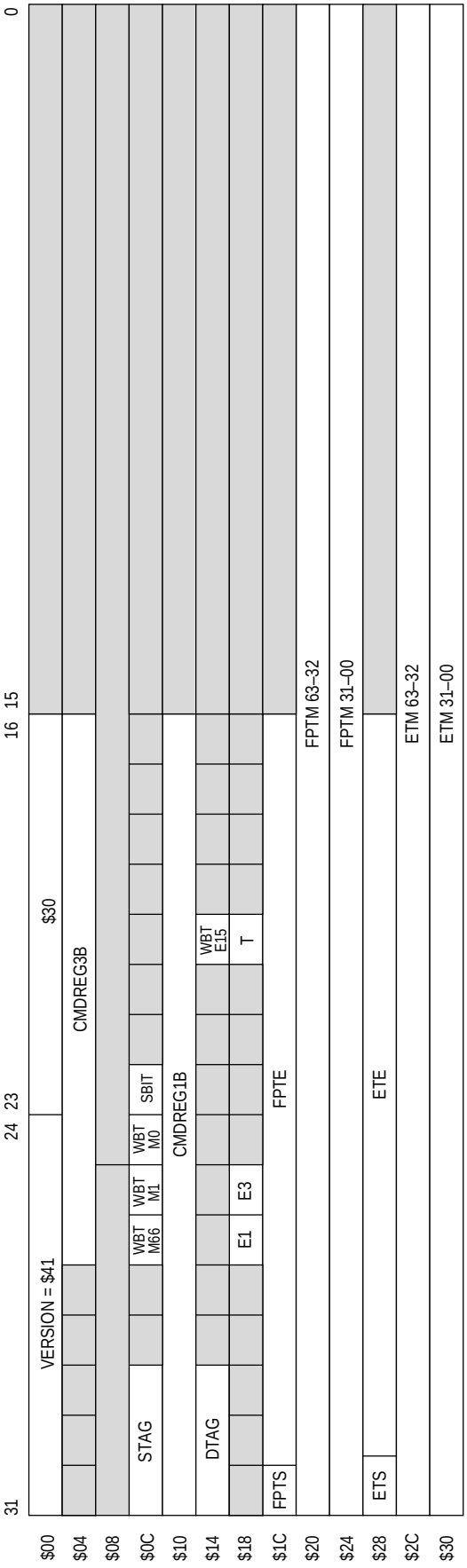


Figure B-22. MC68040 Unimplimented Instruction Stack Frame

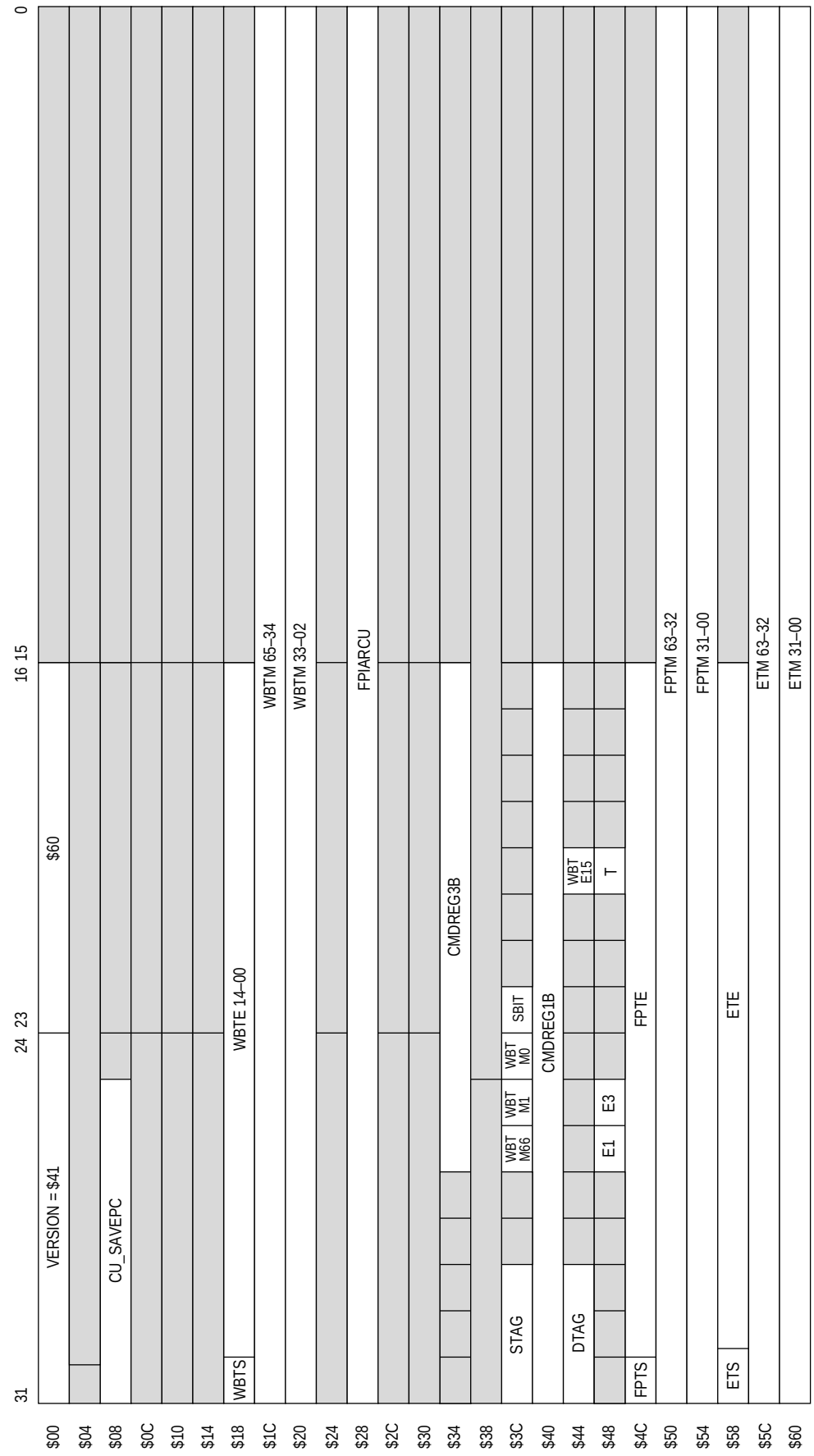


Figure B-23. MC68040 Busy Stack Frame

APPENDIX C
S-RECORD OUTPUT FORMAT

The S-record format for output modules is for encoding programs or data files in a printable format for transportation between computer systems. The transportation process can be visually monitored, and the S-records can be easily edited.

C.1 S-RECORD CONTENT

Visually, S-records are essentially character strings made of several fields that identify the record type, record length, memory address, code/data, and checksum. Each byte of binary data encodes as a two- character hexadecimal number: the first character represents the high- order four bits, and the second character represents the low-order four bits of the byte. Figure C-1 illustrates the five fields that comprise an S-record. Table C-1 lists the composition of each S- record field.

Figure C-1. Five Fields of an S-Record

Table C-1. Field Composition of an S-Record

Field	Printable Characters	Contents
Type	2	S-record type—S0, S1, etc.
Record Length	2	The count of the character pairs in the record, excluding the type and record length.
Address	4, 6, or 8	The 2-, 3-, or 4-byte address at which the data field is to be loaded into memory.
Code/Data	0–2n	From 0 to n bytes of executable code, memory loadable data, or descriptive information. For compatibility with teletypewriters, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-record).
Checksum	2	The least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

When downloading S-records, each must be terminated with a CR. Additionally, an S-record may have an initial field that fits other data such as line numbers generated by some time-sharing systems. The record length (byte count) and checksum fields ensure transmission accuracy.

C.2 S-RECORD TYPES

There are eight types of S-records to accommodate the encoding, transportation, and decoding functions. The various Motorola record transportation control programs (e.g. upload, download, etc.), cross assemblers, linkers, and other file creating or debugging programs, only utilize S-records serving the programOs purpose. For more information on support of specific S-records, refer to the userOs manual for that program.

An S-record format module may contain S-records of the following types:

S0 — The header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. Under VERSAdos, the resident linkerOs IDENT command can be used to designate module name, version number, revision number, and description information that will make up the header record. The address field is normally zeros.

S1 — A record containing code/data and the 2-byte address at which the code/data is to reside.

