2

inquiries welcome.

World copyrighted.

All rights

FOR A PARTICULAR PURPOSE.

SET

This table gives the addressing modes for each instruction. The Cycle and Flag Code column gives a code for operand sizes, instruction length, timing, and affect on flags. Example: Under ADD stc., Dd the flag code A applies to all adressing modes; cycle code 1 applies to ADD.4 Ds.pd and ADD.4 Ds.pd and ADD.4 Ds.pd and Cycle code 4 applies to ADD.4. Ds.pd. See Flag and Cycle Code tables.

(1) (See ADD, but no As, bd) Logical AND S ANDI (See ADDI) S ANDI HdaB, CCR BS2 A ANDI (See ADDI) S ANDI HdaB, CCR BS2 A (FI) Logical AND immediate ASL Os, bd BUS 110 A/S Hda3, bd BUS A ASI BAB, AND BUS A ASI BAB, AND BUS A ASI BAB, AND BUS A ASI BUS A B

HACKENSACK, NJ 07602 BCHG #da8,dst
Od L17
(Ad) B35
(Ad)+ B35
(Ad)+ B35
(Ad)+ B35
(Ad)+ B35
(Ad)+ B46
din6(Ad) B46
din6(Ad)- B70
addr.W B70
addr.W B70
addr.W B70
addr.W B70
addr.W B70
addr.W B70
BCB CHG
Exceptive
DLR (See BCHG,
Exceptive
Ds.Dd L9
#da8,Dd L24
Clear bit specified by
src in location
given by dst; put
complement of
original bit in Z (2)
BRA di8 11 N
BRA di16 12 N
Branch always; di8-0
not allowed
BSCT (See BCHG) V
Set bit specified by
src in location given
by dst; put
complement of
original bit in Z (2)
SRA di8 43 N
Branch always; di8-0
not allowed
BSCT (See BCHG) V
Set bit specified by
SRC in location given
by dst; put
complement of
original bit in Z (2)
SRA di8 43 N
BSR di16 47 N
Branch (call) to
subroutine; push long
word address of next
instruction using SP;
dde0 not allowed
BCB Os,dst V 3

greater than upper bound (0~\$7FFF); src is 2's complement (\$8000-\$FFFF) of

(SBUUL-SFIFT) or upper bound CLR Dd Bull L3 S (Ad) Bull 5 L51 (Ad) Bull 5 L53 (Ad) Bull 6 L63 addr. W Bull 5 L73 addr. L Bull 5 L90 CLear operand to 0; set N=0, Z=1 (A) CMP src, Dd 1 L3 As Ul L3 (As) Bull L3 (As) Bull L45 d18 (As, Xn) Bull L45 d18 (As, Xn) Bull L45 d18 (As, Xn) Bull 5 L53 addr. W Bull 6 L52 (As) Bull 6 L53 addr. W Bull 6 L53 compare Dd to src; subtract src from Dd, set flags, don't Change operands; BLT, for example, branches after CMP if Dd 1 L52 (As) Ul 1 L22 (As) Ul 1 L23 (As) Ul 1 L24 (

CMPI #da,dst T
Od 8WB L27
(Ad) 8WB L28
(Ad) 8WB L88
(Ad) 8WB L88
(Ad) 8WB L88
(Ad) 8WB L88
(Ad) 8WB L89
(Ad) 8WB L8

max
Division of John State of State of

higher byte or word with lower MSS ILLEGAL 103 ILLEGAL

LEA src,Ad N (As) L1 di16(As) L8 di8(As,Xn) L17 addr.W L8 di8(As,Xn) L17
addr.U L8
addr.L L20
di16(PC) L8
di8(PC,Xn) L17
Load effective
address; calculate
long word absolute
address of operand,
and put address in Ad
for later use
LINK As,di16 36 N
Link and allocate
stack space; save As
contents on stack,
copy new SP into As,
and add di16 to SP;
SP then points to
lowest and As to
highest address +1 of
stack space; di16
must be 2 s
complement of size;

Return from exception; pop SR then PC from stack

MOVEP Ds,di16(Ad)
W36 1.74 N
MOVEP di16(As),Dd
W36 1.72 N
MOVEP di16(As),Dd
W36 1.72 N
MOVE 2 or 4 bytes of
peripheral data
between a Data reg
and alternate bytes
of memory; high order
reg byte moves
to/from address, next
lower sis even,
lower sis even,
lower half of data bus if
address is even,
lower half of address
is odd; An does not
change; for 8-bit
peripherals
MOVEQ #da8,Dd 11 S
MOVEQ

NOP
No operation
NOT (See CLR) S
Logical (1's)
complement
OR (See ADU), except no
As,Ad)
Coglcal CR
(See ADU) S
Logical CR
(RI (See ADU)) S
Logical CR
(Logical CR

CYCLE CODES

Listed in order of increasing ULKs, increasing words, increasing reads-writes, and increasing writes. For ordering only, s=1 and r=2 are assumed.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 6 17 18 19 20 21 22 22 22 25 26 27 28 29 30 31 23 33 34 35 36 37 38 39 40 14 24 34 44 45 46 47 48 49 50 51 52 1.4.1.0 2.4.0.0 1.8.1.0 1.8.2.1 1.8.2.1 1.8.2.1 2.8.2.0 1.10.1.0 1.10.2.0 1.10.2.0 1.12.2.0 1.12.2.1 1.12.1.2 2.12.2.1 2.12.2.1 2.12.2.1 2.12.2.1 2.12.2.1 3.12.3.0 1.14.3.0 1.14.2.1 2.14.2.0 2.14.2.1 3.14.3.0 1.16.3.0 1.16.3.0 2.16.2.2 2.844r.2.r 3.16.3.0 3.16.4.0 3.16.3.1 1.18.4.0 1.18.2.2 2.18.3.0 2.18.4.0 2.18.3.1 2.18.2.2 3.18.4.0 1.20.5.0 1.20.3.2

Cycle codes give instruction length in words and execution time in CLKs. Bus read and write cycles are assumed to be four CLK cycles each. If bus wait states occur, you must add them to the CLKs. The numbers of read and write bus cycles per instruction are given for this purpose.

r = # of regs moved s = shift count

33 2.20.4.0
54 2.20.3.1
55 2.20.2.2
56 2.20.2.2
57 3.20.4.1
58 3.20.4.1
59 3.20.4.1
59 3.20.4.1
50 3.20.4.1
51 3.12.44r.3.r
52 3.20.4.1
53 4.20.4.1
54 1.22.2.2
55 3.22.4.0
57 2.22.2.2
58 3.22.4.0
57 2.22.2.2
58 3.22.4.0
59 3.22.5.0
70 3.22.4.1
71 3.14.44r.3.r
72 2.24.4.2
73 2.24.4.2
74 2.24.2.4
75 2.24.4.2
76 3.24.4.1
77 2.24.4.2
78 3.24.4.1
78 3.16.4.7
78 3.24.4.1
78 3.16.4.7
78 3.2.6.2
78 3.2.2.5
79 3.22.5.0
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2
70 3.22.2.2

CONDITION DEFS

(Applicable where indicated)
(1) Z is cleared if result is non-zero, but unchanged otherwise. Set or reset Z before start of operation to detect zero result, especially of repeated operation.
(2) Bit number is modulo 8 for memory operand, modulo 32 for reg operand (3) Shift or rotate count is modulo 8 for memory operand, modulo 64 for reg operand (4) A dst operand in memory is read before it is written (5) Flags are not affected if dst is Ad, except for CMPA

INSTRUCTION NOTES

MOVE TABLE

	dst~>				add	երև
		./	,di8	(Ad,	Xp)	
	(Ad)/(Ad	(Ad)	/ acc	L I M	-	
	src Do	1/Ad*	۱۳۲٬		1	1
	MOVE.8 ar	nd MO			,dst	
Ě	Ds	1	7	19	26	40
	As (As)	1 6	7 15	19 35	26 46	40 59
	(As)+	6	15	35	46	59
Š	-(As)	11	23	46	54	70
	di16(As)	18		59	70	80
	diB(As,Xr		46	70	77	86
ă	addr.W	18		59	70	80
	addr.L		59	80	86	94
	di16(PC) di8(PC,Xr	18 1) 25	35 46	59 70	70 77	80 86
	#da	1) 23	19	40	49	63
	n ou			40	70	00
	MOVE.L SI	c,ds	t:			
	0s	1	16	36	47	60
	As	1	16	36	47	60
	(As)	44	51	73	83	90

FLAGS

CC Carry Clear C*
CS Carry Set C
EQ Equal Z
F False 0
EG Greater or Equal
(N.V)+(N*,V*) GE Greater or Equal (N.V)+(N.V*)
GT Greater Than (N.V.Z*)+(N*,V*.Z*)
HI Higher C*.Z*
HIS Higher or Same C*
LE Less or Equal.

2*(N.V*)+(N*,V*)
LO Lower
LS Lower or Same C*.Z*
LT Less Than (N.V*)+(N*,V)
MI Minus
MI Minus
T. True I N. Tru

S = N 2 VC CC carry or borrow (1 T = N 2 VC CC carry or borrow (1 T = N 2 VC C hoste only V = N 2 VC C Rotate only V = N 2 VC C Rotate only V = N 2 VC C Rotate only N = None A, B, C mean all; N means none; S, T, U, V, W mean some f? becomes undefined fC cleared to zero f set/cleared according to result - not changed

-1ag	CCR Bit	Flag Name	Condition for flag=1, except as noted in instruction descriptions
X N	4 3	Extend Negative	Carry MSB of result = 1

1 Overflow Overflow occurs
O Carry Carry or borrow occurs

#115A

68000

STACKS AND

QUEUES

SYSTEM STACK: A7 is the system Stack Pointer used for subroutine calls. See Operands and Addressing. The stack grows from higher to lower addresses; SP points to last word pushed on stack; SP decrements before push, increments after pop. Any instruction using -(A7) as the destination operand is a push; any instruction using (A7)+ as the source operand is a pop.

USER STACK: To grow from higher address to lower address, use -(Ad) to push, (As) to pop; An points to top item. To grow from lower address to higher address, use (Ad) to push, -(As) to pop; An points to next free spot.

USER QUEUE: A FIFO list. To grow from lower address to higher address, use (Ad)+ to put, -(As) to get. To grow from higher address to lower address, use -(Ad) to put, (As)+ to get.

ADDRESSING MODES

HOW TO USE THIS **MICRO CHART**

The INSTRUCTION SET section describes each instruction and gives its addressing modes, assembler syntax, size, execution time, and effect on the flags.

The OPERANOS AND ADDRESSING section has general information on operand sizes, data organization in memory and registers, addressing modes, stacks, and queues.

The EXCEPTION PROCESSING section explains the 68000's response to errors, traps, interrupts, and other unusual conditions and its use of reserved memory locations.

The PINOUTS section lists the IC package pin numbers and signal names.

The ABBREVIATIONS section defines abbreviations used throughout this Micro Chart.

ABBREVIATIONS

* = Active low signal name suffix, or boolean inversion \$ = Nexadecimal Ad = Destination Address register (AO-A7) An = Address register (AO-A7) \$ = Source Address reniste

Ar Source Address register (AO-A7)
As = Source Address register (AO-A7)
addr = address
addr.l = 32-bit absolute
address
addr.l = 16-bit absolute
address
B = Operand size is byte
BU = Operand size is byte or

BM = Operand size is byte, word

BML = Operand size is byte, word, or long word

C = Carry flag in CCR

cc = Any of the sixteen
condition codes: CC, CS,
EC, F, GE, GT, HI, LE, LS,
LT, MI, NE, PL, T, VC, or

VS

CCR = Condition Code
resister

CCR = Condition Code register CLKs = Execution time of instruction in CLK cycles da = Immediate data da3 = 3-bit immediate data da4 = 4-bit immediate vector

number da8 = Immediate data byte da16 = Immediate data word da32 = Immediate data long

da8 = Immediate data byte
da18 = Immediate data word
da12 = Immediate data word
da22 = Immediate data word
da22 = Immediate data long
word
Od = Destination Data
register (OD-O7)
di8 = 8-bit displacement
th = Data register (OD-O7)
Da = Source Data register
(OD-O7)
data = 6-bit displacement
Dn = Data register (OD-O7)
Da = Source Data register
(OD-O7)
data = Effective address
Hax = Hexadecimal
III = Interrupt mask (Bits
110,9,8) in SR
L = Operand size is long
word
LSB = Least significant (low
order) bit; Bit O
MSB = Mest significant (high
order) bit; Bit O
MSB = Mest significant (high
order) bit; Bit O
MSB = Mest significant (high
order) bit; Bit O
MSB = Destination register
register
PI = Privileged instruction
Rd = Destination register
(AD-A7 or OD-O7)
Reads = Number of read bus
cycles during instruction
execution
execution
execution
execution
sequisters
Rsp = Source soperand
SSP = Supervisor bit (Bit 13)
in SR
SP = Stack Pointer register
including CRR
sr = Source operand
SSP = Supervisor Stack
Pointer register
V = Overflow flag in CCR
V = Overflow flag in

USP = USB.
register
V = Overflow flag in CCR
W = Operand size is word
WL = Operand size is word or

ML = Öperand sire is word or long word Words = Length of instruction in words Writes = Number of write bus cycles during instruction execution X = Extend flag in CCR Xn = Index register (AO-A7 or OO-D7) Z = Zero flag in CCR

The data bus strobes define how the data bus is used:

H H L L L H L L H L H L H n n 15-8 7-0 n 7-0 15-8 n 15-8 7-0 7-0m 7-0 15-8 15-8m

* = Active low signal H = High L = Low x = Don't care n = No valid data m = Maybe

OPERANDS AND ADDRESSING

INSTRUCTIONS: 1 to 5 words. Operation, register, length, and sometimes operand are given in first (Operation) word. D-4 Extension words specify immediate data, source address, and destination address operands in that order; each, if present, is 1-2 words.

Present, 18 1-2 words.

REGISITER'S Sixteen 32-bit general purpose registers consisting of eight processes and the second processes are selected for the second processes registers (AD-AT). One 32-bit Program Counter (PC), and one 18-bit Status Register (SR), The Condition Code register (CCR) is the lower byte of the SR. AT is the system Stack Pointer. One of two registers, SSP or USP, is used as AT; when one is active, the other is inaccessible; see Supervisor and User States below.

STATUS AND CONDITTON CODE REGISTERS:

	ę	5	/s	ter	n I	Зу	te		U	se:	r f	Зу	te	(1	CC	₹)	
Bit:	1	 5						8	7							ō	
iR:	ĩ	õ	s	õ	ō	ī	ī	ī	õ	ō	õ	ž	N	ī	v	Ē	

I: 1 = Irace mode, 0 = execute mode S: 1 = Supervisor state, 0 = User state III: Interrupt priority: 111 = 7 (highest and non-maskable) 000 = 0 (lowest) X,N,Z,V,C - See Flags Other bits are usually zero

SUPERVISOR STATE: The CPU is in Supervisor state when S=1. A7 is the SSP. All memory accesses are to the Supervisor memory space. All instructions are allowed. Only these privileged instructions can switch the CPU to User state by clearing the S bit: ANDI to SR, EORI to SR, MOVE to SR, or RTE.

USER STATE: The CPU is in User state when the S=O. A7 is the USP. All memory accesses are to the User memory space. An attempt to execute a Privileged instruction will cause an exception. Only an exception can switch the CPU to the Supervisor state.

OPERANDS

BIT NUMBERS: Low order (least significant) bit is numbered 0.

OPERAND SIZES: Add suffix .8, .W, or .L to instruction mnemonic for Byte (8 bits), Word (16), or Long Word (52). The default size is Word.

DATA REGISTER OPERANDS (00-07): can be 1, 8, 16, or 32 bits. Only low order part of register is used or changed for byte and word operands; high order part is not affected. Only one bit is used or changed for bit operations.

ADDRESS REGISTER OPERANDS (AO-A7): If destination, all 32 bits are affected, and SOURCE WORD OPERAND IS SIGN-EXTENDED to 32 bits before operation. If source, all or low order half is used.

INDEX REGISTER (AD-A7 or DO-D7): Any address or data register can be used as a word (Xn.W, sign-extended low order word) or a long word (Xn.L) index register.

register.

MEMORY OPERANDS: can be 1, 8, 16, or 32 bits. 1 byte per address. High order byte of word has same address (always even) as word; low order byte has next higher address (odd). Instructions and multibyte data start on even addresses. Long word at address N+2; second long word is at address N+2; second long word is at address N+4. Most significant digit of BCD byte is in high order bits; less significant digits are in bytes at higher addresses. The FC2-FCD outputs distinguish program references from data references; all writes are data references; all operand reads except PC relative are data references.

FC2	FC1	FLU	Cycle Type
L	L	L.	(reserved by Motorola)
Ļ	Ŀ	H	User Data
Ļ	н	Ŀ	User Program
L H	Н	н	(reserved for user def.)
Н	1.8	· н	(reserved by Motorola)
н	H	- 7.	Supervisor Data Supervisor Program
й	н	H	Interrent Acknowledge

LSI	MSD)	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 1 2 3	0000 0001 0010 0011	SOH		SP !	0 1 2 3	e A B C	P Q R S	a b c	s a b
4 5 6 7	0100 0101 0110 0111	ENQ ACK	NAK SYN	\$ % & -	4 5 6 7	0 5 5 6	T U V	d e f	tuva
8 9 A B	1000 1001 1010 1011	HT	CAN EM SUB ESC	() * +	8 9 :	H I J K	X Y Z [h i j k	x y z [
COEF	1100 1101 1110 1111	CR SO	F5 65 85 55	: ;	< = >?	1820	}	1 m n	DEL

EXCEPTION PROCESSING

The CPU's response to unusual internal or external conditions.

EXCEPTION VECTORS: Number Addr Dec Hex Hex Use

Vectors 0 and 1 are in Supervisor Program memory space; all others are in Supervisor Data memory space.

EXCEPTION VECTORS: Each (except 0) holds the long word address of an exception handling routine. Vector 0 is not a vector; it is the value loaded into the SSP after a RESET*.

VECTOR NUMBER: Provided by CPU or external logic. When multiplied by four, gives address of vector.

EXCEPTION PROCESSING TIMES: CLKs is the number of CPU CLK cycles to process the exception and fetch the first two words of the handler routine. Assumes a four CLK interrupt acknowledge bus cycle and no wait states. If CLKs are not shown here, see the Instruction Set section.

EXCEPTION PRIORITIES (Highest to Lowest):
Reset; bus error and halt; address error;
trace; external (user) interrupts ?
through 1; illegal instruction; privilege
violation; trap, check, and divide by
zero.

EXCEPTION DESCRIPTIONS Listed in order of decreasing priority. (reserved): Reserved for future use by Motorola; do not use

RESIT*, If RESIT* and HALT* are BOTH input low, the current bus cycle is aborted, and exception processing begins when they return high. The interrupt mask is set to 7 (III=1), no stacking occurs, and set to 5 (III=1), ro stacking occurs, and of the SIP and DC are loaded from Vectors 0 and 1. No other CPU selfeters are affected. The CPU output RESIT* low when it executes the RESIT instruction, but no registers are affected.

registers are affected.

BUS ERROR: When BERR* is input low, the CPU aborts the current bus cycle and floats the address and data busses. When the BERR* input returns high, the CPU stacks the Program Counter (unpredictable value), the Status Register, and four more words in this order: 1. The first word of the executing instruction; 2. The lower 16 bits of the aborted bus address; 3. The upper 16 bits of the address; 4. Five bits of bus cycle information: Bit 4: 1=read, 0=write; Bit 3 = 0 if the CPU was executing an instruction or processing a TRAP, TRAP, CRK or divide by zero exception; Bit 3 = 1 if the CPU was processing any other exception; Bits 2-0: FCZ-FCO.

When HALT* and BERR* are both input low, the CPU will abort the cycle, then re-run it when BERR* then HALT* return high. If a bus error occurs during bus or address error exception processing or while reading the vector table, the CPU halts.

HALT*: When HALT* is input low (with RESET* and BERR* high), the CPU finishes the current bus cycle, stops, and floats the address and data lines. But arbitration operates momenly during halt. The CPU will continue when HALT* returns

SOURCE,DESTINATION: Instructions that move data from a source to a destination are written in the form:
mnemonic src,dst

IMPLIED: Operand is in one of these registers: CCR, PC, SR, SP, SSP, or USP. Example: TRAPV

QUICK IMMEDIATE (Q #): 3-bit operand (1 to 8) is in operation word for ADDQ and SUBQ; 8-bit operand (-128 to +127) is in operation word for MOVEQ. Example: ADDQ #7,03

IMMEDIATE (#da): Byte operand is in low order byte of extension word; word operand is in extension word; long word operand is in 2 extension words. Example: ORI.B #\$7F,06

ABSOLUTE SHORT (addr.W): Extension word, sign-extended to 32 bits, is address of operand. Example: ASL VARG.W

ABSOLUTE LONG (addr.L): Two extension words are 32-bit address of operand. Example: CLR COUNT.L

PROGRAM COUNTER RELATIVE WITH DISPLACEMENT (dif6(PC)): Address of operand is sum of address of extension word and sign-extended displacement in extension word. Example: LEA LOCKUP(PC),A4

PROGRAM COLNIER RELATIVE MITH INDEX AND DISPLACEMENT (dis(PC,Xn)): Address of operand is sum of address of extension word, contents of index register, and sign-extended displacement in low byte of extension word. Index register can be any Address or Data register. Example: JMP NEXT(PC,OI.L)

DATA REGISTER DIRECT (Dn): Operand is in data register. Example: CLR.B DO

ADDRESS REGISTER DIRECT (An): Operand is in address register.
Example: CMPA.L DO,AO

ADDRESS REGISTER INDIRECT ((An)): Address of operand is in address register. Example: LSR (A5)

ADDRESS RECISTER INDIRECT WITH PREDECREMENT (-{An}) OR POSTINCREMENT ((An)+): Address of operand is in address register. Address register is decremented before use or incremented after use by 1, 2, or 4 depending on operand size. If size is byte and register is SP, adjustment is by 2, not 1. Examples: TAS -{A1} NEG.8 (A6)+

ADDRESS RECISTER INDIRECT WITH DISPLACEMENT (di16(Ar)): Address of operand is sum of sign-extended extension word and address register contents. Example: EORI.B #\$55,LIGHTS(A2)

ADDRESS REGISTER INDIRECT WITH INDEX AND DISPLACEMENT (di8(An,Xn): Address of operand is sum of address register contents, index register contents, and sign-extended displacement in LOW BYTE of extension word.

Example: ROL.W BIAS(AO,A1.W)

ASCII

_SD	000 001	010	011	100	101	110	111
	NUL DLE SOH DC1 STX DC2 ETX DC3		0 1 2 3	A B C	PQRS	a b c	b d t
5 0101 6 0110	EDT DC4 ENQ NAK ACK SYN BEL ETB	\$ %	4 5 6 7	D E F G	T U V W	d e f g	tuva
8 1000 9 1001 A 1010 B 1011		() * +	8 9 :	H I J K	X Y Z [h i j k	x y z [
C 1100 D 1101 E 1110 F 1111	CR GS SO RS	<u>:</u> ;	<= >?	L 8 2 0	}	1 m n o] DEL

high. The CPU outputs HALT* low when it stops because of double bus fault. Then only a low input on RESET* can restart the CPU. See RESET* and BERR*.

MICRO ®

CHART

ADDRESS ERROR: When the CPU fetches a word from an odd address, it responds as it does for a bus error. If a bus error occurs during address error exception processing, the CPU halts.

TRACE: When I=1 in the SR, an exception is forced after each instruction executes. An exception caused by an instruction is processed before the Trace exception is.

processed before the Trace exception is.

EXTERNAL INTERRUPTS: External logic encodes a priority level on IP(2*, IP(1*, IP(0*) (level sensitive). Level 7 is highest and not maskable. Level 1 is lowest. Level 0 is no interrupt. If the encoded level is 7, or greater than III, the CPU starts exception processing after it completes the current instruction. The CPU sets III to the encoded value when it forces S=1 and I=0 in the SR. The vector number is supplied internally (interrupt Acknowledge bus cycle) if VPR* is high; if BERR* is low, the Spurious Interrupt vector is used. Unintialized 68000 support chips give vector number 15.

USER INTERRUPTS: These are external interrupts for which external logic provides an 8-bit vector (\$40-\$FF) during the interrupt Reknowledge bus cycle.

the Interrupt Acknowledge bus cycle.

ILLEGAL, EMULATOR, AND UNIMPLEMENTED
INSTRUCTIONS: Any invalid instruction
opcode will cause an exception. Motorola
reserves each of these for future
definition except as follows. Opcodes
SAHFA, SAHFB, and SAHFC will always cause
an Illegal Instruction exception; the
first two are reserved for Motorola
products, and the third is reserved for
customer use. An opcode with 1010
(SAXXXX) or 1111 (SXXXXX) in Bits 15-12
will cause a Line 1010 or line 1111
Emulator exception, respectively. All
tother uninglemented opcodes cause an
Illegal Instruction exception.

PRIVILEGE VIOLATION: Execution of a privileged instruction (PI) in User state causes a privilege violation exception (ANDI #da16,SR; EDRI #da16,SR; MOVE SPC.SR; MOVE 48,USP; MOVE USP,Ad; ORI #da16,SR; RESET; RTE; STOP #da16). The sawed PC is the address of the first word of the PI.

TRAP, TRAPU, AND CHK: The TRAP instruction always causes a trap exception, and four bits in the instruction word provide part of the vector number. The TRAPU and CHK instructions cause an exception if certain conditions exist when they execute.

BUS ARBITRATION: determined by the BR*, BG*, and BGACK* signals.

PINOUTS 64-Pin DIP, Top View

EXCEPTION PROCESSING: All exception processing is done in the Supervisor state including use of the SSP for stacking. Except as noted below, the CPU: 1. Saves SR internally. 2. Forces S-1 and I=O in SR. 3. Gets the vector number. 4. Pushes the saved SR then the PC onto the stack using the SSP. 5. Loads the PC from the exception vector. 6. Executes a handler routine. The saved PC is usually the address of the first word of the next instruction. * means active low. < and > show direction. = means bidirectional. no means no connection inside.

n. D4= 1 64-D5
L: D3= 2 63-D6
D3= 2 63-D6
D3= 3 62-D7
D3= 6 62-D7
D3= 62-D7
D3= 62-D7
D4= 62-D7
D4