S2 — A record containing code/data and the 3-byte address at which the code/data is to reside.

S3 — A record containing code/data and the 4-byte address at which the code/data is to reside.

S5 — A record containing the number of S1, S2, and S3 records transmitted in a particular block. This count appears in the address field. There is no code/data field.

S7 — A termination record for a block of S3 records. The address field may optionally contain the 4-byte address of the instruction to which control is to be passed. There is no code/data field.

S8 — A termination record for a block of S2 records. The address field may optionally contain the 3-byte address of the instruction to which control is to be passed. There is no code/data field.

S9 — A termination record for a block of S1 records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed. Under VERSAdos, the resident linkerOs ENTRY command can be used to specify this address. If this address is not specified, the first entry point specification encountered in the object module input will be used. There is no code/data field.

Each block of S-records uses only one termination record. S7 and S8 records are only active when control is to be passed to a 3- or 4- byte address; otherwise, an S9 is used for termination. Normally, there is only one header record, although it is possible for multiple header records to occur.

C.3 S-RECORD CREATION

Dump utilities, debuggers, a VERSAdos resident linkage editor, or cross assemblers and linkers produce S-record format programs. On VERSAdos systems, the build load module (MBLM) utility allows an executable load module to be built from S-records. It has a counterpart utility in BUILDS that allows an S-record file to be created from a load module.

Programs are available for downloading or uploading a file in S- record format from a host system to an 8- or 16-bit microprocessor- based system. A typical S-record-format module is printed or displayed as follows:

```
S00600004844521B
S1130000285F245F2212226A000424290008237C2A
S11300100002000800082629001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S9030000FC
```

The module has an S0 record, four S1 records, and an S9 record. The following character pairs comprise the S-record-format module.

S0 Record:

S0 — S-record type S0, indicating that it is a header record.
06 — Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.
0000—A 4-character, 2-byte address field; zeros in this example.
48 — ASCII H
44 — ASCII D
52 — ASCII R
1B — The checksum.

First S1 Record:

S1 — S-record type S1, indicating that it is a code/data record to be loaded/verified at a 2-byte address.
13 — Hexadecimal 13 (decimal 19), indicating that 19 character pairs, representing 19 bytes of binary data, follow.
0000—A 4-character, 2-byte address field (hexadecimal address 0000) indicating where the data that follows is to be loaded.

The next 16 character pairs of the first S1 record are the ASCII bytes of the actual program code/data. In this assembly language example, the programOs hexadecimal opcodes are sequentially written in the code/data fields of the S1 records.

Opcode	Instruction	
285F	MOVE.L	(A7) +, A4
245F	MOVE.L	(A7) +, A2
2212	MOVE.L	(A2), D1
226A0004	MOVE.L	4(A2), A1
24290008	MOVE.L	FUNCTION(A1), D2
237C	MOVE.L	#FORCEFUNC, FUNCTION(A1)

The rest of this code continues in the remaining S1 recordOs code/data fields and stores in memory location 0010, etc.

2A — The checksum of the first S1 record.

The second and third S1 records also contain hexadecimal 13 (decimal 19) character pairs and end with checksums 13 and 52, respectively. The fourth S1 record contains 07 character pairs and has a checksum of 92.

S9 Record:

S9 — S-record type S9, indicating that it is a termination record.
03 — Hexadecimal 03, indicating that three character pairs (3 bytes) follow.
0000—The address field, zeros.
FC — The checksum of the S9 record.

Each printable character in an S-record encodes in hexadecimal (ASCII in this example) representation of the binary bits that transmit. Figure C-2 illustrates the sending of the first S1 record. Table C-2 lists the ASCII code for S-records.

.

TYPE				RECORD LENGTH				ADDRESS								CODE/DATA										CHECKSUM			
S		1		1		3		0		0		0		0		2		8		5		F		****		2		A	
5	3	3	1	3	1	3	3	3	0	3	0	3	0	3	0	3	2	3	8	3	5	4	6	****		3	2	4	1
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0011	0000	0011	0010	0011	1000	0011	0101	0100	0110	****		0011	0010	0100	0001

Figure C-2. Transmission of an S1 Record

Table C-2. ASCII Code

Least Significant Digit	Most Significant Digit							
	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL