```
* THIS VERSION MODIFIED FOR TEESSIDE X68K PC-2.2 CROSS ASSEMBLER
*     ftp://nyquist.ee.ualberta.ca/pub/motorola/m68k/x68k.zip

*     CHANGES FROM ORIGINAL MOTOROLA SOURCE:
*         MANY "MOVE.L" CHANGED TO "MOVEQ"
*             SIMILAR CHANGES ON A FEW OTHER COMMANDS WHERE
*             X68K DOES NOT AUTOMATICALLY MOTOROLA-IZE.
*         X68K HANDLES CHARACTER CONSTANTS DIFFERENTLY
*             DC.W  '1'    -->  $3100  (GOOD)
*             DC.W  '1'+0  -->  $0031  (WHAT!)
*     ONE DIFFERENCE REMAINS IN A "DON'T CARE" BYTE
*         $008FE0 1A3C00FF  MOVE.B  #-1,D5
*                     ^^ $00 INSTEAD OF $FF

*=============================================================================*
* MOTOROLA EDUCATION COMPUTER BOARD ROM PROGRAM "TUTOR"  REVISION # 1.3 *
*=============================================================================*

* MACROS


ADDR2MEM MACRO
         PEA     \1(PC)          MOVE ADDRESS TO MEMORY; POSITION
         MOVE.L  (A7)+,\2        INDEPENDANT = TO "MOVE.L #\1,\2"
         ENDM


SAVEREGS MACRO
         MOVE.L  A7,REGA7        SAVE STACK POINTER
         LEA     SV\@(PC),A7     A7 = RETURN ADDRESS (FOR CALL TO SAVE)
         MOVE.L  A7,TEMP         TEMP = RETURN ADDRESS
         BRA     SAVE            BSR WITHOUT USING STACK
SV\@     DS      0
         ENDM




*-----------------------------------------------------------------------------

* EQUATES (in alphabetical order)

BELL     EQU     $07
BLANK    EQU     $20
BKPOINT  EQU     $4AFB
BUFFSIZE EQU     $80
BUFSIZE  EQU     80
CR       EQU     $0D
CTLD     EQU     $04
CTLH     EQU     $08
CTLW     EQU     $17
CTLX     EQU     $18
DEL      EQU     $7F
DELAYC1  EQU     $1000
EOT      EQU     $04
LF       EQU     $0A
LOCVARSZ EQU     16
RESET    EQU     $43             MASTER RESET FOR ACIA



*-----------------------------------------------------------------------------
* File ARAM      DATA VARIABLES                                  05/10/82

*****************************************************************
*  N O T E:  Ram locations:   starting at zero                  *
*****************************************************************

         ORG     $000000

*                                DEC HEX  DESCRIPTION
         DS.L    1               0   $00  AREA OVERLAID BY ROM SR
         DS.L    1               1   $01  AND PC


AV2      DS.L    1               2   $02  BUS ERROR            "BUS "
AV3      DS.L    1               3   $03  ADDRESS ERROR        "ADDR"
AV4      DS.L    1               4   $04  ILL INSTRUCTION      "OPCO"
         DS.L    1               5   $05  DIVIDE BY ZERO       "DIV0"
```

```
        DS.L    1               6   $06  CHECK TRAP           "CHCK"
        DS.L    1               7   $07  TRAP V               "TP V"
AV8     DS.L    1               8   $08  PRIVILEDGE VIOLATION "PRIV"
AV9     DS.L    1               9   $09  TRACE
        DS.L    1               10  $0A  1010 LINE EMULATION  "1010"
AV11    DS.L    1               11  $0B  1111 LINE EMULATION  "1111"
AV12    DS.L    1               12  $0C  USED AS TEMPORARY STORAGE FOR VECTOR MSGS.
        DS.L    1               13  $0D  NOT USED
        DS.L    1               14  $0E
        DS.L    1               15  $0F
        DS.L    1               16  $10
        DS.L    1               17  $11
        DS.L    1               18  $12
        DS.L    1               19  $13
        DS.L    1               20  $14
        DS.L    1               21  $15
        DS.L    1               22  $16
        DS.L    1               23  $17
AV24    DS.L    1               24  $18   0  AUTO VECTORS     "SPUR"
        DS.L    1               25  $19   1                   "AV#1"
        DS.L    1               26  $1A   2                   "AV#2"   TEST BUTTON
        DS.L    1               27  $1B   3                   "AV#3"
        DS.L    1               28  $1C   4                   "AV#4"
        DS.L    1               29  $1D   5                   "AV#5"
        DS.L    1               30  $1E   6                   "AV#6"
AV31    DS.L    1               31  $1F   7                   "AV#7   [ABORT BUTTON]
        DS.L    1               32  $20   TRAP  0             "UT 0"
        DS.L    1               33  $21   TRAP  1             "UT 1"
        DS.L    1               34  $22   TRAP  2             "UT 2"
        DS.L    1               35  $23   TRAP  3             "UT 3"
        DS.L    1               36  $24   TRAP  4             "UT 4"
        DS.L    1               37  $25   TRAP  5             "UT 5"
        DS.L    1               38  $26   TRAP  6             "UT 6"
        DS.L    1               39  $27   TRAP  7             "UT 7"
        DS.L    1               40  $28   TRAP  8             "UT 8"
        DS.L    1               41  $29   TRAP  9             "UT 9"
        DS.L    1               42  $2A   TRAP 10             "UT A"
        DS.L    1               43  $2B   TRAP 11             "UT B"
        DS.L    1               44  $2C   TRAP 12             "UT C"
        DS.L    1               45  $2D   TRAP 13             "UT D"
AV46    DS.L    1               46  $2E   TRAP 14             "UT E"
AV47    DS.L    1               47  $2F   TRAP 15             "UT F"
AV48    DS.L    1               48  $30
        DS.L    1               49  $31
        DS.L    1               50  $32
        DS.L    1               51  $33
        DS.L    1               52  $34
        DS.L    1               53  $35
        DS.L    1               54  $36
        DS.L    1               55  $37
        DS.L    1               56  $38
        DS.L    1               57  $39
        DS.L    1               58  $3A
        DS.L    1               59  $3B
        DS.L    1               60  $3C
        DS.L    1               61  $3D
        DS.L    1               62  $3E
        DS.L    1               63  $3F
        DS.L    1               64  $40
        DS.L    1               65  $41
        DS.L    1               66  $42
        DS.L    1               67  $43
        DS.L    1               68  $44
        DS.L    1               69  $45
        DS.L    1               70  $46
        DS.L    1               71  $47
        DS.L    1               72  $48
        DS.L    1               73  $49
        DS.L    1               74  $4A
        DS.L    1               75  $4B
        DS.L    1               76  $4C
        DS.L    1               77  $4D
        DS.L    1               78  $4E
        DS.L    1               79  $4F
        DS.L    1               80  $50
        DS.L    1               81  $51
        DS.L    1               82  $52
        DS.L    1               83  $53
        DS.L    1               84  $54
```

```
DS.L    1               85  $55
DS.L    1               86  $56
DS.L    1               87  $57
DS.L    1               88  $58
DS.L    1               89  $59
DS.L    1               90  $5A
DS.L    1               91  $5B
DS.L    1               92  $5C
DS.L    1               93  $5D
DS.L    1               94  $5E
DS.L    1               95  $5F
DS.L    1               96  $60
DS.L    1               97  $61
DS.L    1               98  $62
DS.L    1               99  $63
DS.L    1               100  $64
DS.L    1               101  $65
DS.L    1               102  $66
DS.L    1               103  $67
DS.L    1               104  $68
DS.L    1               105  $69
DS.L    1               106  $6A
DS.L    1               107  $6B
DS.L    1               108  $6C
DS.L    1               109  $6D
DS.L    1               110  $6E
DS.L    1               111  $6F
DS.L    1               112  $70
DS.L    1               113  $71
DS.L    1               114  $72
DS.L    1               115  $73
DS.L    1               116  $74
DS.L    1               117  $75
DS.L    1               118  $76
DS.L    1               119  $77
DS.L    1               120  $78
DS.L    1               121  $79
DS.L    1               122  $7A
DS.L    1               123  $7B
DS.L    1               124  $7C
DS.L    1               125  $7D
DS.L    1               126  $7E
DS.L    1               127  $7F
DS.L    1               128  $80
DS.L    1               129  $81
DS.L    1               130  $82
DS.L    1               131  $83
DS.L    1               132  $84
DS.L    1               133  $85
DS.L    1               134  $86
DS.L    1               135  $87
DS.L    1               136  $88
DS.L    1               137  $89
DS.L    1               138  $8A
DS.L    1               139  $8B
DS.L    1               140  $8C
DS.L    1               141  $8D
DS.L    1               142  $8E
DS.L    1               143  $8F
DS.L    1               144  $90
DS.L    1               145  $91
DS.L    1               146  $92
DS.L    1               147  $93
DS.L    1               148  $94
DS.L    1               149  $95
DS.L    1               150  $96
DS.L    1               151  $97
DS.L    1               152  $98
DS.L    1               153  $99
DS.L    1               154  $9A
DS.L    1               155  $9B
DS.L    1               156  $9C
DS.L    1               157  $9D
DS.L    1               158  $9E
DS.L    1               159  $9F
DS.L    1               160  $A0
DS.L    1               161  $A1
DS.L    1               162  $A2
DS.L    1               163  $A3
```

```
DS.L    1               164  $A4
DS.L    1               165  $A5
DS.L    1               166  $A6
DS.L    1               167  $A7
DS.L    1               168  $A8
DS.L    1               169  $A9
DS.L    1               170  $AA
DS.L    1               171  $AB
DS.L    1               172  $AC
DS.L    1               173  $AD
DS.L    1               174  $AE
DS.L    1               175  $AF
DS.L    1               176  $B0
DS.L    1               177  $B1
DS.L    1               178  $B2
DS.L    1               179  $B3
DS.L    1               180  $B4
DS.L    1               181  $B5
DS.L    1               182  $B6
DS.L    1               183  $B7
DS.L    1               184  $B8
DS.L    1               185  $B9
DS.L    1               186  $BA
DS.L    1               187  $BB
DS.L    1               188  $BC
DS.L    1               189  $BD
DS.L    1               190  $BE
DS.L    1               191  $BF
DS.L    1               192  $C0
DS.L    1               193  $C1
DS.L    1               194  $C2
DS.L    1               195  $C3
DS.L    1               196  $C4
DS.L    1               197  $C5
DS.L    1               198  $C6
DS.L    1               199  $C7
DS.L    1               200  $C8
DS.L    1               201  $C9
DS.L    1               202  $CA
DS.L    1               203  $CB
DS.L    1               204  $CC
DS.L    1               205  $CD
DS.L    1               206  $CE
DS.L    1               207  $CF
DS.L    1               208  $D0
DS.L    1               209  $D1
DS.L    1               210  $D2
DS.L    1               211  $D3
DS.L    1               212  $D4
DS.L    1               213  $D5
DS.L    1               214  $D6
DS.L    1               215  $D7
DS.L    1               216  $D8
DS.L    1               217  $D9
DS.L    1               218  $DA
DS.L    1               219  $DB
DS.L    1               220  $DC
DS.L    1               221  $DD
DS.L    1               222  $DE
DS.L    1               223  $DF
DS.L    1               224  $E0
DS.L    1               225  $E1
DS.L    1               226  $E2
DS.L    1               227  $E3
DS.L    1               228  $E4
DS.L    1               229  $E5
DS.L    1               230  $E6
DS.L    1               231  $E7
DS.L    1               232  $E8
DS.L    1               233  $E9
DS.L    1               234  $EA
DS.L    1               235  $EB
DS.L    1               236  $EC
DS.L    1               237  $ED
DS.L    1               238  $EE
DS.L    1               239  $EF
DS.L    1               240  $F0
DS.L    1               241  $F1
DS.L    1               242  $F2
```

```
         DS.L    1              243  $F3
         DS.L    1              244  $F4
         DS.L    1              245  $F5
         DS.L    1              246  $F6
         DS.L    1              247  $F7
         DS.L    1              248  $F8
         DS.L    1              249  $F9
         DS.L    1              250  $FA
         DS.L    1              251  $FB
         DS.L    1              252  $FC
         DS.L    1              253  $FD            3RD
         DS.L    1              254  $FE            2ND
         DS.L    1              255  $FF VECTOR FOR 1ST IPC DISK CONTROLLER


*  PSEUDO REGISTERS

REGPC    DS.L    1              USERS PROGRAM COUNTER
REGSR    DS.L    1              USERS CONDITION CODES
REGS     DS.L    8              D REGISTERS
         DS.L    7              A0 THROUGH A6 REGISTERS
REGA7    DS.L    1              A7 REGISTER
REGUS    DS.L    1              USER STACK



*************************************************************
*              WORKING STORAGE                              *
*  NOTE: SUBROUTINE "INITHRAM" ZEROES RAM...                *
*        FROM "BEGHRAM" THROUGH "ENDHRAM" WHEN IT IS EXECUTED  *
*************************************************************

BEGHRAM  DS      0              INITIALIZE STARTS HERE

OFFSET   DS.L    8              ASSUMED OFFSETS (VIA "R@" FORMAT)
MEMSIZE  DS.L    1              MEMORY SIZE IN BYTES
BPADD    DS.L    8              BREAKPOINT ADDRESSES
BPTILL   DS.L    1              TEMPORARY BREAKPOINT
BPCNT    DS.L    9              BREAKPOINT COUNTS
BPDATA   DS.W    9              HOLD USER WORDS REPLACED BY TRAP IN SET BP
BERRD    DS.L    2              SPECIAL DATA FOR BUS AND ADDR ERROR EXCEPTIONS
SAVEAV4  DS.L    1              HOLDS USER'S AV4 VECTOR (WE USE FOR BP)
TEMP     DS.L    1              TEMP
TRACECNT DS.L    1              TRACE COUNTER (-1=TRACE 1 & RUN)
TRACEON  DS.W    1              FLAG FOR TRACE ON
BPSTATUS DS.W    1              1=PB ARE IN  0=ARE OUT OF MEMORY
ECHOPT1  DS.L    1              ECHO FLAG TO PORT ONE



* THE FOLLOWING MUST REMAIN AS IS
*  User docomentation DEPENDS upon it!
*
OPTIONS  DS.W    0              FORCE WORD BOUNDRY
         DS.B    1              X-ON CHARACTER
         DS.B    1              X-OFF CHARACTER
         DS.B    1              NO NO-AUTO LINEFEED
         DS.B    1              SHORT FORM REGISTER DISPLAY
         DS.B    1              TM  trailing character
         DS.B    1              TM  exit character
XONOFF   EQU     OPTIONS
TMCHARS  EQU     OPTIONS+4

* END of as is section



**************************
* TARGET SUPERVISOR STACK *
**************************

         DS.L    20
SSA7     DS.L    1



**************
* I/O BUFFER *
**************
```

```
BUFFER   DS.B    BUFFSIZE

DUMPTEMP DS.B    80              HEADER TEMP BUFFER

SCREEN1  DS.L    1               PRINT THIS BEFORE TRACE DISPLAY
SCREEN2  DS.L    1               PRINT THIS AFTER TRACE DISPLAY

NULLPADS DS.B    2               CHARACTER NULL PADS
CRPADS   DS.B    2               CARRIAGE RETURN NULL PADS

OUTTO    DS.L    1               HOLDS ADDRESS OF OUTPUT ROUTINE
INFROM   DS.L    1               HOLDS ADDRESS OF INPUT ROUTINE
ALTSER1  DS.L    1               ALTERNATE SERIAL PORT#1
ALTSER2  DS.L    1               ALTERNATE SERIAL PORT#2
INPORT1  DS.L    1               INPUT ROUTINE ADDRESS
OUTPORT1 DS.L    1               ADDRESS FOR OUPUT ROUTINE
INPORT2  DS.L    1               ADDRESS FOR INPUT ROUTINE
OUTPORT2 DS.L    1               FOR OUTPURT ROUTINE
INPORT3  DS.L    1               THIS MIGHT BE FOR TAPE
OUTPORT3 DS.L    1               THIS MIGHT BE FOR PRINTER
INPORT4  DS.L    1               CASSETTE
OUTPORT4 DS.L    1               CASSETTE
MD1CON   DS.W    1               ACIA PROFILE (PORT1/PORT2)
PDIPORT  DS.L    1               PDIPORT ADDRESS
CRTPNT   DS.W    1               OUTPUT TO PRINTER AND CRT
TAPENULS DS.B    1               NULLS FOR CASSETTE

         DS.B    1               PAD BYTE

CTLINK   DS.L    1               POINTER TO FIRST TABLE



ENDHRAM  DS.W    0               MUST START ON WORD BOUNDRY



*********************
* SYSTEM STACK AREA *
*********************

         DS.W    0               FORCE ON WORD BOUNDRY
         DS.B    300             ROOM FOR STACK
SYSTACK  DS.W    1               START OF STACK (ADDRESS DECREASES)
         DS.B    4               STRETCHED STACK (USED BY 'SAVE')

         DS.B    120             EXTENDED AREA USED IF DISASSEMBLER

         DS.B    0               LAST LOW MEMORY LOCATION USED + 1


*---------------------------------------------------------------------------

*CODE68K
*                                OFFSET  0
ESKB     EQU     $00000000       DS      0
TDATA    EQU     $00000000       DS.B    10
TNB      EQU     $0000000A       DS.B    1
TLSPEC   EQU     $0000000B       DS.B    1
TLENGTH  EQU     $0000000C       DS.W    1
PCOUNTER EQU     $0000000E       DS.L    1
PTROP    EQU     $00000012       DS.L    1               POINTER TO OPERAND
PENDOP   EQU     $00000016       DS.L    1               POINTER END OF OPERAND
PTRBUFE  EQU     $0000001A       DS.L    1               POINTER TO END OF FORMATED SOURCE
LINK     EQU     $0000001E       DS.L    1               SAVE FOR UNLINK
ESKE     EQU     $00000022       DS.B    0


*DCODE68K
*                                OFFSET  -LOCVARSZ
DDATA    EQU     $FFFFFFF0       DS.L    3
HISPC    EQU     $FFFFFFFC       DS.L    1


*LOAD
*                                OFFSET  -((BUFFSIZE/2)+4)
CC       EQU     $FFFFFFBC       DS.L    1               CC (BYTE COUNT)
ADDRESS  EQU     $FFFFFFC0       DS.L    1               ADDRESS + OFFSET
```

```
LDATA    EQU     $FFFFFFC4      DS.B    1



*-----------------------------------------------------------------------
* File B         Init Vectors+Ram                                 05/29/82

         ORG     $008000

FIRST    DC.L    REGA7          SUPERVISOR STACK
         DC.L    START          PROGRAM COUNTER
V2       BRA.L   TRACE


*********************************
* INITIALIZE HIGH RAM SUBROUTINE *
*********************************

INITHRAM LEA     BEGHRAM,A0     START OF WORK RAM (PAST REGISTERS)
         MOVE.L  #(ENDHRAM-BEGHRAM),D0  BYTES TO ZERO
         CLR.L   D1
INIT     MOVE.B  D1,(A0)+       ZERO MEMORY
         SUBQ.L  #1,D0
         BNE     INIT

         ADDR2MEM BERRMSG,AV2   POINT AT BUS TRAP ERROR MESSAGE ROUTINE

         ADDR2MEM ADDRMSG,AV3   POINT AT ADDRESS TRAP ERROR MESSAGE ROUTINE

         RTS


************************************************
* SPECIAL HANDLING FOR BUS ERROR AND ADDRESS ERROR *
************************************************

BERRMSG  MOVE.L  #'BUS ',$30

         BRA.S   VECTBE

ADDRMSG  MOVE.L  #'ADDR',$30


VECTBE   MOVE.L  (A7)+,BERRD
         MOVE.L  (A7)+,BERRD+4

         SAVEREGS
         BSR     FIXBUF
         MOVE.W  #$D0A,(A6)+
         MOVE.W  BERRD,D0
         BSR     PNT4HX         FORMAT FUNCTION CODE

         MOVE.B  #BLANK,(A6)+   SPACE
         MOVE.L  BERRD+2,D0
         BSR     PNT8HX         FORMAT ACCESS ADDRESS

         MOVE.B  #BLANK,(A6)+   SPACE
         MOVE.W  BERRD+6,D0
         BSR     PNT4HX         FORMAT INSTRUCTION REGISTER

         BSR     OUTPUT
         BRA     EVECT2         GO DISPLAY REGISTERS & PROMPT

MSG008   DC.B    'SYNTAX '

MSG008E  DC.B    'ERROR '

MSGEOT   DC.B    EOT

MSG021   DC.B    'WHAT',EOT

         DC.B    0              PAD BYTE


* PRINT WHAT

WHAT     LEA     MSG021(PC),A5  PRINT 'WHAT' AND ENTER MACSBUG
WHAT93   BSR.S   FIXDATA
```

```
         CLR.W   TRACEON
MSG      BSR     OUT1CR
         BRA     MACSBUG


* PRINT ERROR

ERROR    LEA     MSG008E(PC),A5
         BRA     WHAT93

SYNTAX   LEA     MSG008(PC),A5  'SYNTAX ERROR'
         BRA     WHAT93

* FORMAT PHYSICAL ADDRESS FROM (D0)
PPHY     LEA     MSG019(PC),A5
         BSR.S   FIXDATA
         BRA.S   P2PHY2

* FORMAT TWO PHYSICAL ADDRESSES FROM (A0) & (A1)
P2PHY    LEA     MSG019(PC),A5  'PHYSICAL ADDRESS='
         BSR.S   FIXDATA
         MOVE.L  A0,D0
         BSR     PNT8HX         FORMAT ADDR1
         MOVE.B  #BLANK,(A6)+   SPACE FOR FORMATING
         MOVE.L  A1,D0
P2PHY2   BSR     PNT8HX         FORMAT ADDR2
         BSR     OUT1CR         DISPLAY IT
         RTS

MSG019   DC.B    'PHYSICAL ADDRESS=',EOT




************************************************************
* -FIXDATA- SUBROUTINE...  MOVES MESSAGE POINTED TO BY (A5) *
*                          INTO "BUFFER". EOT, ($04), ENDS  *
*                          THE MOVE. AT COMPLETION (A5) IS  *
*                          POINTING AT THE BEGINNING, (A6)  *
*                          POINTS AT END.                   *
************************************************************

FIXDATA  LEA     BUFFER,A6
FIXDADD  CMPI.B  #EOT,(A5)
         BEQ.S   FIXD2
         MOVE.B  (A5)+,(A6)+
         BRA     FIXDADD
FIXD2    LEA     BUFFER,A5
         RTS


************************************************************
* -FIXDCRLF- SUBROUTINE INSERTS A CARRIAGE RETURN AND LINE *
*                        FEED IN FRONT OF THE TEXT, THEN   *
*                        USES THE REMAINING PORTION OF THE *
*                        FIXDATA.                          *
************************************************************

FIXDCRLF LEA     BUFFER,A6
         MOVE.W  #$0D0A,(A6)+   CR,LF
         BRA     FIXDADD




*-------------------------------------------------------------------------
* File E         VERSAbug entry point                          01/08/81

*********************
* INITIALIZE VECTORS *
*********************
*                               Set most vectors to point at "????" routine
INITVECT LEA     8,A0           Skip (Restart) STACK & ADDRESS vectors
         LEA     ABORTE(PC),A1  A1 = "Default" TRAP ERROR routine address

INIT0    MOVE.L  A1,(A0)+       INITIALIZE VECTOR
         CMPA.L  #$400,A0       Done?
         BMI.S   INIT0          *
```

```
        RTS


* SPECIAL ENTRY THAT DOES NOT CHANGE VECTORS

START1S  MOVEM.W D0,REGSR+2     Assure good parity.
         MOVE.W  SR,REGSR+2     SAVE TARGET'S STATUS REGISTER
         MOVE.L  A7,REGA7       SAVE TARGET'S STACK
         MOVE.L  (A7),REGPC     .PROGRAM COUNTER
         LEA     REGA7,A7
         MOVEM.L D0-D7/A0-A6,-(A7) .REGISTERS
         LEA     SYSTACK,A7
         BRA     START11


************************
*    INITIALIZATION    *
************************

* SAVE PROCESSOR REGISTERS (EXCEPT A7 & PC)

START    MOVEM.W D0,REGSR+2     Assure good parity
         MOVE.W  SR,REGSR+2     SAVE STATUS REGISTER
         MOVEM.L D0-D7/A0-A6,-(A7)

         LEA     SYSTACK,A7     SET UP STACK
         MOVE.L  A7,REGA7

         CLR.L   D1
         MOVE.L  D1,REGPC       PC = 000000

         BSR     INITVECT


START11  MOVE.W  #$2700,SR      MASK OFF INTERRUPTS

         MOVE.L  USP,A0
         MOVE.L  A0,REGUS       USER STACK

         BSR     INITHRAM       ZERO (INITIALIZE) HIGH RAM


* VECTMSG.SA
         BSR     INITVMSG


* H.SA
         ADDR2MEM CHKBP,AV4     ILLEGAL INSTRUCTION


* TM.SA
         MOVE.W  #$1801,TMCHARS CNTLX,CNTL/A


* W.SA
         ADDR2MEM  ABORTB,AV31  ABORT


* Y.SA
         ADDR2MEM  OUT1CR0,OUTPORT1 INITIALIZE I/O ROUTINES
         ADDR2MEM  OUTPUT20,OUTPORT2
         ADDR2MEM  PRCRLF,OUTPORT3     PRINTER DRIVER
         ADDR2MEM  TAPEOUT,OUTPORT4    CASSETTE
         ADDR2MEM  PORTIN10,INPORT1
         ADDR2MEM  PORTIN20,INPORT2
         ADDR2MEM  PORTIN10,INPORT3
         ADDR2MEM  TAPEIN,INPORT4      CASSETTE

         MOVE.B  #8,TAPENULS    NULLS FOR CASSETTE
         MOVE.L  #PDI1,PDIPORT  PRINTER


*        INITIALIZE MC68230 PI/T
         MOVE.L  #PDI1,A0       BASE ADDRESS OF PI/T
         MOVE.L  #$0000FF00,D0
         MOVEP.L D0,1(A0)

*        SELECT MODE 0
*        IRQ'S INACTIVATED
```

```
*        PORT A--ALL BITS OUTPUTS
*        PORT B--ALL BITS INPUTS

         MOVE.B  #$60,13(A0)    SUBMODE 01 FOR PORT A; INTERLOCKED HANDS
         MOVE.B  #$A0,15(A0)    SUBMODE 1X FOR PORT B
         MOVE.B  #$30,1(A0)     ENABLE HANDSHAKE LINES
         MOVE.B  #$A8,15(A0)    RESET AND INIT PRINTER
         MOVE.L  #PDI1+$10,PDIPORT

         MOVE.B  #$A0,15(A0)    CLEAR INIT

* INITIALIZE THE PDI'S

         MOVE.W  #$1515,MD1CON
         BSR     INITSER        RESET & PROGRAM PDI

* INITIALIZE XON/XOFF (READER ON / READER OFF)
*            AUTO-LINE FEED OVERRIDE

         MOVE.L  #$00000000,XONOFF



* TRAP14.SA
         ADDR2MEM  TRAP14,AV46
         MOVE.L  #(254<<24)+CT,CTLINK




*************************************************************************
*                   V E R S I O N   N U M B E R   A N D   P R O M P T *
*************************************************************************

MACSBUG  MOVE.W  #$2700,SR      MASK OFF INTERRUPTS
         LEA     SYSTACK,A7     RESTORE SYSTEM STACK
         BSR     SWAPOUT        GET BP OUT OF USER MEMORY

         CLR.L   BPTILL         GET RID OF 'TILL' BREAKPOINT
         CLR.L   OUTTO          INITIALIZE I/O TO DEFAULT
         CLR.L   INFROM         INITIALIZE I/O TO DEFAULT
         CLR.B   ECHOPT1        NO ECHO TO PORT1

         LEA     MSG001(PC),A5  > (Prompt)
         BSR     FIXDATA

         TST.W   TRACEON        SEE IF IN TRACE MODE
         BEQ.S   MACSBUG1
         MOVE.B  #':',(A6)+     IN TRACE MODE
MACSBUG1 MOVE.B  #'>',(A6)+     PROMPT
         MOVE.B  #BLANK,(A6)+   .. SPACE
         BSR     OUTPUT         GO PRINT IT

* INPUT LINE
         BSR     FIXBUF         GET READY FOR INPUT
         BSR     PORTIN1        GET A COMMAND
DECODE6  MOVE.B  #BLANK,(A6)    BLANK OUT END+1

*  DECODE A COMMAND
*
*  DECODE SPECIAL CHARACTER USAGE:
*    LEADING SPACES IGNORED
*    LEADING NULLS IGNORED
*    IF SECOND CHAR  *  CHAR CAN BE ANY CHAR

         CMP.L   A6,A5          SEE IF ANYTHING ENTERED
         BMI.S   DECODE1
         TST.W   TRACEON        SEE IF IN TRACE MODE
         BNE     TCMDHOT        DIRECT TO TRACE 1 COMMAND

DECODE1  CMP.L   A6,A5          SEE IF AT END OF BUFFER
         BHI     WHAT           GO TO 'WHAT' IF CONFUSED
         MOVE.B  (A5),D0        GRAB FIRST CHARACTER
         CMPI.B  #'*',D0        SEND LINE COMMAND
         BNE.S   DECODE10
         ADDQ.L  #1,A5          GET PAST PHOENY PROMPT
         BSR     OUTPUT2        SEND LINE+CR (NO LF) TO PORT2
         BRA     MACSBUG        REENTER COMMAND MODE
```

```
DECODE10 CMPI.B  #$20,D0        IGNORE LEADING SPACES
         BNE.S   DECODE2        WHERE TO GO IF NOT A SPACE
         ADDQ.L  #1,A5          BUMP START OF BUFFER
         BRA.S   DECODE1        TRY NEXT CHARACTER
*
DECODE2  MOVE.B  (A5),D1        GET 2 LETTERS OF COMMAND
         LSL.W   #8,D1          MAKE ROOM FOR SECOND CHAR
         MOVE.B  1(A5),D1       GET SECOND CHAR
         CLR.L   D3             D3 = CLEAR "NO" SWITCH

DECODE21 LEA     SOLIST(PC),A1  A1 = COMMAND LIST ADDRESS
DECODE4  MOVE.W  (A1)+,D2       D2 = 2 CHAR COMMAND FROM LIST
         CLR.L   D0             CLEAR HIGH BITS
         MOVE.W  (A1)+,D0       D0 = OFFSET FROM START OF ROM

         TST.L   D3
         BEQ.S   DECODE41       NOT A "NO"
         TST.B   D2             IS "NO" OPTION SUPPORTED THIS COMMAND?
         BPL     DECODE4        NO...THEN RUN OUT OF COMMANDS

DECODE41 ANDI.W  #$7F7F,D2      CLEAR "INVISABLE" & "NO" BITS
         CMPI.W  #$7F7F,D2      END OF LIST?
         BEQ     WHAT           Command not found

         CMPI.B  #'*',D2        SEE IF DON'T CARE CHARACTER
         BNE.S   DECODE3
         MOVE.B  D1,D2          DEFAULT

DECODE3  CMP.W   D1,D2          Command from table = the input?
         BNE     DECODE4        COMMAND NOT FOUND

         CLR.W   TRACEON        TURN OFF TRACE MODE

         ADDQ.L  #2,A5          POINT A5 PAST 2 DIGIT COMMAND

         PEA     FIRST(PC)      BUILD GO TO ADDRESS
         ADD.L   D0,(A7)        ON STACK.
         ADD.L   D3,(A7)

         MOVE.L  (A7)+,A0       GO TO COMMAND
         JSR     (A0)           * SAVE MARK FOR RETURN
         BRA     MACSBUG        *  RETURN HERE

*
*    NO COMMAND
*
NOCMD    MOVEQ   #-4,D3         SET "NO" SWITCH
         MOVE.B  (A5),D1        MOVE CHAR #3
         ASL.W   #8,D1          MOVE OVER 1 CHAR
         MOVE.B  1(A5),D1       MOVE CHAR #4
         BRA     DECODE21       WHICH "NO" COMMAND?


*-------------------------------------------------------------------------
* File COMMANDS  Command list                                   06/20/82


MSG001   DC.B    CR,LF,'TUTOR  1.3 ',EOT    "PROMPT"




*******************************************************
* C O P Y R I G H T . 1 9 8 1 . B Y . M O T O R O L A *
*******************************************************

* VERSAbug command generation macro
CMD      MACRO
FLAG     SET     0              *
         IFC     '\2','HELP=NO'
FLAG     SET     FLAG+$8000     * "Help" will not display this command
         ENDC
         IFC     '\3','HELP=NO'
FLAG     SET     FLAG+$8000     * "Help" will not display this command
         ENDC
         IFC     '\2','NORTN=YES'
FLAG     SET     FLAG+$80       * "NO\1".Command
         ENDC
```

```
         IFC     '\3','NORTN=YES'
FLAG     SET     FLAG+$80        * "NO\1".Command
         ENDC
         IFC     '\1','PER'      Check for the "PER" command
         DC.W    '.*'+FLAG       * Reg commands (.A2 .D6 .PC .R0 etc.)
         DC.W    \1CMD-FIRST     ************************************
         ENDC
         IFNC    '\1','PER'      If not PERCMD...
         IFEQ    '\1'&($FF00)    If 1 digit code, 2nd will be a blank.
         DC.W    '\1 '+FLAG      * "\1"....Command  -  -  (Single Digit)
         ENDC
         IFNE    '\1'&($FF00)    If 2 digit code, leave as is.
         DC.W    '\1'+FLAG       * "\1"...Command
         ENDC
         DC.W    \1CMD-FIRST     ************************************
         ENDC
         ENDM


SOLIST   DS      0               Start Of LIST

         CMD     PER,HELP=NO,X,X

         CMD     NO,HELP=NO,X,X

         CMD     BF,X,X

         CMD     BM,X,X


         CMD     BR,NORTN=YES,X,X

         CMD     BS,X,X

         CMD     BT,X,X

         CMD     DC,X,X

         CMD     DF,X,X

         CMD     DU,X,X


         CMD     G,X,X

         CMD     GD,X,X

         CMD     GO,X,X

         CMD     GT,X,X

         CMD     HE,X,X

         CMD     LO,X,X

         CMD     M,X,X


         CMD     MD,X,X

         CMD     MM,X,X

         CMD     MS,X,X

         CMD     OF,X,X

         CMD     PA,NORTN=YES,X,X

         CMD     PF,X,X


         CMD     T,X,X

         CMD     TM,X,X
```

```
         CMD     TR,X,X

         CMD     TT,X,X

         CMD     VE,X,X

         DC.W      $FFFF     End of list indicator



*-----------------------------------------------------------------------
* File VECTMSG   Messages for vectors                            05/29/82

************************************************************
*Reprogram some VECTORS to specific ERROR handler routines *
************************************************************

INITVMSG LEA     VECT(PC),A0    A0 = START OF VECTOR TABLE
         LEA     AV4,A1         A1 = FIRST VECTOR TO INITIALIZE
         MOVEQ   #10,D0         D0 = COUNT
VECTI    MOVE.L  A0,(A1)+       MOVE ADDRESS TO VECTOR
         ADD.L   D0,A0          BUMP ADDRESS
         CMPA.L  #AV11+4,A1
         BNE     VECTI

         LEA     AV24,A1        A1 = NEXT VECTOR TO INITIALIZE
VECTI2   MOVE.L  A0,(A1)+       MOVE ADDRESS TO VECTOR
         ADD.L   D0,A0          BUMP ADDRESS
         CMPA.L  #AV48,A1
         BNE     VECTI2
         RTS


************************************************************************
* STANDARD VECTOR "MESSAGE" HANDLING ROUTINE ($30 IS TEMP STORAGE AREA) *
************************************************************************

VECT     MOVE.L  #'OPCO',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'DIV0',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'CHCK',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'TP V',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'PRIV',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'TRAC',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'1010',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'1111',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT5
         MOVE.L  #'SPUR',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

EVECT5   BRA.S   EVECT6
         MOVE.L  #'AV#1',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT6
         MOVE.L  #'AV#2',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT6
         MOVE.L  #'AV#3',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT6
         MOVE.L  #'AV#4',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT6
         MOVE.L  #'AV#5',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT
```

```
         BRA.S   EVECT6
         MOVE.L  #'AV#6',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT6
         MOVE.L  #'AV#7',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

EVECT6   BRA.S   EVECT7
         MOVE.L  #'UT 0',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 1',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 2',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 3',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 4',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 5',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 6',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT7
         MOVE.L  #'UT 7',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

EVECT7   BRA.S   EVECT
         MOVE.L  #'UT 8',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT
         MOVE.L  #'UT 9',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT
         MOVE.L  #'UT A',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT
         MOVE.L  #'UT B',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT
         MOVE.L  #'UT C',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT
         MOVE.L  #'UT D',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BRA.S   EVECT
         MOVE.L  #'UT E',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT

         BSR.S   EVECT
         MOVE.L  #'UT F',$30    MOVE TO $30, USE SHORT BRANCHES AND PRINT IT


*
*    SAVE REGISTERS AND PRINT VECTOR MSG
*
EVECT    BRA     EVECTL



*---------------------------------------------------------------------
* File BF        Block Fill command                              06/16/82

*  BLOCK FILL    ADDRESS1 ADDRESS2 WORD-DATA

BFCMD    LEA     SYNTAX(PC),A0
         BSR     FNEXTF
         BSR     GETA
         BSR     CKWADR         CHECK WORD BOUNDRY ADDRESS
         MOVE.L  D0,D6          D6 = FROM BEGIN ADDRESS

         BSR     FNEXTF
         BSR     GETA
         BSR     CKWADR         CHECK WORD BOUNDRY ADDRESS
         MOVE.L  D0,A1          A1 = FROM END ADDRESS
```

```
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETNUMA         D0 = VALUE
         MOVE.L  D0,D7

         MOVE.L  D6,A0           A0 = FROM BEGIN ADDRESS
         BSR     P2PHY           DISPLAY ADDRESSES
         CMP.L   A0,A1
         BCS     SYNTAX          END ADDR TOO SMALL

         CMPI.L  #$10000,D7
         BCC     SYNTAX          WORD OVERFLOW

BFCMD11
         MOVE.W  D7,(A0)         STORE DATA
         MOVE.W  (A0)+,D1
         CMP.W   D7,D1           VERIFY DATA
         BNE     MM90            'DATA DID NOT STORE'

         CMP.L   A0,A1
         BCC     BFCMD11
         BRA     MACSBUG




*-----------------------------------------------------------------------
* File BM        BM (Block Move) Command                        11/27/81

*  BLOCK MOVE

BMCMD    LEA     SYNTAX(PC),A0
         BSR     FNEXTF
         BSR     GETA
         MOVE.L  D0,A3           A3 = FROM BEGIN ADDRESS

         BSR     FNEXTF
         BSR     GETA
         MOVE.L  D0,A1           A1 = FROM END ADDRESS

         BSR     FNEXTF
         BSR     GETA
         MOVE.L  D0,A2           A2 = TO BEGIN ADDRESS

         MOVE.L  A3,A0
         BSR     P2PHY           PRINT ADDRESSES (A0) & (A1)
         MOVE.L  A2,D0
         BSR     PPHY            PRINT TO ADDRESS

         MOVE.L  A1,D1
         SUB.L   A3,D1           D1 =  SIZE - 1
         BCS     SYNTAX          END LESS THAN BEGIN
         ADDQ.L  #1,D1           D1 = COUNT (SIZE)

         CMP.L   A3,A2
         BCC.S   BM122           MOVING LOW TO HIGH

BM112    MOVE.B  (A3)+,(A2)+     MOVING HIGH TO LOW
         SUBQ.L  #1,D1           COUNT
         BNE     BM112
         BRA.S   BM142

BM122    ADD.L   D1,A3           MOVING LOW TO HIGH
         ADD.L   D1,A2
BM132    MOVE.B  -(A3),-(A2)
         SUBQ.L  #1,D1
         BNE     BM132
BM142    BRA     MACSBUG




*-----------------------------------------------------------------------
* File BR        BR, GD, GT, TR, TT, PER Commands               12/04/81

*   ***T***   TRACE COMMAND

TCMD     DS      0              "T" Alias for "TR" command
TRCMD    LEA     TCMDHOT(PC),A0 IF NO PARAMTERS
         BSR     FNEXTF         FIND NEXT FIELD
```

```
         BSR     GETNUMA         FIND NUMBER OF INST TO TRACE
         BSR     CKADDR          INSURE 24 BITS OR LESS
         TST.L   D0
         BNE.S   TCMD15

TCMDHOT  DS      0               SPECIAL ENTRY FROM DECODE
         MOVEQ   #1,D0           ZERO; MAKE TRACE ONE
TCMD15   MOVE.L  D0,TRACECNT
         BRA.S   TRACE2


*
** TT ** "TRACE TILL" COMMAND
*

TTCMD    LEA     SYNTAX(PC),A0
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETA            GET ADDRESS
         BSR     CKWADR          CHECK WORD BOUNDRY ADDRESS

         MOVE.L  D0,D6
         BSR     PPHY            DISPLAY TILL ADDRESS

         MOVE.L  D6,BPTILL       9TH BP
         MOVE.L  #$FFFF,TRACECNT  SET FOR A VERY LONG TIME


TRACE2   MOVE.W  #-1,TRACEON     FOR DECODE OF NEXT COMMAND

         MOVE.L  REGPC,D0
         BSR     PPHY            DISPLAY START (PC) ADDRESS
         BRA     UNTRACE


*
*   ***GT***  RUN PROGRAM TO TEMP BREAKPOINT
*

GTCMD    LEA     SYNTAX(PC),A0  GET NUMBER ELSE PRINT "WHAT"
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETA            GET ADDRESS
         BSR     CKWADR          CHECK WORD BOUNDRY ADDRESS
         MOVE.L  D0,D6           D6 = UNTIL ADDRESS
         BSR     PPHY            PRINT PHYSICAL ADDRESS

* IF TILL ADDRESS IN BREAKPOINT TABLE; DISPLAY WHAT

         LEA     BPADD,A0        A0 = POINTER TO BP ADDRESSES
         MOVEQ   #8,D7           D7 = # OF ADDRESS ALLOWED
GT21     CMP.L   (A0)+,D6
         BEQ     BCMD55          ALREADY IN TABLE -SHOW TABLE-
         SUBQ.L  #1,D7
         BNE     GT21

         MOVE.L  D6,BPTILL       9TH BREAKPOINT
         BSR.S   GOCMD1A         NORMAL STARTUP
         BRA.S   GOCMD1

GOSET1   LEA     GOCMD1A(PC),A0 WHERE TO GO IF NO PARAMERS
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETA            GET ADDRESS
         BSR     CKWADR          CHECK WORD BOUNDRY ADDRESS
         MOVE.L  D0,REGPC

GOCMD1A  MOVE.L  REGPC,D0        (ALSO SUBROUTINE ENTRY)
         BSR     PPHY            PRINT ".PC" PHYSICAL ADDRESS
         RTS

GCMD     DS      0               "G" ALIAS FOR "GO"
GOCMD    BSR     GOSET1          "GO" (AFTER TRACING ONE INST)
GOCMD1   MOVE.L  #-1,TRACECNT    "FLAG" COUNTER AS SPECIAL

         BRA     UNTRACE

GDCMD    BSR     GOSET1          "GO DIRECT" Command
         BRA     UNSTACK


*   ***BR***  SET AND PRINT BREAKPOINTS
```

```
*                   DELETE ALL BREAKPOINTS

NOBRCMD  BSR.S   FIXBP          GET POINTERS
BCMD01   CLR.L   (A0)+          CLEAR THE ADDRESS TABLE
         SUBQ.L  #1,D7          DO IT 8 TIMES
         BNE     BCMD01
         BRA.S   BCMD7          DISPLAY EMPTY TABLE

FIXBP    LEA     BPADD,A0       SET UP ADDRESS & COUNTER
         MOVEQ   #8,D7          COUNTER
         LEA     BPCNT,A2       COUNTS
         RTS

************************************************************
* THIS COMMAND SUPPORTS THE "NO" OPTION. COMMANDS THAT ALLOW
* THE "NO" OPTION PROVIDE A HARD BRANCH 4 BYTES BEFORE THE
* REGULAR COMMAND.
************************************************************

         BRA.L   NOBR0          ENTRY FOR "NOBR" (IT LOOKS FOR PARMS OR NOT)
BRCMD    LEA     BCMD7(PC),A0   WHERE TO GO IF NO PARMS
BCMD0    BSR     FNEXTF         FIND NEXT FIELD
         BSR     GETA           GET ADDRESS
         BSR     CKWADR         CHECK WORD BOUNDRY ADDRESS

         BSR     FIXBP          SET UP COUNTER & ADDRESS
BCMD00   CMP.L   (A0),D0        SEE IF ALREADY IN TABLE
         BEQ.S   BCMD33         GO CHECK FOR COUNT
         ADDQ.L  #4,A2          BUMP OTHER POINTER
         ADDQ.L  #4,A0          BUMP MAIN POINTER
         SUBQ.L  #1,D7
         BNE     BCMD00

         BSR     FIXBP          GET ADDRESS & POINTERS
BCMD3    TST.L   (A0)           FIND AN EMPTY STOP
         BNE.S   BCMD5
         MOVE.L  D0,(A0)        PUT NEW ADDRESS IN TABLE
BCMD33   CLR.L   (A2)           CLEAR CURRENT COUNT
         MOVE.B  (A5),D1        CHECK INPUT FOR COUNT
         CMPI.B  #';',D1
         BNE.S   BCMD6          NO COUNT
         ADDQ.L  #1,A5          BUMP THE BUFFER SCANNER
         BSR     GETNUMA        GO GET THE COUNT
         MOVE.L  D0,(A2)        MOVE TO TABLE

BCMD6    LEA     BCMD7(PC),A0   WHERE TO GO IF NO MORE PARAMETERS
         BRA     BCMD0

BCMD5    ADDQ.L  #4,A0          BUMP TABLE POINTER
         ADDQ.L  #4,A2          BUMP POINTER TO COUNTS
         SUBQ.L  #1,D7          LOOP AROUND
         BNE     BCMD3

BCMD55   LEA     MSG008E(PC),A5 TABLE FULL; ERROR MESSAGE
         BSR     FIXDATA
         BRA.S   BCMD77

BCMD7    LEA     MSGBR(PC),A5   "BREAKPOINTS"
         BSR     FIXDCRLF
BCMD77   BSR     OUT1CR

         BSR     FIXBP          SET ADDRESS & COUNTER
BCMD8    MOVE.L  (A2)+,D6       D6 = COUNT
         MOVE.L  (A0),D0        D0 = ADDRESS
         BEQ.S   BCMD9          ZERO ADDRESS
         BSR     FRELADDR       FORMAT RELATIVE ADDRESS
         MOVE.B  #BLANK,(A6)+

         MOVE.L  (A0),D0        FORMAT ABSOLUTE ADDRESS
         BSR     PNT6HX
         MOVE.L  D6,D0
         BEQ.S   BCMD81         DON'T PRINT ZERO COUNT
         MOVE.B  #';',(A6)+
         BSR     PNTZHX         PRINT WITH ZERO SURPRESS

BCMD81   BSR     OUT1CR
BCMD9    ADDQ.L  #4,A0
         SUBQ.L  #1,D7          LOOP AROUND
```

```
         BNE     BCMD8
         BRA     MACSBUG

MSGBR    DC.B    'BREAKPOINTS',EOT



* NOBR COMMAND

NOBR0    LEA     NOBRCMD(PC),A0 WHERE TO GO IF NO PARAMETERS
NOBR1    BSR     FNEXTF         FIND NEXT FIELD
         BSR     GETA           GO DECODE NUMBER/ADDRESS
         TST.L   D0
         BEQ     BCMD55         ZERO NOT VALID BREAKPOINT
         BSR     CKWADR         CHECK WORD BOUNDRY ADDRESS

         LEA     BPADD,A0       SET UP TABLE POINTER
         MOVEQ   #8,D7          COUNTER
NOBR3    MOVE.L  (A0),D1        GET BREAKPOINT IN TABLE
         CMP.L   D1,D0          SEE IF SAME
         BEQ.S   NOBR4
         ADDQ.L  #4,A0
         SUBQ.L  #1,D7
         BNE     NOBR3
NOBR4    CLR.L   (A0)           CLEAR THIS BREAKPOINT
         LEA     BCMD7(PC),A0   WHERE TO GO IF NO PARAMETER
         BRA     NOBR1




*---------------------------------------------------------------------------
* File BS        BS (Block Search) Command                        11/27/81

*
*  BLOCK SEARCH   BS <ADDR1> <ADDR2> 'LITERAL STRING'
*                 BS <ADDR1> <ADDR2> <DATA> [<MASK>] [;<OPTION>]

BSCMD    LEA     SYNTAX(PC),A0
         BSR     FNEXTF
         BSR     GETA
         MOVE.L  D0,A3          A3 = FROM BEGIN ADDRESS

         BSR     FNEXTF
         BSR     GETA
         MOVE.L  D0,A1          A1 = FROM END ADDRESS

         BSR     FNEXTF
         MOVE.B  (A5),D0
         CMPI.B  #$27,D0
         BEQ     BS311          LITERAL STRING SEARCH

         BSR     GETNUMA
         MOVE.L  D0,D7          D7 = DATA WE ARE LOOKING FOR
         CLR.L   D5             SCAN FOR OPTIONS
         MOVE.L  A5,A0
BS91     CMP.L   A6,A0
         BCC.S   BS97           AT END OF BUFFER

         MOVE.B  (A0)+,D0
         CMPI.B  #';',D0
         BNE     BS91           NOT SEMICOLON

         MOVE.L  A0,A6
         SUBQ.L  #1,A6          ADJUST END OF BUFFER POINTER

         MOVE.B  (A0),D0        D0 = OPTION CHARACTER
         CMPI.B  #'B',D0
         BEQ.S   BS97           BYTE  D5 = 0

         MOVEQ   #1,D5
         CMPI.B  #'W',D0
         BEQ.S   BS97                WORD  D5 = +

         MOVEQ   #-1,D5         LONG WORD  D5 = -
         CMPI.B  #'L',D0
         BNE     SYNTAX
BS97     DS      0
```

```
         MOVEQ   #-1,D4          D4 = DEFAULT SEARCH MASK
         LEA     BS101(PC),A0
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETNUMA
         MOVE.L  D0,D4           D4 = MASK
BS101    DS      0

         MOVE.L  A3,A0
         BSR     P2PHY           PRINT ADDRESSES (A0) & (A1)

         CMP.L   A3,A1
         BCS     SYNTAX          END LESS THAN BEGIN

BS200    CMP.L   A1,A3
         BCC     MACSBUG         DONE

         BSR     FIXBUF
         MOVE.L  A3,D0
         BSR     PNT6HX          FORMAT ADDRESS
         MOVE.B  #BLANK,(A6)+    SPACE INTO BUFFER

         TST.L   D5
         BEQ.S   BS225           BYTE SIZE

         MOVE.L  A3,D0
         BSR     CKWADR          INSURE WORD BOUNDRY
         TST.L   D5
         BPL.S   BS219           WORD SIZE (2 BYTES)

         MOVEQ   #2,D6           D6 =  SIZE (LONG WORD)
         MOVE.L  (A3),D0         FETCH
         AND.L   D4,D0           MASK
         CMP.L   D7,D0           COMPARE
         BNE.S   BS215

         MOVE.L  (A3),D0         DISPLAY NON-MASKED DATA
         BSR     PNT8HX

BS213    BSR     OUT1CR

BS215    ADD.L   D6,A3           BUMP FETCH ADDRESS  (D6 = SIZE)
         BRA     BS200

BS219    MOVEQ   #2,D6           SIZE = WORD
         MOVE.W  (A3),D0
         AND.W   D4,D0
         CMP.W   D7,D0
         BNE     BS215

         MOVE.W  (A3),D0
         BSR     PNT4HX          DISPLAY
         BRA     BS213

BS225    MOVEQ   #1,D6           SIZE = BYTE
         MOVE.B  (A3),D0
         AND.B   D4,D0
         CMP.B   D7,D0
         BNE     BS215

         MOVE.B  (A3),D0
         BSR     PNT2HX          DISPLAY
         BRA     BS213

* LITERAL STRING SEARCH
*   SAVE STRING

BS311    LEA     DUMPTEMP,A2
BS313    ADDQ.L  #1,A5
         CMP.L   A6,A5
         BCC     SYNTAX          ADDR1 GREATER THAN ADDR2
         MOVE.B  (A5),(A2)+
         MOVE.B  (A5),D0
         CMPI.B  #$27,D0
         BNE     BS313
         SUBQ.L  #1,A2           A2 = END OF STRING + 1

         MOVE.L  A3,A0
         BSR     P2PHY           DISPLAY ADDRESSES
```

```
         CMP.L   A3,A1
         BCS     SYNTAX

* LOOK FOR STRING MATCH

BS323    LEA     DUMPTEMP,A4    A4 = BEGIN OF STRING
         MOVE.L  A3,A0          A0 = BEGIN OF MEMORY
BS325    MOVE.B  (A0),D0
         CMP.B   (A4)+,D0       DATA COMPARE
         BNE.S   BS365          MIS-MATCH

         CMP.L   A0,A1          ADDRESS CHECK
         BCS     MACSBUG        OUT OF MEMORY

         ADDQ.L  #1,A0
         CMP.L   A2,A4
         BCS     BS325          MORE STRING TO CHECK

         BSR     FIXBUF         HAVE MATCH
         MOVE.L  A3,D0          FORMAT ADDRESS
         BSR     PNT6HX
         MOVE.B  #BLANK,(A6)+   SPACE

         MOVE.B  #$27,(A6)+     LEADING QUOTE
         LEA     DUMPTEMP,A4    MOVE STRING TO BUFFER
BS355    MOVE.B  (A4)+,(A6)+
         CMP.L   A4,A2
         BCC     BS355

         BSR     OUT1CR         DISPLAY

BS365    ADDQ.L  #1,A3
         CMP.L   A3,A1
         BCS     MACSBUG        DONE
         BRA     BS323




*-------------------------------------------------------------------------
* File BT        BT (Block Test) command                          11/30/81

* BT   BLOCK OF MEMORY TEST

BTCMD    BSR     MTSETUP        PREPARE PARMS (FROM,TO/COUNT)
         BSR     RAMTEST        EXEC MEMORY TEST SUBROUTINE
         BEQ     MACSBUG        NO ERRORS, RETURN TO MACSBUG
         MOVE.L  D1,-(A7)       SAVE DATA READ
         MOVE.L  D0,-(A7)       SAVE DATA WRITTEN CAUSING ERROR

         BSR     FIXBUF         SET UP TO PRINT
         MOVE.L  #'FAIL',(A6)+
         MOVE.L  #'ED A',(A6)+
         MOVE.W  #'T ',(A6)+
         MOVE.L  A2,D0          ADDRESS WHERE IT FAILED
         BSR     PNT6HX

         MOVE.L  #'  WR',(A6)+
         MOVE.L  #'OTE=',(A6)+
         MOVE.L  (A7)+,D0       WHAT WAS WRITTEN
         BSR     PNT4HX

         MOVE.L  #'   R',(A6)+
         MOVE.L  #'EAD=',(A6)+
         MOVE.L  (A7)+,D0       WHAT WAS READ
         BSR     PNT4HX

         BRA     MSG            PRINT IT




*-------------------------------------------------------------------------
* File DC        DC (Data Conversion) Command                     12/10/81

*
*
*  ***DC***  NUMBER CONVERSIONS
*
```

```
DCCMD    DS      0               -DATA CONVERT COMMAND-
NUMCON0  LEA     SYNTAX(PC),A0   IF NO PARAMETERS
         BSR     FNEXTF          POINT TO NEXT PARAMETER
         BSR     GETA            EVALUATE EXPRESSION

         BSR     FIXBUF          SET UP FOR PRINT
         MOVE.L  D0,D7           D7=VALUE
         TST.L   D7              SEE IF NEGATIVE
         BPL.S   NUMCON1
         MOVE.B  #'$',(A6)+
         MOVE.L  D7,D0           MOVE NEGATIVE RESULT TO D0
         BSR     PNT8HX
         MOVE.B  #'=',(A6)+
NUMCON1  MOVE.L  D7,D0           RESTORE
         BPL.S   NUMCON2
         NEG.L   D0
         BMI.S   NUMCON2         SPECIAL CASE (-0)
         MOVE.B  #'-',(A6)+
NUMCON2  MOVE.B  #'$',(A6)+
         BSR     PNTZHX
         MOVE.B  #'=',(A6)+      NOW PRINT DECIMAL VALUE
         MOVE.L  D7,D0
         BPL.S   NUMCON3
         NEG.L   D0
         BMI.S   NUMCON3         SPECIAL CASE (-0)
         MOVE.B  #'-',(A6)+
NUMCON3  MOVE.B  #'&',(A6)+
         BSR     HEX2DEC         PUT VALUE IN BUFFER
         BRA     MSG             GO PRINT IT



*------------------------------------------------------------------------
* File DFDI      DF (Display registers) WITH disassembler         05/27/82

REGNAMES DC.L    'PCSR'          TABLE OF NAMES OF REGISTERS
         DC.L    'USSS'
         DC.W    '??'            END OF TABLE

DFCMD    DS      0               DF COMMAND  ENTRY
         BRA.S   TD07

TDISPLY  BSR     FIXBUF          PRINT TRACE DISPLAY SUBROUTINE
         TST.B   XONOFF+3
         BNE.S   TD25            SHORT DISPLAY

TD07     BSR     FIXBUF          PRINT PRELUDE
         CLR.L   OUTTO           FORCE DISPLAY TO OPERATORS CONSOLE
         MOVE.L  SCREEN1,(A6)+
         BEQ.S   TD09            SKIP PRELUDE
         BSR     OUTPUT
TD09

         LEA     REGNAMES(PC),A4 REGISTER NAMES
         LEA     REGPC,A2        REGISTER DATA
         BSR     FIXBUF          SET UP I/O BUFFER

TD1      MOVE.W  (A4)+,D0        GET REG NAME
         MOVE.L  (A2)+,D7        GET REG CONTENT
         CMPI.W  #'??',D0        SEE IF AT END OF REGS
         BNE.S   TD4
         BSR     OUT1CR          PRINT BUFFER

         MOVE.B  #'D',D7         CLASS=DATA
         LEA     REGS,A3         OFFSET
         BSR     PNTCLS          GO PRINT THE BLOCK

         MOVE.B  #'A',D7         CLASS=ADDRESS
         LEA     REGS+32,A3      OFFSET
         BSR     PNTCLS          GO PRINT THE BLOCK

         BSR     FIXBUF
         MOVEQ   #20,D1          LOOP COUNTER
TD27     MOVE.B  #'-',(A6)+      FILL BUFFER WITH BOARDER
         SUBQ.L  #1,D1
         BNE TD27

TD25     MOVE.L  REGPC,A4        DISASSEMBLE
```

```
         MOVEM.L (A4),D0-D2
         MOVE.L  A6,A5
         BSR     DCODE68K
         LEA     BUFFER,A5

         BSR     OUT1CR          PRINT

         BSR     FIXBUF          PRINT END STUFF FOR SCREEN CONTROL
         MOVE.L  SCREEN2,(A6)+
         BEQ.S   TD39            SKIP END STUFF
         BSR     OUTPUT
TD39     RTS     RETURN FOR MORE WORK

TD4      MOVE.W  D0,D2           PRINT REG NAME IN BUFFER
         ASR.W   #8,D0           MOVE IT OVER
         MOVE.B  D0,D3           SAVE REGISTER TYPE A,D,W,M ETC
         MOVE.B  D0,(A6)+        SAVE FIRST LETTER
         MOVE.B  D2,(A6)+        SAVE NEXT LETTER
         MOVE.B  #'=',(A6)+      EQUAL SIGN
         CMPI.W  #'US',D2
         BNE.S   T44
         MOVE.L  REGUS,D0        USER STACK
         BRA.S   T449

T44      CMPI.W  #'SS',D2        SUPER STACK IS SPECIAL
         BNE.S   T448
         MOVE.L  REGA7,D0        GET A7
         BRA.S   T449

T448     MOVE.L  D7,D0           REGISTER VALUE
         CMPI.W  #'SR',D2        SEE IF STATUS REGISTER
         BEQ.S   TDCC
T449     BSR     PNT8HX          FORMAT 8 HEX CHAR
TD9      MOVE.B  #BLANK,(A6)+    SPACE BETWEEN REGS
         BRA TD1

*   CONDITION CODE FORMAT
*
TDCC     MOVE.W  D0,D3           SAVE FOR A MOMENT
         BSR     PNT4HX
         MOVE.B  #'=',(A6)+
         MOVE.L  #$80000054,D7   TRACE BIT
         BSR.S   TDCC9
         MOVE.L  #$20000053,D7   SUPERVISOR BIT
         BSR.S   TDCC9
         MOVE.W  D3,D0           INTERRUPT LEVEL
         LSR.W   #8,D0
         ANDI.B  #$07,D0         "7" MAX IN SR FOR LEVEL
         BSR     PUTHEX
         MOVE.L  #$100058,D7     X BIT
         BSR.S   TDCC9
         MOVE.L  #$8004E,D7      N BIT
         BSR.S   TDCC9
         MOVE.L  #$4005A,D7      Z BIT
         BSR.S   TDCC9
         MOVE.L  #$20056,D7      V BIT
         BSR.S   TDCC9
         MOVE.L  #$10043,D7      C BIT
         BSR.S   TDCC9
         BRA     TD9
*
TDCC9    MOVE.L  D7,D6           DUPLICATE STUFF
         SWAP    D6              GET BIT POSITION
         AND.W   D3,D6           SEE IF ON
         BNE.S   TDCC91
         MOVE.B  #'.',D7         PUT IN PERIOD IF OFF
TDCC91   MOVE.B  D7,(A6)+        PUT IN LETTER IF ON
         RTS




*---------------------------------------------------------------------------
* File DUMP      DU     Dump "S-Records"                          05/10/82

*   ***DUMP***  DUMP  'S' RECORDS
*    FORMAT:  DU  ADDRESS1 ADDRESS2  [TEXT....]
*
DUCMD    DS      0
```

```
         BSR     SCANPORT        SEE WHERE TO SEND OUTPUT (DUMMY CALL)
         MOVE.L  OUTTO,D6        SAVE OUTTO FOR ACTUAL S-RECORD DUMP
         CLR.L   OUTTO           OVERRIDE SCANPORT CALL

         LEA     SYNTAX(PC),A0   WHERE TO GO IF NO PARAMTERS
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETA            GET ADDRESS1
         MOVE.L  D0,A3           A3 = BEGIN ADDRESS

         BSR     FNEXTF
         BSR     GETA            GET ADDRESS2
         MOVE.L  D0,A4           A4 = END ADDRESS

* PROCESS HEADER
         LEA     PUNCH5(PC),A0   WHERE TO GO IF NO HEADER
         BSR     FNEXTF          LOOK FOR HEADER
PUNCH5   DS      0

* MOVE TEXT TO 'TEMP STORAGE'
         LEA     DUMPTEMP,A2
         MOVE.L  A2,D5           D5 = START OF TEXT
PUM11    CMP.L   A5,A6
         BEQ.S   PUM13
         MOVE.B  (A5)+,(A2)+     MOVE
         BRA     PUM11
PUM13    EXG     A2,D5           D5 = END OF TEXT +1

* DISPLAY ADDRESSES
         MOVE.L  A3,A0
         MOVE.L  A4,A1
         BSR     P2PHY           DISPLAY TWO ADDRESSES

* INSURE END EQUAL GREATER THAN BEG
         CMP.L   A3,A4
         BCS     SYNTAX

         MOVE.L  D6,OUTTO        RESTORE OPTIONAL DESTINATION OF DUMP

* FINISH PROCESSING HEADER
*  A2 = START OF TEXT
         BSR     FIXBUF          A5,A6=#BUFFER
         MOVEQ   #2,D6           THE BYTE COUNT
         CLR.L   D4              CLEAR THE CHECKSUM
         MOVE.L  #'S0??',(A6)+   START OF S RECORD
         MOVE.L  #'0000',(A6)+   DUMMY ADDRESS
MORES0   CMP.L   D5,A2           SEE IF AT END OF TEXT
         BGE.S   ENDS0           WHERE TO GO WHEN ALL CHARACTERS USED
         ADDQ.L  #1,D6           ANOTHER BYTE

         MOVE.B  (A2)+,D0        GET ANOTHER BYTE OF TEXT

         ADD.L   D0,D4           FOR CHECKSUM
         BSR     PNT2HX          PUT IT IN BUFFER
         BRA     MORES0
ENDS0    DS      0

         BSR     PNTSREC         GO PRINT THE 'S' RECORD
         MOVE    A3,A2           A2 WILL SCAN BETWEEN A3-A4

* DO ANOTHER 'S' RECORD
MORESP   BSR     FIXBUF          A5,A6=#BUFFER
         CLR.L   D4              CLEAR CHECKSUM REGISTER
         MOVE.L  A3,D0           READY TO PRINT ADDRESS
         MOVE.L  A3,D1           GET READY TO AND ADDRESS
         MOVEQ   #$10,D3         MAXIMUM BYTES ON S REC LINE
         ADD.L   D3,D1           INSURE END OF LINE ADDRESS IS MAX
         ANDI.L  #$FF0000,D1     SEE IF 3 BYTE ADDRESS
         BNE.S   S2REC           WHERE TO GO IF 3 BYTES NEEDED
         MOVE.L  #'S1??',(A6)+   PUSH
         MOVE    A3,D0           SET UP TO PRINT 2 BYTE ADDRESS
         BSR     PNT4HX          PRINT 4 HEX CHAR ADDRESS
         MOVEQ   #2,D6           BYTE COUNT
         MOVE.W  A3,D0           FIX UP CHECKSUM
         ADD.B   D0,D4           LOW BYTE
         ASR     #8,D0           SHIFT IT OVER
         ADD.B   D0,D4           HIGH BYTE OF ADDRESS
         BRA.S   PNCA3           GO PUNCH A LINE

S2REC    MOVE.L  #'S2??',(A6)+   PUSH
```

```
         BSR     PNT6HX          PRINT 6 HEX CHAR ADDRESS
         MOVEQ   #3,D6           BYTE COUNT
         MOVE.L  A3,D0           FIX UP CHECKSUM
         ADD.B   D0,D4           LOW BYTE
         ASR     #8,D0           SHIFT IT OVER
         ADD.B   D0,D4           MIDDLE BYTE
         SWAP    D0              SET UP FOR HIGH BYTE
         ADD.B   D0,D4           ADD HIGH BYTE
PNCA3    CMP.L   A4,A3           SEE IF AT ENDING ADDRESS

* END OF FILE
         BLE.S   A3OUT           WHERE TO GO IF BELOW OR AT END ADDRESS
         BSR.S   PNTSRECX        END IT BY PRINTING LAST RECORD
         BSR     FIXBUF          A5,A6=#BUFFER
         CLR.L   D4              CLEAR THE CHECKSUM
         MOVE.L  #'S9??',(A6)+   MOVE TO PRINT BUFFER
         MOVE.L  #'0000',(A6)+   MOVE '0000' TO PRIT BUFFER
         MOVEQ   #2,D6           BYTE COUNT
         BSR.S   PNTSREC         PRINT 'S9' END-OF-FILE RECORD
         BRA     MACSBUG         REENTER MACSBUG

A3OUT    MOVE.B  (A3)+,D0        GRAB THE BYTE FROM MEMORY
         ADD.W   D0,D4           ADD TO CHECKSUM
         ADDQ.W  #1,D6           BUMP THE BYTE COUNT
         ADDQ.L  #1,A1           ADD TO COUNT OF BYTES PROCESSED
         BSR     PNT2HX          PUT THE HEX IN THE PRINT BUFFER
         SUBQ.L  #1,D3           COUNT DOWN CHAR TO GO IN LINE
         BNE     PNCA3
         BSR.S   PNTSREC         END OF LINE-PUNCH IT
         BRA     MORESP          GO FIX UP NEXT LINE

* FIX UP & PRINT THE 'S' RECORD/LINE

PNTSRECX CMP.W   #0,A1           SEE IF ANY CHAR MOVED
         BEQ.S   PNTSRTS         NO CHAR MOVED

PNTSREC  ADD     #1,D6           ONE MORE BYTE (CHECKSUM)
         ADD.W   D6,D4           ADD BYTE COUNT TO CHECKSUM
         NOT.B   D4              COMPLIMENT THE CHECKSUM
         MOVE.L  D4,D0           READY FOR PRINT-HEX
         BSR     PNT2HX          PUT CHECKSUM IN RECORD
         MOVE.L  A6,D7           SAVE FOR THE MOMENT
         MOVE.L  A5,A6           START OF BUFFER
         ADDQ.L  #2,A6           BYPASS RECORD TYPE (4 CHAR)
         MOVE.B  D6,D0           SET UP BYTE COUNT FOR PNTHEX ROUTINE
         BSR     PNT2HX          PUT THE BYTE COUNT IN THE PRINT BUFFER
         MOVE.L  D7,A6           RESTORE REAL END OF BUFFER
         BSR     OUT1CR          DO THE ACTUAL DISPLAY/PUNCH
         SUB.L   A1,A1           CLEAR COUNTER OF BYTES PROCESSED
PNTSRTS  RTS




*---------------------------------------------------------------------------
* File GETA      GET ADDRESS Subroutine                          12/01/81

* GET ADDRESS
*  ENTER (A5) POINTER START OF BUFFER
*        (A6) POINTER END OF BUFFER
*
*  RETURN:  D0 = ADDRESS

*        (A6) POINTER END OF BUFFER
*
*  RETURN:  D0 = ADDRESS


* FORMATS HANDLED:
*  1.  NUMBER        DEFAULTS TO HEX
*  2.  $NUMBER       HEX
*  3.  &NUMBER       DECIMAL
*  4.  (A@)
*  5.  NUMBER(A@)
*  6.  (A@,D@)
*  7.  NUMBER(A@,D@)
*  8.  [NUMBER]      MEMORY INDIRECT
*
*   FORMATS 1,2,3,8  ADD OFFSET R0 UNLESS R1 - R7 SPECIFIED
```

```
* WORK REGISTERS
*  D4    VALUE BEING BUILT
*
*  D5    FLAG REGISTER
*     = 8000XXXX  R@ GIVEN (GARO)
*     = XXXX80XX  [  GIVEN (GALB)
*     = XXXXXX80  (  GIVEN (GALP)
*
*  D6    FLAG REGISTER
*     = 8000XXXX  A@ GIVEN           (GAAVF)
*     = XXXX80XX  NEED PLUS OR MINUS (GANPM)
*     = XXXXXX2B  +  PLUS GIVEN      (GAPMS)
*             2D  -  MINUS GIVEN

GETA     MOVEM.L D4-D6/A0,-(A7) SAVE SOME REGISTERS
         CLR.L   D4             VALUE BEING BUILT
         CLR.L   D5             FLAG REG
         CLR.L   D6             FLAG REG

         MOVE.B  (A5)+,D0       GET BYTE
         CMP.L   A5,A6
         BCS     GAP191         END OF BUFFER

         CMPI.B  #'[',D0
         BNE.S   GAP113

*  [  SET INDIRECT

         ORI.W   #$8000,D5      SET LEFT BRACKET (GALB)

GAP111   MOVE.B  (A5)+,D0       GET BYTE
         CMP.L   A5,A6
         BCS     GAP191         END OF BUFFER
GAP113   DS      0

         CMPI.B  #'+',D0
         BEQ.S   GAP121         PLUS SIGN

         CMPI.B  #'-',D0
         BEQ.S   GAP121         MINUS SIGN

         CMPI.B  #']',D0
         BEQ.S   GAP131         RIGHT BRACKET (INDIRECT)

         CMPI.B  #'(',D0
         BEQ.S   GAP141         LEFT PARIN

         CMPI.B  #',',D0
         BEQ     GAP161         COMMA

         CMPI.B  #')',D0
         BEQ     GAP181         RIGHT PARIN

         CMPI.B  #';',D0        "SEMI-COLON"
         BEQ     GAP191         TERMINATE

         CMPI.B  #BLANK,D0      "SPACE"
         BEQ     GAP191         TERMINATE

         TST.W   D6             (GANPM)
         BMI.S   GAE            NEEDS PLUS OR MINUS

         CMPI.B  #'R',D0
         BEQ     GAP171         RELATIVE OFFSET

         TST.B   D6             (GALP)
         BMI.S   GAE            (...   NUMBER NOT ALLOWED

* NONE OF ABOVE ASSUME NUMERIC VALUE
         SUBQ.L  #1,A5          ADJUST (A5) TO FIRST CHAR
         CLR.L   D0
         BSR     GETNUMA

         CMPI.B  #'-',D6        (GAPMS)
         BEQ.S   GAP118         MINUS SIGN

         ADD.L   D0,D4          PLUS SIGN
```

```
         BRA.S   GAP119

GAP118   SUB.L   D0,D4

GAP119   CLR.B   D6             (GAPMS)  RESET PLUS MINUS FLAG
         ORI.W   #$8000,D6      (GANPM)  SET NEED PLUS MINUS
GAP111S  BRA     GAP111

* (*) (-) SET ARITHMETIC OPERATOR

GAP121   TST.B   D6             (GAPMS)
         BNE.S   GAE            MULTI OPERATORS
         MOVE.B  D0,D6          (GAPMS)
         ANDI.W  #$00FF,D6      RESET (GANPM) NEED PLUS MINUS
         BRA     GAP111S

*  ]  CLOSE INDIRECT

GAP131   TST.W   D5             (GALB)
         BPL.S   GAE            [ MISSING

* IF NO R@ GIVEN ADD R0
         TST.L   D5             (GARO)
         BMI.S   GAP135         R@ GIVEN
         ADD.L   OFFSET,D4      NO R@ GIVEN; ADD R0
GAP135   DS      0

         MOVE.L  D4,A0
         MOVE.L  (A0),D0
         BRA     GAP199

*  (  DO PARIN PAIR

GAP141   TST.B   D5             (GALP)
         BMI.S   GAE            MULTI (
         TST.L   D5             (GARO)
         BMI.S   GAE            R@ NOT ALLOWED WITH (..)
         ORI.B   #$80,D5        (GALP) SET LEFT PAREN

* LEFT PARIN SET; MUST BE A@ NEXT
         MOVE.B  (A5)+,D0       GET BYTE
         CMP.L   A5,A6
         BCS.S   GAE
         CMPI.B  #'A',D0
         BNE.S   GAE            NOT A-REG
         LEA     REGS+32,A0
         BSR.S   GASRGN         GET VALUE IN A@
         ADD.L   D1,D4
         ORI.L   #$80000000,D6  (GAAVF) A-REG VALUE FLAG
         BRA     GAP111S

GAE      BRA     SYNTAX

*  COMMA  A-REG or
*  COMMA  D-REG REQUIRED

GAP161   TST.L   D6             (GAAVF)
         BPL     GAE            NO A-REG SPECIFIED
         MOVE.B  (A5)+,D0       GET BYTE
         CMP.L   A5,A6
         BCS     GAE
         CMPI.B  #'A',D0
         BNE.S   GAP163
         LEA     REGS+32,A0     GET VALUE IN A@
         BRA.S   GAP165
GAP163   CMPI.B  #'D',D0
         BNE     GAE            NOT D-REG
         LEA     REGS,A0        GET VALUE IN D@
GAP165   BSR.S   GASRGN
         ADD.L   D1,D4
         BRA     GAP111S

*  R@  OFFSET

GAP171   CMPI.B  #'+',D6        (GAPMS)
         BNE     GAE            ONLY + ALLOWED

* ONLY ONE R@ ALLOWED
```

```
         TST.L   D5             (GARO)
         BMI     GAE            MULIT R@
         ORI.L   #$80000000,D5  SET R@ GIVEN (GARO)

         LEA     OFFSET,A0
         BSR.S   GASRGN         GET VALUE IN R@
         ADD.L   D1,D4
         BRA     GAP119

*  )  CLOSE THE WORLD

GAP181   TST.L   D6             (GAAVF)
         BPL     GAE            NO (
         BRA.S   GAP197

*  SPACE  TERMINATOR

GAP191   TST.W   D5             (GALB)
         BMI     GAE            [ WITHOUT ]
         TST.B   D5             (GALP)
         BMI     GAE            ( WITHOUT )
         SUBQ.L  #1,A5          ADJUST CHAR POINTER

* IF NO R@ GIVEN ADD R0

         TST.L   D5             (GARO)
         BMI.S   GAP197         R@ GIVEN
         ADD.L   OFFSET,D4
GAP197   MOVE.L  D4,D0

GAP199   MOVEM.L (A7)+,D4-D6/A0
         RTS

* GET NEXT NUMBER
*  A0 = POINTER TO TABLE OF VALUES
*  D1 = VALUE ON RETURN

GASRGN   CLR.L   D0
         MOVE.B  (A5)+,D0       GET BYTE
         CMP.L   A5,A6
         BCS     GAE
         SUBI.B  #'0',D0        ADJUST TO ZERO
         CMPI.B  #7,D0
         BHI     GAE            NOT 0 - 7
         MULU    #4,D0          4 * OFFSET
         MOVE.L  (A0,D0),D1
         RTS




*-------------------------------------------------------------------------
* File H         Register save,Trace, Breakpoint                 03/03/82

*       SAVE ALL REGISTERS ROUTINE-JMP [TEMP] BACK
*          USUALLY CALLED BY THE MACRO "SAVEREGS"

SAVE     DS      0
         LEA     REGA7,A7       WHERE TO START STORING
         MOVEM.L D0-D7/A0-A6,-(A7)  SAVE REGISTERS

         LEA     SYSTACK+4,A7   SET UP STRETCHED VERSAUG STACK

         MOVE.L  REGA7,A0       A0 = TARGETS SUPERVISOR STACK POINTER
         MOVE.W  (A0)+,REGSR+2  GET TARGETS SR

         MOVE.L  (A0)+,REGPC    GET TARGETS PC

         MOVE.L  A0,REGA7       WHERE TARGET STACK REALLY POINTS

         MOVE.L  USP,A1         GET USERS STACK POINTER
         MOVE.L  A1,REGUS       SAVE IT FOR DISPLAY ETC

         MOVE.L  REGPC,D0       GET PROGRAM COUNTER
         SUBQ.L  #4,A7          SET REAL MACSBUG STACK
         MOVE.L  TEMP,-(A7)     PUT RETURN ADDRESS ON STACK
         RTS                    RETURN TO SENDER

*
```

```
*     ***TRACE***  TRACE ENTRY POINT
*       ENTER FROM VECTOR 9  (LOCATION 24) FOR
*

* REGISTER USAGE
*  A0 = ADDRESS OF BREAKPOINT ADDRESS
*  A2 = ADDRESS OF BREAKPOINT COUNTS
*  D1 = PC
*  D5 = TRACECNT
*  D7 = BKPT COUNT

TRACE    MOVE.W  #$2700,SR      MASK OFF INTERRUPTS
         SAVEREGS

* IF PC POINTS TO 'TRACE'; DOUBLE EVENT OCCURED
*  CLEAR LAST EVENT BY IGNORING

         CMPI.L  #V2,REGPC

         BNE.S   TRACE16

         MOVE.L  REGA7,A5       TRIM LAST EVENT FROM STACK
         MOVE.W  (A5)+,REGSR+2  *MOVE VALUES FROM STACK TO
         MOVE.L  (A5)+,REGPC    *PSUEDO SR, PC,
         MOVE.L  A5,REGA7       *REFLECT ADJUSTMENTS IN PSUEDO STACK
TRACE16  ANDI.W  #$7FFF,REGSR+2 RESET "T" (TRACE) BIT


         MOVE.L  TRACECNT,D5
         BMI.S   TRACE39        EXECUTING ONE-INSTRUCTION
         BEQ     ABORT335       NOT TRACEING

*   MAKE SURE WE ARE NOT TRACING OURSELVES
         MOVE.L  REGPC,D1       GET PC
         LEA.L   FIRST(PC),A0
         CMP.L   A0,D1
         BMI.S   TISOK          TRACING BELOW OURSELVES
         LEA     LAST(PC),A0
         CMP.L   A0,D1
         BHI.S   TISOK          TRACING ABOVE OURSELVES
         LEA     MSG020(PC),A5  TRACING OURSELVES
         BRA     CHKBP4
TISOK

         LEA     BPADD,A0
         LEA     BPCNT,A2
         MOVEQ   #9,D7          9TH BP IS "UNTILL" FEATURE

TRACE01  CMP.L   (A0),D1        SEE IF PC MATCHES ADDRESS IN TABLE
         BNE.S   TRACE08
         MOVE.L  (A2),D0        GET COUNT
         BEQ     CHKBP3         COUNT ZERO
         SUBQ.L  #1,D0          COUNT DOWN
         MOVE.L  D0,(A2)        SAVE IT
         BEQ     CHKBP3         COUNT WENT TO ZERO
         BRA.S   TRACE03

TRACE08  ADDQ.L  #4,A0          BUMP TABLE POINTER
         ADDQ.L  #4,A2          BUMP COUNT TABLE POINTER
         SUBQ.L  #1,D7          LOOP AROUND
         BNE     TRACE01

TRACE03  BSR     TDISPLY        DO TRACE DISPLAY
         SUBQ.L  #1,TRACECNT    TRACE COUNT
         BEQ     MACSBUG        STOP WHEN ZERO
         BRA.S   UNTRACE        CONTINUE WITH TRACE


* END UP HERE AFTER BREAKPOINTING ONE INSTRUCTION
* -- PUT BP BACK IN AND CONTINUE TO RUN

TRACE39  CLR.L   TRACECNT
         BSR.S   SWAPIN         PUT BP INTO USER'S MEMORY
         BRA.S   UNSTACK        CONTINUE TO RUN


UNTRACE  ORI.W   #$8000,REGSR+2 SET UP TRACE BIT!
         ADDR2MEM TRACE,AV9     TAKE TRACE VECTOR
```

```
UNSTACK  MOVE.L  REGUS,A1
         MOVE.L  A1,USP          US = TARGET'S USER STACK
         MOVE.L  REGPC,A2        A2 = TARGET'S PC


         DS      0               INSURE MEMORY AT LOCATION OF PC
         MOVE.W  (A2),D0         * ADDR TRAP ERROR * IF NO MEMORY

         MOVE.L  REGA7,A1        A1 = TARGET SYSTEM STACK (SS)

         DS      0               INSURE MEMORY AT TARGET'S STACK
         MOVE.L  A2,-(A1)        MOVE PC ONTO TARGET'S STACK

         MOVE.W  REGSR+2,D0
         MOVE.W  D0,-(A1)        SR ONTO TARGET'S STACK

         MOVE.L  A1,TEMP         TEMP = TARGETS SS

         LEA     REGS,A7         A7 = POINTER TO PSUEDO REGISTERS
         MOVEM.L (A7)+,D0-D7/A0-A6  ARM MOST OF TARGET REGISTERS
         MOVE.L  TEMP,A7         SS = TARGET'S

         RTE                     GO BACK TO THE USER


SWAPIN   BSR.S   SWAPOUT         MAKE SURE THEY ARE ALL OUT

         MOVE.L  AV4,SAVEAV4     SAVE VECTOR (WHOM EVER'S IT WAS)
         LEA     CHKBP(PC),A6
         MOVE.L  A6,AV4          REPLACE IT WITH THE "CHKBP" RTN
         LEA     SWAPIN1(PC),A6 A6 = ROUTINE ADDRESS
         BRA.S   SWAP

SWAPOUT  TST.W   BPSTATUS         SEE IF OUT ALREADY
         BEQ.S   SWAPEND          YES...DONE
         MOVE.L  SAVEAV4,AV4      NO....REPLACE THE VECTOR

         LEA     SWAPOUT1(PC),A6 A6 = ROUTINE ADDRESS
SWAP     LEA     BPADD,A0        A0 = ADDRESS OF TABLE
         LEA     BPCNT,A2        A2 = ADDRESS OF COUNTS
         MOVEQ   #9,D7           DO 9 BP
         LEA     BPDATA,A3       CONTENT TABLE

SWAP1    MOVE.L  (A0),A4         GET POSSIBLE ADDDRESS
         TST.L   (A0)            IS POSSIBLE ADDRESS ZERO
         BEQ.S   SWAP99          ZERO MEANS NO BP DEFINED
         JMP     (A6)            GO TO RIGHT ROUTINE

SWAPIN1
         MOVE.W  (A4),D0         USER'S PROGRAM GOES INTO TABLE

         MOVE.W  D0,(A3)
         MOVE.W  #BKPOINT,(A4)   PUT BREAKPOINT IN

         MOVE.W  #1,BPSTATUS     FLAG AS BP IN
         BRA.S   SWAP99

SWAPOUT1 MOVE.W  (A3),D0         GET CONTENTS FROM TABLE
         MOVE.W  D0,(A4)         PUT CONTENTS BACK INTO PROGRAM

         CLR.W   BPSTATUS        FLAG AS BREAKPOINTS OUT
SWAP99   ADDQ.L  #4,A0           BUMP ADDRESS TABLE POINTER
         ADDQ.L  #2,A3           BUMP CONTENT TABLE POINTER
         SUBQ.L  #1,D7
         BNE     SWAP1
SWAPEND  RTS

*  ILLEGAL INSTRUCTION ENTRY POINT
*   BREAKPOINT TEST
*
CHKBP    MOVE.W  #$2700,SR       MASK OF INT.
         SAVEREGS
         MOVE.L  REGPC,D0        GET PROGRAM COUNTER TO COMPARE
         MOVE.L  D0,A5           A5 = TARGET PC
         MOVE.W  (A5),D1         SEE WHAT OP CODE WAS XEQ

         BSR     SWAPOUT         TAKE BREAKPOINTS OUT

         CMPI.W  #BKPOINT,D1     USED FOR BP
```

```
         BNE.S   CHKBP11        NOT AT BREAKPOINT

         LEA     BPADD,A0       A0 = ADDRESS OF BP ADDRESSES
         LEA     BPCNT,A2       A2 = ADDRESS OF BP COUNTS
         MOVEQ   #9,D7          D7 = LOOP COUNT

CHKBP1   CMP.L   (A0),A5        SEE IF WE ARE THERE
         BEQ.S   CHKBP2         AT BREAKPOINT; AT AN ADDRESS

         ADDQ.L  #4,A0          BUMP BOTH POINTERS
         ADDQ.L  #4,A2
         SUBQ.L  #1,D7
         BNE     CHKBP1

* IS NOT A BREAKPOINT; ASSUME ILLEGAL INSTRUCTION
CHKBP11  LEA     MSG009(PC),A5  'ILLEGAL INSTRUCTION'
         CLR.W   TRACEON        RESET TRACE MODE
         BRA.S   CHKBP4

* AT BREAKPOINT...CHECK COUNT
CHKBP2   MOVE.L  (A2),D0        GET COUNT
         BEQ.S   CHKBP3         COUNT ZERO
         SUBQ.L  #1,D0          DECREMENT
         MOVE.L  D0,(A2)        PUT COUNT BACK
         BNE     GOCMD1         COUNT NOT ZERO; KEEP GOING

CHKBP3   LEA     MSG014(PC),A5  "AT BREAKPOINT"

CHKBP4   BSR     FIXDCRLF
         BSR     OUT1CR
         BSR     TDISPLY        PRINT TRACE DISPLAY
         BRA     MACSBUG

MSG009   DC.B    'ILLEGAL INSTRUCTION',EOT



MSG014   DC.B    'AT BREAKPOINT',EOT


MSG020   DC.B    '.PC within "DEBUGGER"',BELL,EOT




         DC.B    0              PAD BYTE



*-------------------------------------------------------------------------
* File HE        HELP Command                                    12/01/81

* HELP  COMMAND
*  PRINT: REGISTERS AND COMMANDS IN TABLES

HECMD    LEA     MSG002(PC),A5
         BSR     FIXDATA
         BSR     OUT1CR         PRINT

         LEA     SOLIST(PC),A4  A4 = POINTER TO COMMAND LIST

         BSR     FIXBUF
         MOVE.B  #8,D2          D2 = # CMDS PER LINE
HELP4    MOVE.W  (A4)+,D1       GET 2 BYTE COMMAND
         CMPI.W  #$FFFF,D1
         BEQ.S   HELP6          DONE PRINT LAST BUFFER
         ADDQ.L  #2,A4          BUMP POINTER UP BY 2
         TST.W   D1             IS THE INVISIBLE INDICATOR ON?
         BMI     HELP4          YES... THEN BYPASS THIS ONE
         MOVE.W  D1,D3          SAVE XX FOR "XX" AND "NOXX" IN HELP
         ANDI.W  #$7F7F,D1      REMOVE CONTROL BITS
         MOVE.W  D1,(A6)+       NO.... MOVE THIS COMMAND TO "PRINT" LINE
         MOVE.L  #'    ',(A6)+  MOVE BLANKS FOR SPACING
         BSR.S   HELP81         PRINT THE LINE IF FULL
         BTST    #7,D3          IS "NO" OPTION SUPPORTED?
```

```
         BEQ.S   EOHLOOP         NO...BYPASS THIS COMMAND, ELSE...
         MOVE.W  #'NO',(A6)+     "NO  "   IN MSG
         ANDI.W  #$7F7F,D3       RESET CONTROL BITS
         MOVE.W  D3,(A6)+        "NOCC"   IN MSG (WHERE CC=COMMAND CODE)
         MOVE.W  #'  ',(A6)+     "NOCC  " IN MSG    "    "    "      "
         BSR.S   HELP81          PRINT THE LINE IF FULL
EOHLOOP  BRA     HELP4           NEXT COMMAND.

*        DISPLAY LINE OF COMMANDS (IF FULL)

HELP81   SUBQ.B  #1,D2           D2 = COUNT OF COMMANDS PER LINE
         BNE.S   HELP85          NOT FULL, JUST RETURN FOR MORE
         BSR     OUT1CR          YES, THEN WRITE IT
         BSR     FIXBUF          SET-UP LINE FOR NEXT COMMANDS
         MOVE.B  #8,D2           D2 = MAX NUMBER CAN BE PLACED IN LINE
HELP85   RTS

HELP6    BSR     OUT1CR          PRINT ANY REMAINING PORTION
*        BRA     HELP1           SEE IF COMPLETE
         BRA     MACSBUG         HELP IS COMPLETE SO RETURN

MSG002   DC.B    '.PC .SR .US .SS',CR,LF



         DC.B    '.D0 .D1 .D2 .D3 .D4 .D5 .D6 .D7',CR,LF




         DC.B    '.A0 .A1 .A2 .A3 .A4 .A5 .A6 .A7',CR,LF




         DC.B    '.R0 .R1 .R2 .R3 .R4 .R5 .R6',CR,LF,EOT




         DC.B    0               PAD BYTE



*-------------------------------------------------------------------------
* File HEX2DEC   HEX2DEC convert hex to decimal                  11/02/81


*    CONVERT BINARY TO DECIMAL  REG D0 PUT IN (A6) BUFFER AS ASCII

HEX2DEC  MOVEM.L D1-D4/D6-D7,-(A7)   SAVE REGISTERS
         MOVE.L  D0,D7           SAVE IT HERE
         BPL.S   HX2DC
         NEG.L   D7              CHANGE TO POSITIVE
         BMI.S   HX2DC57         SPECIAL CASE (-0)
         MOVE.B  #'-',(A6)+      PUT IN NEG SIGN
HX2DC    CLR.W   D4              FOR ZERO SURPRESS
         MOVEQ   #10,D6          COUNTER
HX2DC0   MOVEQ   #1,D2           VALUE TO SUB
         MOVE.L  D6,D1           COUNTER
         SUBQ.L  #1,D1           ADJUST - FORM POWER OF TEN
         BEQ.S   HX2DC2          IF POWER IS ZERO
HX2DC1   MOVE.W  D2,D3           D3=LOWER WORD
         MULU    #10,D3
         SWAP    D2              D2=UPPER WORD
         MULU    #10,D2
         SWAP    D3              ADD UPPER TO UPPER
         ADD.W   D3,D2
         SWAP    D2              PUT UPPER IN UPPER
         SWAP    D3              PUT LOWER IN LOWER
         MOVE.W  D3,D2           D2=UPPER & LOWER
         SUBQ.L  #1,D1
         BNE     HX2DC1
```

```
HX2DC2   CLR.L   D0             HOLDS SUB AMT
HX2DC22  CMP.L   D2,D7
         BLT.S   HX2DC3         IF NO MORE SUB POSSIBLE
         ADDQ.L  #1,D0          BUMP SUBS
         SUB.L   D2,D7          COUNT DOWN BY POWERS OF TEN
         BRA.S   HX2DC22        DO MORE
HX2DC3   TST.B   D0             ANY VALUE?
         BNE.S   HX2DC4
         TST.W   D4             ZERO SURPRESS
         BEQ.S   HX2DC5
HX2DC4   ADDI.B  #$30,D0        BINARY TO ASCII
         MOVE.B  D0,(A6)+       PUT IN BUFFER
         MOVE.B  D0,D4          MARK AS NON ZERO SURPRESS
HX2DC5   SUBQ.L  #1,D6          NEXT POWER
         BNE     HX2DC0
         TST.W   D4             SEE IF ANYTHING PRINTED
         BNE.S   HX2DC6
HX2DC57  MOVE.B  #'0',(A6)+     PRINT AT LEST A ZERO
HX2DC6   MOVEM.L (A7)+,D1-D4/D6-D7   RESTORE REGISTERS
         RTS                    END OF ROUTINE



*-------------------------------------------------------------------------
* File LOAD      LO & VE (Load & Verify) Commands.               02/22/82

* THIS FUNCTION
*   1.) READS A "S-RECORD"
*   2.) FORMATS THE DATA ON THE STACK
*   3.) CALCULATES THE CHECKSUM
*   4.) THEN STORES (VERIFIES) THE DATA TO MEMORY
*
* WHEN VERIFYING RECORDS MAKE BE SKIPPED WHILE DISPLAYING THE MISS-MATCHED
*  RECORDS.  DISPLAY FORMAT
*   S1CCAAAA.-.-.-33.-
*     CC                    BYTE COUNT
*       AAAA                ADDRESS
*           .-.-.-  .-      DATA FIELDS THAT MATCHED
*                 33        DATA MISS MATCH (DISPLAY DATA FROM S-RECORD)

*   ***LOAD*** AND ***VERIFY***  'S' RECORDS
*
*        D4 = ERROR FLAG
*
*        D5 = V000I0CC
*             0.......  = LOAD
*             8.......  = VERIFY
*             ....0...  = CALCULATE CHECKSUM
*             ....8...  = IGNORE CHECKSUM
*             ......CC  = CHECKSUM

VECMD    MOVEQ   #1,D5          MARK THE MODE
         ROR.L   #1,D5          D5 = $80000000
         BRA.S   CHKCHKS        GO CHECK CHECKSUM OPTION

LOCMD    CLR.L   D5             READ MODE
CHKCHKS  CLR.L   D4             RESET ERROR FLAG
         BSR     SCANPORT       SET UP OUTPUT P1,P2,P3 ETC

* SEE IF CHECKSUM -C OPTION AND =SEND THIS OPTION

READ01   CMP.L   A6,A5          SEE IF AT END OF BUFFER
         BCC.S   READ09
         MOVE.B  (A5)+,D0       GET A CHARACTER
READ03   CMPI.B  #'=',D0        SEE IF EQUAL SIGN
         BEQ.S   READ08

         CMPI.B  #'X',D0        SEE IF ECHO
         BNE.S   READ021
         MOVE.B  #-1,D5         D5.B = ECHO TO PORT ONE
         BRA     READ01

READ021  CMPI.B  #'-',D0        SEE IF MINUS SIGN
         BNE.S   READ01
         CMP.L   A6,A5          SEE IF AT END OF BUFFER
         BEQ.S   READ09
         MOVE.B  (A5)+,D0       GRAB SECOND CHARACTER
         CMPI.B  #'C',D0        SEE IF LETTER C AS IN -C
```

```
         BNE     READ03
         ORI.W   #$8000,D5      MARK AS IGNORE CHECKSUM
         BRA     READ01

READS1   CLR.L   D6             D6 = TYPE "S1"
         SUBQ.L  #4,D3          BYTE COUNT

         CLR.L   D0
         BSR     READHEX4       FORM ADDRESS
         BRA.S   READS202

READ08   BSR     OUTPUT2        SEND REST OF LINE(+CR) TO PORT

READ09   MOVE.B  D5,ECHOPT1     MOVE ECHO FLAG
         MOVE.L  OUTPORT1,OUTTO SEND OUTPUT TO CONSOLE


READ0    LEA     SYSTACK,A7     FORCE STACK (FOR ERROR RECOVERY)

         LINK    A4,#-((BUFFSIZE/2)+4)  CREATE BUFFER ON STACK

         CLR.B   D5             ZERO CHECKSUM

         BSR     FIXBUF         START OF INPUT BUFFER
         BSR     PORTIN2        GET A RECORD FROM PORT

         MOVE.L  A5,A3 SAVE     START ADDRESS OF BUFFER
READ00   MOVE.B  (A3)+,D0       GET FIRST CHARACTER
         CMP.L   A3,A6
         BCS     READ0          END OF BUFFER WITHOUT "S"

READ005  CMPI.B  #'S',D0        SEE IF IT IS AN S
         BNE     READ00         GET ANOTHER CHARACTER
         BSR     GETCHR         GET RECORD TYPE
         MOVE.L  D0,D6

         BSR     READHEX        GET CHAR COUNT
         CLR.L   D3
         MOVE.B  D0,D3

         CMPI.B  #'0',D6        'S0'???
         BEQ     READ0          JUST IGNORE
         CMPI.B  #'1',D6
         BEQ.S   READS1         S1 RECORD TYPE (2 BYTE ADDRESS)
         CMPI.B  #'8',D6
         BEQ     READS8         S8 RECORD TYPE
         CMPI.B  #'9',D6
         BEQ     READS9         S9 RECORD TYPE
         CMPI.B  #'2',D6
         BNE.S   READ005        KEEP LOOKING FOR "Sn"

READS2   MOVEQ   #2,D6          D6 = TYPE = S2 (3 BYTE ADDRESS)
         BSR     READHEX6
         SUBQ.L  #5,D3          BYTE COUNT
READS202 MOVE.L  D3,CC(A4)

         MOVE.L  D0,ADDRESS(A4) ADDRESS + OFFSET
         LEA     LDATA(A4),A2   A2 = STORE TO STACK POINTER

READ100  BSR.S   READHEX        GET DATA BYTE
         MOVE.B  D0,(A2)+       MOVE DATA TO STACK
         DBRA    D3,READ100

         TST.W   D5
         BMI.S   READ120        IGNORE CHECKSUM

         MOVE.B  D5,D7
         BSR.S   READHEX        GET CHECKSUM FROM DATA
         NOT.B   D7             CALCULATED CHECKSUM
         CMP.B   D7,D0
         BNE.S   READCKSM       ERROR
READ120


* STORE DATA (VERIFY) TO MEMORY

         MOVE.L  CC(A4),D3      BYTE COUNT
         MOVE.L  ADDRESS(A4),A3 MEMORY STORE ADDRESS
         LEA     LDATA(A4),A2   DATA ADDRESS ON STACK
```

```
         TST.L   D5
         BMI.S   READ400        VERIFY

READ130  MOVE.B  (A2),(A3)      STORE DATA
         CMP.B   (A2)+,(A3)+    VERIFY DATA STORED
         BNE     SETME          DATA DID NOT STORE
         DBRA    D3,READ130
READ135  BRA     READ0


***     VERIFY

READ400  LEA     8(A5),A1       A1 = PTR TO INPUT STRING
         ADD.L   D6,A1

         CLR.L   D7             D7 = MISS-MATCH FLAG

READ410  CMP.B   (A2)+,(A3)+    VERIFY DATA
         BNE.S   READ440        MISS-MATCH
         MOVE.B  #'.',(A1)+     OVERLAY INPUT STRING IF OK
         MOVE.B  #'-',(A1)+
READ420  DBRA    D3,READ410

         TST.L   D7
         BEQ     READ135        RECORD OK

         MOVE.L  A1,A6          DONOT DISPLAY CHECKSUM
READ430  BSR     OUT1CR         DISPLAY IT
         BRA     READ135        READ NEXT RECORD

READ440  MOVEQ   #-1,D7         D7 = Set miss-match flag
         MOVEQ   #-1,D4         D4 = Set Summary error flag
         ADDQ.L  #2,A1
         BRA     READ420

READCKSM MOVEQ   #-1,D4         D4 = Set summary error flag
         LEA     MSGLOAD2(PC),A5
         BSR     FIXDADD
         MOVE.L  D7,D0
         BSR     PNT2HX         CALCULATED CHECKSUM
         BRA     READ430

READHEX  BSR.S   GETHEXC        FORM BYTE
         ASL.B   #4,D0
         MOVE.B  D0,D1
         BSR.S   GETHEXC
         OR.B    D1,D0          D0 = BYTE FORMED
         ADD.B   D0,D5          UPDATE CHECKSUM
         RTS

GETCHR   MOVE.B  (A3)+,D0
         CMP.L   A3,A6
         BCS     READCKSM       OVERFLOW
         RTS

GETHEXC  BSR     GETCHR

         SUBI.B  #$30,D0        SEE IF LESS THAN ZERO
         BLT.S   RHEX3
         CMPI.B  #$09,D0        SEE IF GT 9
         BLE.S   RHEX2
         SUBQ.B  #7,D0          NORMALIZE $A TO 10
         CMPI.B  #$10,D0        SEE IF TOO LARGE
         BCC.S   RHEX3
RHEX2    RTS

RHEX3    LEA     MSGLOAD1(PC),A5 'NOT HEX=X?'  MESSAGE
         BSR     FIXDADD
         MOVE.B  -(A3),(A6)+    BAD CHARACTER
         BRA     READ430        GO TRY NEXT RECORD

MSGLOAD1 DC.B    'NOT HEX=',EOT


MSGLOAD2 DC.B    ' CHKSUM=',EOT


READHEX6 CLR.L   D0             FORM ADDRESS (3 BYTE)
```

```
         BSR     READHEX
         ASL.L   #8,D0
READHEX4 BSR     READHEX        FORM ADDRESS (2 BYTE)
         ASL.L   #8,D0
         BSR     READHEX
         ADD.L   OFFSET,D0
         RTS

READS8   BSR     READHEX6
READS800 MOVE.L  D0,REGPC       SAVE IT IN THE USER PREG

         TST.L   D4
         BNE     ERROR          DISPLAY "ERROR"
         BRA     MACSBUG        END OF ROUTINE

READS9   CLR.L   D0
         BSR     READHEX4       GET ADDRESS
         BRA     READS800


*---------------------------------------------------------------------------
* File MDDI      MD[S] (Memory Display) Command                  06/16/82

*   ***MD***   MEMORY DISPLAY         ENTRY POINT
*     FORMAT:  MD[S] <ADDRESS> [<COUNT>] [;DI]
* 1.  IF 'S' USED IN COMMAND THEN IT WILL DUMP 16 LINES, (1 Screen), AND
*     PROMPT-ENTER CR FOR 16 MORE LINES ETC OR ANY MACSBUG COMMAND.
*

MDCMD    DS      0
         BSR     SCANPORT       WHERE TO SEND OUTPUT
         MOVE.L  INPORT1.L,INFROM ONLY ALLOW INPUT FROM PORT1

         MOVEQ   #-1,D6         D6 = HOW MANY LINES PER PAGE

         SUB.L   A3,A3          A4 = DEFAULT ZERO  START
         MOVE.L  A3,A4          A3 = END ADDR

         CLR.L   D7             D7 = DO NOT DISASSEMBLE
         MOVE.L  A5,A2
PRINT8   CMP.L   A2,A6          LOOK FOR OPTIONS
         BCS.S   PRINTDI        NO  OPTIONS
         CMPI.B  #';',(A2)+
         BNE     PRINT8

         MOVE.L  A2,A6
         SUBQ.L  #1,A6          A6 = POINTER TO ;

         CMPI.B  #'D',(A2)+
PRINTMB  BNE     SYNTAX         COMMAND SYNTAX ERROR
         CMPI.B  #'I',(A2)+
         BNE     PRINTMB
         MOVEQ   #-1,D7         DISASSEMBLE OPTION
PRINTDI


* LOOK FOR "S" IN COMMAND
         MOVE.B  (A5),D0
         CMPI.B  #'S',D0
         BNE.S   PRINT5         NO "S" IN COMMAND
         ADDQ.L  #1,A5          MOVE PAST "S"

         MOVEQ   #16,D6         DO 16 LINES AT A TIME
         SUB.L   A3,A3
         SUBQ.L  #1,A3          MAX END ADDRESS

PRINT5   LEA     PRINT7(PC),A0  WHERE TO GO IF NO PARAMETERS
         BSR     FNEXTF         FIND NEXT FIELD
         BSR     GETA
         MOVE.L  D0,A4          A4 = GET ADDRESS

         BSR     FNEXTF
         BSR     GETEXP         D0 = GET COUNT
         ADD.L   A4,D0          END=START+COUNT-1
         SUBQ.L  #1,D0          BACK OFF ONE
         MOVE.L  D0,A3          A3 = END ADDRESS

PRINT7   TST.L   D7
         BEQ.S   PUTADR         NOT DISASSEMBLE
```

```
PRINTDI5 BSR     FIXBUF          ;DI OPTION
         MOVEM.L A3/D6-D7,-(A7)
         MOVEM.L (A4),D0-D2      D0-D2 = DATA TO DISASSEMBLE
         BSR     DCODE68K        DISASSEMBLE
         BSR     OUT1CR
         MOVEM.L (A7)+,A3/D6-D7

         SUBQ.L  #1,D6
         BEQ.S   PRINT9          'MDS' COMMAND

         CMP.L   A4,A3           END CHECK
         BCC     PRINTDI5
         BRA.S   PRINT9

PRINT3   MOVEQ   #16,D6          D6 = LINE BLOCK COUNT
         SUB.L   A3,A3
         SUBQ.L  #1,A3           A3 = MAX END ADDRESS
         BRA     PRINT7

* START A NEW  LINE
*
PUTADR   BSR     FIXBUF          SET UP OUTPUT BUFFER
         MOVE.L  A4,D0           CURRENT LINE ADDRESS
         BSR     FRELADDR        FORM RELATIVE ADDRESS
         MOVE.B  #$20,(A6)+      FORMAT  SPACE
         MOVE.L  A4,A0           A0 IS SCANNING ADDRESS
         MOVEQ   #$10,D3         SET UP COUNTER FOR LOOP

NXTBP    MOVE.B  (A0)+,D0        GET BYTE TO PRINT
         BSR     PNT2HX          PRINT IT
         MOVE.B  #$20,(A6)+      SPACE BETWEEN EACH HEX

         CMPI.B  #9,D3           HALF LINE SPACING
         BNE.S   NXTBP3
         MOVE.B  #$20,(A6)+
NXTBP3

         SUBQ.L  #1,D3
         BNE     NXTBP           LOOP TILL D3 IS ZERO

         MOVE.B  #$20,(A6)+      MOVE A SPACE
         MOVE.L  A4,A0           RELOAD SCANNER FOR ASCII PRINTS
         MOVEQ   #$10,D3         RELOAD COUNTER

NXTCHR   MOVE.B  (A0)+,D0        FETCH BYTE
         ANDI.B  #$7F,D0         REMOVE HIGH ORDER BIT, (ASCII ONLY USES 7 BITS)
         CMPI.B  #$20,D0         SEE IF IT IS CONTROL CHAR
         BLT.S   NOTCHR          BYPASS IF IT IS... ELSE
         CMPI.B  #$7F,D0         IS IT A "7F"?    (CAUSES PRINTER PROBLEM)
         BLT.S   PUTCHR          NO... THEN PRINT IT, ELSE SUBSTITUTE "."
NOTCHR   MOVE.B  #$2E,D0         CHANGE UNPRINTABLE TO PERIOD

PUTCHR   MOVE.B  D0,(A6)+        MOVE "EDITTED" CHARACTER TO PRINT LINE
         SUBQ.L  #1,D3           LOOP AROUND FOR NEXT CHAR
         BNE     NXTCHR          ANY LEFT?
         BSR     OUT1CR          NO... THEN PRINT THE COMPLETE LINE
         ADDQ.L  #$08,A4         UPDATE STARTING ADDRESS OF NEXT ADDRESS
         ADDQ.L  #$08,A4         ........
         CMP.L   A4,A3           DOES NEW LINE START PAST END
         BCS.S   PRINT9          SEE IF STILL GOING
         SUBQ.L  #1,D6           DECR THE COUNTER
         BNE     PUTADR          DO ANOTHER LINE

PRINT9   LEA     MSG001(PC),A5   SET UP FOR PROMPT
         BSR     FIXDATA
         MOVE.B  #'>',(A6)+      PROMPT SAME AS MACSBUG
         MOVE.B  #BLANK,(A6)+
         BSR     OUTPUT          PRINT IT

         BSR     FIXBUF          RESET BUFFER
         BSR     PORTIN1         TAKE IN INPUT
         CMP.L   A6,A5           SEE IF ANYTHING ENTERED
         BEQ     PRINT3          NOTHING ENTERED; DO 16 MORE LINES
         CLR.L   OUTTO           *
         BRA     DECODE6         GO MAKE SURE 2ND DIGIT IS BLANK
```

```
*-------------------------------------------------------------------------
* File MMDI      Modify Memory command WITH asm/disasm           11/27/81

* MODIFY MEMORY
*  COMMON REGISTER USAGE
*   A6  IO BUFFER POINTER  END
*   A5  IO BUFFER POINTER  START
*   A4  ADDRESS TO MODIFY
*   A3
*   A2
*   A1
*   A0
*   D7  DATA READ (DATA STORED)
*   D6  SIZE  1/2/4 BYTES  (ASM/DISASM 2 - 10 BYTES)
*   D5  OVERRIDE BYTE (80XX=NON VERIFY)  (XX80=BYTE SIZE)
*
* ;OPTIONS
*   ;W  WORD
*   ;L  LONG WORD (4 BYTES)
*   ;DI DISASSEMBLE
*     ;O  ODD ADDRESSES ONLY
*     ;V  EVEN ADDRESSES ONLY
*       ;N  NON-VERIFY
*

MMDI     BSR     CKWADR
         BSR     CKADDR
         CLR.L   D6              SIZE = 0
MMDI22   ADD.L   D6,A4           PC = PC + SIZE
MMDI23   BSR     FIXBUF          A5 & A6 = POINTER TO BUFFER

*                  ENTRY
*                   A4 = PROGRAM COUNTER
*                   A5 = POINTER TO STORE DISASSEMBLED LINE
         MOVEM.L   (A4),D0-D2          DATA TO DISASSEMBLE
         BSR       DCODE68K
*                  RETURN
*                   A4 = NEW PROGRAM COUNTER
*                   A5 = PTR START BUFFER
*                   A6 = PRT END BUFFER
*                   D6 = NUMBER OF BYTES DISASSEMBLED

         MOVE.B  #'?',(A6)+
         BSR     OUTPUT

MMDI31   BSR     FIXBUF
         BSR     PORTIN1N        INPUT ASSEMBLY SOURCE (NO LINE FEED)
         CMP.L   A5,A6
         BNE.S   MMDI34
         BSR     OUT1CR          NOTHING INPUT; OUTPUT LINE FEED
         BRA     MMDI23

MMDI34   SUB.L   D6,A4           BACKUP PC

         CMPI.B  #'.',(A5)
         BEQ     MM905           CLOSE

         LEA     SYSTACK+17,A3  STORE BUFFER

*                  A3 = STORE POINTER
*                  A4 = PROGRAM COUNTER
*                  A5 = POINTER TO DATA TO ASSEMBLE
*                  A6 = POINTER TO END OF SOURCE DATA

         BSR     CODE68K         ASSEMBLE

*                  A3 = POINTER TO LINE ASSEMBLED
*                  A4 = PROGRAM COUNTER
*                  A6 = POINTER END OF LINE ASSEMBLED
*                  D0-D2 = DATA ASSEMBLED
*                  D6 = NUMBER OF BYTES ASSEMBLED
*                  D7 = ERROR FLAG & POSITION OF ERROR

         MOVE.L  A3,A5
         ADD.L   #78,A3          A3 = MAX LINE
MMDI26   CMP.L   A6,A3
         BCS.S   MMDI27
         MOVE.B  #BLANK,(A6)+   SPACE FILL LINE
```

```
        BRA     MMDI26
MMDI27  BSR     OUT1CR          PRINT LINE JUST ENTERED

        TST.B   D7
        BNE.S   MMDI30          ERROR; DON'T STORE DATA

        LEA     SYSTACK+4,A2    A2 = TEMP AREA
        MOVEM.L D0/D1/D2,-(A2)  STORE DATA
        MOVE.L  D6,D1           D1 = NUMBER OF BYTES TO STORE
        SUBQ.L  #1,D1
        MOVE.L  A4,A1           A1 = DATA STORE POINTER
MMDI29  MOVE.B  (A2)+,D0
        MOVE.B  D0,(A1)
        MOVE.B  (A1)+,D2        INSURE DATA STORED
        CMP.B   D0,D2
        BNE     MM90
        DBRA    D1,MMDI29
        BRA     MMDI22

MMDI30  BSR     FIXBUF
MMDI44  MOVE.B  #BLANK,(A6)+    SPACES
        DBRA    D7,MMDI44

        MOVE.B  #'X',-1(A6)     X UNDER ERROR
        MOVE.B  #'?',(A6)+      ? ALLOW REINPUT
        BSR     OUTPUT
        BRA     MMDI31


MCMD    DS      0               "M" Alias for "MM" Command
MMCMD   DS      0               "MM" Command -Memory Modify-
        LEA     SYNTAX(PC),A0   A0=ERROR RETURN
        BSR     FNEXTF
        BSR     GETA
        MOVE.L  D0,A4           A4= ADDRESS OF DATA
        MOVEQ   #1,D6           SIZE = BYTE
        CLR.L   D5              NO OVERRIDE

MM05    CMP.L   A5,A6
        BCS.S   MM10            AT END OF BUFFER

* LOOK FOR ;OPTIONS
        MOVE.B  (A5)+,D0
        CMPI.B  #';',D0
        BNE     MM05            IGNORE NOT ;

        MOVE.B  (A5)+,D0        GET NEXT CHAR
        CMPI.B  #'D',D0
        BNE.S   MM045

        CMPI.B  #'I',(A5)+      DISSAMBLY OPTION
        BNE     MM05
        BRA     MMDI

MM045   CMPI.B  #'W',D0
        BEQ.S   MM065           ;W   D6=2

        CMPI.B  #'L',D0
        BNE.S   MM054
        MOVE.B  #4,D6           ;L   D6=4
        CLR.B   D5              RESET BYTE OVERRIDE
        BRA     MM05

MM054   CMPI.B  #'N',D0
        BNE.S   MM056
        ORI.W   #$8000,D5       ;N  D5=$8000
        BRA     MM05

MM056   CMPI.B  #'O',D0
        BNE.S   MM058
        MOVE.L  A4,D0           ;O
        ORI.B   #1,D0           FORCE ODD ADDRESS
        BRA.S   MM060

MM058   CMPI.B  #'V',D0
        BNE     SYNTAX          ERROR
        MOVE.L  A4,D0           ;V
        ANDI.B  #$FE,D0         FORCE EVEN ADDRESS
MM060   MOVE.L  D0,A4
```

```
         ORI.B   #$80,D5        BYTE OVERRIDE
MM064    MOVE.B  #2,D6          SIZE = WORD (2 BYTES)
         BRA     MM05

MM065    CLR.B   D5             RESET BYTE OVERRIDE
         BRA     MM064

* FORMAT ADDRESS FOR PRINTING
MM10     MOVE.L  A4,D0
         CMPI.B  #1,D6
         BEQ.S   MM11           "BYTE"
         TST.B   D5
         BMI.S   MM11           BYTE OVERRIDE
         BSR     CKWADR         CHK ALLIGNMENT
MM11     BSR     CKADDR         *
         BSR     FIXBUF
         BSR     FRELADDR       FORM RELATIVE ADDRESS
         MOVE.B  #BLANK,(A6)+   SPACE

         TST.W   D5
         BMI.S   MM18           NON-VERIFY (DON'T READ MEMORY)

* READ DATA FROM MEMORY & FORMAT IT
         TST.B   D5
         BMI.S   MM12           BYTE OVERRIDE

         CMPI.B  #2,D6
         BEQ.S   MM14           WORD
         CMPI.B  #4,D6
         BEQ.S   MM16           LONG WORD

* BYTE
MM12
         MOVE.B  (A4),D7        D7 = DATA READ
         MOVE.L  D7,D0
         BSR     PNT2HX         FORMAT BYTE
         BRA.S   MM18

* WORD
MM14
         MOVE.W  (A4),D7
         MOVE.L  D7,D0
         BSR     PNT4HX         FORMAT WORD
         BRA.S   MM18

* LONG WORD
MM16
         MOVE.L  (A4),D7
         MOVE.L  D7,D0
         BSR     PNT8HX         FORMAT LONG WORD

MM18     MOVE.B  #BLANK,(A6)+   SPACE
         MOVE.B  #'?',(A6)+
         BSR     OUTPUT

* READ USER INPUT
*  [DATA] (CR)   NEXT
*           ^    LAST
*           =    SAME
*           .    EXIT

         BSR     FIXBUF
         BSR     PORTIN1
         CMP.L   A5,A6
         BEQ     MM50           NO DATA (CR ONLY)

         LEA     MM40(PC),A0    A0 = NO PARAMETER RETURN
         BSR     FNEXTF         FIND NEXT FIELD

* IF = ^ OR .  TAKE ACTION
         MOVE.B  (A5),D0
         CMPI.B  #'=',D0
         BEQ     MM10
         CMPI.B  #'.',D0
         BEQ     MACSBUG
         CMPI.B  #'^',D0
         BEQ.S   MM60

         BSR     GETEXP         GET DATA
```

```
        MOVE.L  D0,D7           D7=DATA STORED

* WE HAVE DATA; STORE IT
        TST.B   D5
        BMI.S   MM22            BYTE OVERRIDE

        CMPI.B  #2,D6
        BEQ.S   MM24            WORD
        CMPI.B  #4,D6
        BEQ.S   MM26            LONG WORD

* BYTE
MM22
        MOVE.B  D0,(A4)         STORE DATA
        TST.W   D5
        BMI.S   MM40            NO-VERIFY
        MOVE.B  (A4),D0
        CMP.B   D7,D0
        BNE.S   MM90            NO MATCH
        BRA.S   MM40

* WORD
MM24
        MOVE.W  D0,(A4)         STORE
        TST.W   D5
        BMI.S   MM40            DO NOT VERIFY
        MOVE.W  (A4),D0
        CMP.W   D7,D0
        BNE.S   MM90            NO MATCH
        BRA.S   MM40

* LONG WORD
MM26
        MOVE.L  D0,(A4)
        TST.W   D5
        BMI.S   MM40            DO NOT VERIFY
        MOVE.L  (A4),D0
        CMP.L   D7,D0
        BNE.S   MM90            NO MATCH

* LOOK FOR  . = ^
MM40    MOVE.B  (A5),D0
        CMPI.B  #'.',D0
        BEQ     MACSBUG         DONE
        CMPI.B  #'^',D0
        BEQ.S   MM60            BACKUP ADDRESS
        CMPI.B  #'=',D0
        BEQ     MM10            ADDRESS STAYS THE SAME
        CMPI.B  #BLANK,D0
        BEQ.S   MM50
        CMP.L   A5,A6
        BNE     SYNTAX          ERROR

* ADDRESS LOW TO HIGH
MM50    ADD.L   D6,A4
        BRA     MM10

* ADDRESS HIGH TO LOW
MM60    SUB.L   D6,A4
        BRA     MM10

MM90    LEA     MSG017(PC),A5  'DATA DID NOT STORE'
MM95    BSR     FIXDCRLF
        BRA     MSG

MM905   LEA     MSGEOT(PC),A5
        BRA     MM95



*-----------------------------------------------------------------------
* File MS        Memory set command                              11/02/81

*    ***MS***   MEMORY SET
*     FORMAT: MS  ADDRESS HEX HEX,HEX,'ASCII'  ETC.
*     COMMAS OR SPACES BETWEEN FIELDS
*     FIELDS ARE SIZE ADJUSTED (STORES UP TO 4 BYTES)
*     ASCII ENCLOSED IN SINGLE QUOTES-ANY LENGTH
```

```
MSCMD    LEA     SYNTAX(PC),A0  IF NO PARAMETERS
         BSR     FNEXTF         FIND NEXT FIELD
         BSR     GETA           GET ADDRESS
         BSR     CKADDR         CHECK VALID ADDRESS

         MOVE.L  D0,A1          A1=START (OPEN) ADDRESS
SETM1    LEA     MACSBUG(PC),A0 IF NO PARAMTER
         BSR     FNEXTF         FIND NEXT FIELD
         MOVE.L  A5,A4          SAVE ADDRESS OF PARAMTER
         MOVE.B  (A5),D0        CHECK OUT NEXT CHARACTER
         CMPI.B  #$27,D0        SEE IF IT IS QUOTE MARK
         BEQ.S   SETM5          SPECIAL ROUTINE
         CMPI.B  #'N',D0        SEE IF NEXT LINE FEATURE
         BEQ.S   SETM7
         BSR     GETNUMA        GET THE DATA
         MOVE.L  A1,A3          ADDRESS
         MOVE.L  A5,D1          COMPUTE BYTES OF DATA
         SUB.L   A4,D1          LEN=END-START
         ASR.L   #1,D1          BYTES=CHAR/2
         BCC.S   SETM3          TAKE CARE OF ODD CHARACTER
         ADDQ.L  #1,D1          WHOLE NUMBER OF BYTES
SETM3    MOVE.L  D1,D2          D1 SCANS DOWN
         SUBQ.L  #1,D2          KNOCK IT DOWN TO INDEX
         MOVE.B  D0,0(A3,D2)    INDEXED BECAUSE BACKWARD

         MOVE.B  0(A3,D2),D3    REREAD TO CHECK IF STORED OK

         CMP.B   D0,D3          ARE SAME?
         BNE.S   SETME          'DATA DID NOT STORE'

         ASR.L   #8,D0          SHIFT ONE BYTE
         ADDQ.L  #1,A1          BUMP ADDRESS
         SUBQ.L  #1,D1
         BNE     SETM3
         BRA.S   SETM1          GO DO NEXT DATA

*  DATA IN IN ASCII STRING
SETM5    ADDQ.L  #1,A5          GET PAST QUOTE MARK
SETM6    CMP.L   A6,A5          SEE IF END OF BUFFER
         BGE     MACSBUG
         MOVE.B  (A5)+,D0       GRAB CHARACTER
         CMPI.B  #$27,D0        SEE IF QUOTE MARK
         BEQ.S   SETM1          IF SO-END OF STRING
         MOVE.B  D0,(A1)        SAVE DATA

         MOVE.B  (A1)+,D1       REREAD FOR CHECK

         CMP.B   D1,D0          SEE IF SAME
         BEQ     SETM6
SETME    LEA     MSG017(PC),A5  'DATA DID NOT STORE'
         BSR     FIXDCRLF
         BRA     MSG

SETM7    BSR     FIXBUF         DISPLAY CURRENT ADDRESS
         MOVE.L  A1,D0
         BSR     PNT8HX         PUT ADDRESS IN BUFFER
         MOVE.L  #'  ? ',(A6)+  PROMPT
         BSR     OUTPUT         DUMP BUFFER WITH NO LF CR
         BSR     FIXBUF         GET READY FOR INPUT
         MOVE.B  #BLANK,(A5)+   ADVANCE IN BUFFER
         MOVE.L  A5,A6          BECAUSE OF SNAFU IN FINDNP
         BSR     PORTIN1        INPUT FROM CONSOLE
         MOVE.B  -(A5),D0       JUST BACK UP IN BUFFER
         CMP.L   A6,A5
         BEQ     MACSBUG
         BRA     SETM1          DO DECODE IT

MSG017   DC.B    'DATA DID NOT STORE',CR,LF,EOT




         DC.B    0              PAD BYTE
```

```
*-----------------------------------------------------------------------
* File MTSETUP   MTSETUP memory test setup                      11/02/81

*
* SET UP PARMS FOR BLOCK TEST AND BLOCK INITIALIZE
*

MTSETUP  DS      0
         MOVE.L  A7,TEMP         STACK FOR EXCEPTION RETURN
         LEA     SYNTAX(PC),A0   WHERE TO GO IF NO PARAMETERS
         BSR     FNEXTF          FIND NEXT FIELD
         BSR     GETA            GET ADDR1
         BSR     CKWADR          CHECK WORD BOUNDRY ADDRESS
         MOVE.L  D0,A3           SAVE STARTING ADDRESS

         LEA     SYNTAX(PC),A0   SET UP TO TRY "TO" ADDRESS
         BSR     FNEXTF          *
         BSR     GETA            GET ADDR2
         BSR     CKWADR
         MOVE.L  D0,A1           A1 = END ADDRESS?
         MOVE.L  A3,A0           A0 = STARTING ADDRESS
         BSR     P2PHY           DISPLAY TWO ADDRESSES
         CMP.L   A0,A1
         BCS     SYNTAX          END ADDR TOO SMALL
         ADDQ.L  #2,A1           ADJUST END ADDR
         RTS




*-----------------------------------------------------------------------
* File OF        OF & PERIOD Command                            12/18/81

* .*  HANDLER
*   GET TWO CHARACTERS FOLLOWING PERIOD

PERCMD   LSL.W   #8,D1
         MOVE.B  (A5),D1         D1 = 2ND,3RD CHARACTERS
         SUBQ.L  #1,A5           A5 = POINTER TO 2ND CHAR  (1ST REAL CHARACTER)

         LEA     REGTBL(PC),A0
PER4     CLR.L   D7
         MOVE.W  (A0)+,D7        SAVE FIRST WORD FOR END OF TABLE TEST
         MOVE.W  (A0)+,D0        GET REAL REGISTER ID INTO D0
         CMPI.W  #$FFFF,D7       ARE WE AT THE END OF THE TABLE?
         BEQ     WHAT            YES...THEN WE DIDNT FIND IT

         CMPI.B  #'@',D0
         BNE.S   PER3            NOT @

* THIRD CHAR MUST BE NUMERIC 0 - 7
         MOVE.B  D1,D0           ALLEGED DIGIT
         CMPI.B  #'0',D0
         BMI     PER4            NOT A DIGIT
         CMPI.B  #'8',D0
         BPL     PER4            NOT A DIGIT

PER3     CMP.W   D1,D0
         BNE     PER4            MISS-MATCH

*****************************************************************
*        AT THIS TIME WE HAVE FOUND THE ENTRY IN THE "REG TABLE"  *
*        WE NOW NEED TO EXTRACT AND USE THE OFFSET                *
*****************************************************************

         LEA     FIRST(PC),A0    A0 = Start of VERSAbug RO
         ADD.L   D7,A0           Add offset
         JMP     (A0)            Now go to the calculated location


***************
REGTBL   EQU  *
***************

         DC.W    SETA7-FIRST     Stack Register Routine
         DC.W    'A7'            *

         DC.W    SETPC-FIRST     Program Counter Routine
         DC.W    'PC'            *
```

```
         DC.W    SETSR-FIRST    Status Register Routine
         DC.W    'SR'           *

         DC.W    SETUS-FIRST    User Stack Routine
         DC.W    'US'           *

         DC.W    SETSS-FIRST    System Stack Routine
         DC.W    'SS'           *

         DC.W    SETD-FIRST     Data Register Routine
         DC.W    'D@'           *

         DC.W    SETA-FIRST     Address Register Routine
         DC.W    'A@'           *

         DC.W    PNTCLSA-FIRST  All Address Registers Routine
         DC.W    'A '           *

         DC.W    PNTCLSD-FIRST  All Data Registers Routine
         DC.W    'D '           *

         DC.W    SETRN-FIRST    All Registers Routine
         DC.W    'R@'           *

         DC.W    $FFFF          END OF TABLE

*    PRINT & INPUT REGISTER ROUTINES

SETD     LEA     REGS,A4        START OF REGISTERS
         BRA.S   SETR

SETA     LEA     REGS+32,A4     OFFSET IN REGISTER TABLE
         BRA.S   SETR

SETPC    LEA     REGPC,A4       WHERE PC IS
         BRA.S   SETR0

SETSR    LEA     REGSR,A4       WHERE SR IS
         BRA.S   SETR0

SETA7    MOVE.L  REGSR,D1       GET CONDITION CODES
         ANDI.W  #$2000,D1      CHECK SUPERVISOR BIT
         BEQ.S   SETUS
SETSS    LEA     REGA7,A4       WHERE SUPERVISOR STACK IS
         BRA.S   SETR0

SETUS    LEA     REGUS,A4       USER STACK
         BRA.S   SETR0

SETRN    LEA     OFFSET,A4      SET OFFSET
         CMPI.B  #'7',D1
         BEQ     SYNTAX         NOT ALLOWED TO CHANGE A7


*  ROUTINE TO ENTER DATA FOR A SINGLE REGISTER
*   A5-A6 ARE COMMAND BUFFER
*   D0 HAS REGISTER DIGIT A4 HAS CLASS OFFSET

SETR     BSR     GETHEX         GET REG NUMBER
         LSL.L   #2,D0          SHIFT LEFT...MULT BY 4
         ADD.L   D0,A4          A4 NOW HAS EXACT ADDRESS
SETR0    ADDQ.L  #2,A5          NOW FIND PARAMETERS
         MOVE.B  #':',D0        SEE IF COLON IN COMMAND
         BSR.S   SCAN
         BEQ.S   SETR5
*SEE IF ANY PARAMER (HEX)
         LEA     SETR4(PC),A0   WHERE TO GO IF NO PARAMETERS
         BSR     FNEXTF         FIND NEXT FIELD

         CMPI.B  #'.',D0
         BEQ.S   SEMACS         PERIOD; GET OUT

         BSR     GETA           GET ADDRESS VALUE
         MOVE.L  D0,(A4)        SAVE NEW VALUE
SEMACS   BRA     MACSBUG

*JUST PRINT IT
```

```
SETR4    BSR.S   PRINTR          FIX UP TO PRINT
         BRA     MSG             GO PRINT MESSAGE-GO TO MACSBUG

SETR5    BSR.S   SETSR1
         BRA     SEMACS

SETSR1   BSR.S   PRINTR          FIX UP TO PRINT
         MOVE.B  #BLANK,(A6)+    SPACE
         MOVE.B  #'?',(A6)+      PROMPT
         MOVE.B  #BLANK,(A6)+    SPACE
         BSR     OUTPUT          PRINT IT

         BSR     FIXBUF
         BSR     PORTIN1

         LEA     SETSR15(PC),A0 A0=DEFAULT (NO PARM) ADDRESS
         BSR     FNEXTF          FIND FIELD

         CMPI.B  #'.',D0
         BEQ     SEMACS          PERIOD; GET OUT

         BSR     GETA            CONVERT IT
         MOVE.L  D0,(A4)         STORE NEW DATA

SETSR15  DS      0
         RTS


* SEE IF CHARACTER IS IN BUFFER

SCAN     MOVE.L  A5,A0           A0 IS WORKING SCANNER
SCAN2    CMP.L   A6,A0           SEE IF AT END OF BUFFER
         BHI.S   RETURN5
         CMP.B   (A0),D0         LOOK AT CHARACTER
         BEQ.S   RETURN5
         ADDQ.L  #1,A0           GET PAST CHARACTER
         BRA.S   SCAN2
RETURN5  RTS

* ROUTINE TO SET UP TO PRINT REG

PRINTR   BSR     FIXBUF

         ADDQ.L  #3,A6           GET PAST REG NAME (.XX)
         MOVE.B  #'=',(A6)+      PUT IN EQUAL SIGN
         MOVE.L  (A4),D0         GET VALUE
         CMPA.L  #REGSR,A4       SEE IF THIS IS CONDITION CODES
         BNE.S   PRINTR2
         BSR     PNT4HX          JUST PRINT WORD
         RTS

PRINTR2  BSR     PNT8HX          PRINT THE VALUE
         RTS

*
*   PRINT ALL REGISTERS IN A CLASS (A OR D OR R)
*

OFCMD    DS      0               "OF" Command -Display Offset registers-
SETO     MOVE.B  #'R',D7
         LEA     OFFSET,A3
         BRA.S   PNTCLSB

PNTCLSD  MOVE.B  #'D',D7         CLASS=DATA
         LEA     REGS,A3         OFFSET
         BRA.S   PNTCLSB

PNTCLSA  MOVE.B  #'A',D7         CLASS=ADDRESS
         LEA     REGS+32,A3      OFFSET
PNTCLSB  BSR.S   PNTCLS
         BRA     MACSBUG

PNTCLS   BSR     FIXBUF
         CLR.L   D6              REGISTER COUNTER
PNTCLS1  BSR.S   PNTREG          PRINT THE REGISTER
         CMPI.B  #4,D6           DISPLAY AFTER 3&7
         BNE.S   PNTCLS2
         BSR     OUT1CR
         BSR     FIXBUF
```

```
         BRA     PNTCLS1         DO SOME MORE

PNTCLS2  CMPI.B  #8,D6           AT END?
         BNE     PNTCLS1
         BSR     OUT1CR          PRINT IT
         RTS

*  SUBROUTINE TO PRINT REGISTER  X#=01234567.

PNTREG   MOVE.B  D7,(A6)+        CLASS
         MOVE.B  D6,D0           REG#
         BSR     PUTHEX
         MOVE.B  #'=',(A6)+      EQUAL SIGN
         MOVE.L  D6,D0           COMPUTE ADDRESS OF REG
         LSL.L   #2,D0           MULT BY FOUR
         ADD.L   A3,D0           ADD IN OFFSET
         MOVE.L  D0,A4           SET UP TO GET DEFFERED
         CMPA.L  #REGA7,A4       SEE IF REG A7
         BNE.S   PNTREG1
         MOVE.L  REGSR,D0        GET STATUS REGISTER
         ANDI.W  #$2000,D0       CHECK SUPERVISOR BIT
         BNE.S   PNTREG1
         LEA     REGUS,A4        TAKE ADDRESS OF USER STACK
PNTREG1  MOVE.L  (A4),D0         GET REG CONTENT
         BSR     PNT8HX          PUT IN BUFFER
         MOVE.B  #BLANK,(A6)+    SPACE
         ADDQ.L  #1,D6           BUMP REG#
         RTS



*---------------------------------------------------------------------------
* File PF        "PFCMD", Port format                              05/19/82

* **PF**   PF      DISPLAY PORT PROFILE
*          PF1     DISPLAY/CHANGE PORT 1
*          PF2     DISPLAY/CHANGE PORT 2

PFCMD    MOVE.B  (A5),D6         D6 = PORT #
         CMPI.B  #'1',D6
         BEQ.S   PFCMD1          CHANGE PORT 1
         CMPI.B  #'2',D6
         BEQ.S   PFCMD1          CHANGE PORT 2

         LEA     MD1CON,A1       PRINT BOTH PORTS
         LEA     MSG003(PC),A5
         BSR.S   PFPT            DISPLAY/CHANGE
         BSR     OUTPUT

         LEA     NULLPADS,A1
         LEA     MSG004(PC),A5
         BSR.S   PFPT
         BSR     OUTPUT

         LEA     CRPADS,A1
         LEA     MSG005(PC),A5
         BSR.S   PFPT

         LEA     MSG031(PC),A5   TELL WHERE XONOFF IS
         BSR     FIXDADD
         MOVE.L  #OPTIONS,D0
         BSR     PNT6HX          "OPTIONS@AAAAAA"
         BRA     MSG

PFPT     BSR     FIXDCRLF        FORMAT FROM A5
         MOVE.B  (A1),D0
         BSR     PNT2HX          FORMAT DATA
         MOVE.B  #BLANK,(A6)+    SPACE
         MOVE.B  1(A1),D0
         BSR     PNT2HX          FORMAT DATA PORT 2
         RTS


PFCMD1   LEA     MD1CON,A1
         LEA     MSG003(PC),A5
         BSR.S   PFCH            DISPLAY/CHANGE

         LEA     NULLPADS,A1
```

```
         LEA     MSG004(PC),A5
         BSR.S   PFCH

         LEA     CRPADS,A1
         LEA     MSG005(PC),A5
         BSR.S   PFCH
         BSR     INITSER         PLACE NEW SBITS INTO ACIA
         BSR     FIXBUF          BLANK LINE FOR SPACING
         BRA     MSG

PFCH     BSR     FIXDATA         FORMAT FROM A5
         CMPI.B  #'1',D6
         BEQ.S   PFCH2           PORT 1
         ADDQ.L  #1,A1           PORT 2
PFCH2    MOVE.B  (A1),D0
         BSR     PNT2HX          FORMAT DATA
         MOVE.B  #'?',(A6)+
         BSR     OUTPUT

         BSR     PORTIN1         INPUT LINE
         CMP.L   A5,A6
         BEQ.S   PFCH4           NOTHING INPUT

         BSR     GETNUMA
         MOVE.B  D0,(A1)
PFCH4    RTS

MSG003   DC.B    'FORMAT=   ',EOT


MSG004   DC.B    'CHAR NULL=',EOT


MSG005   DC.B    'C/R  NULL=',EOT


MSG031   DC.B    CR,LF,'OPTIONS@',EOT




*------------------------------------------------------------------------
* File RAMTEST   RAMTEST                                        11/02/81

************************************************************************
* MEMORY TEST SUBROUTINE.                                              *
*                                                                      *
*        A0=BEGINNING ADDRESS (MUST BE EVEN ADDRESS)                   *
*        A1=ENDING ADDRESS+1  (MUST BE EVEN ADDRESS)                   *
*        A2=FAILING ADDRESS                                            *
*        D0=DATA WRITTEN TO RAM                                        *
*        D1=DATA READ FROM RAM                                         *
*                                                                      *
*        ZERO FLAG IS SET IF TEST IS SUCCESSFUL                        *
*           REGISTERS D2,D3,D4 DESTROYED                               *
************************************************************************

*       -1ST TEST-
RAMTEST  DS      0
         MOVE.L  A0,D3           D3 = BEGINNING ADDRESS
         MOVE.L  D3,A2           USE A2 AS POINTER IN MEMORY
WALK3    MOVEQ   #$FE,D0         PREPARE FOR "WALKING BIT" TEST
WALK0
         MOVE.W  D0,(A2)         STORE D0 INTO MEMORY
         MOVE.W  (A2),D1         D1 CONTAINS RAM DATA
         CMP.W   D0,D1           WRITTEN VS. READ
         BNE.S   RAMERR          STOP ON ERROR
         ROL.W   #1,D0           ROLL A ZERO IN A FIELD OF ONES
         BCS.S   WALK0           CONTINUE TILL DONE

         MOVEQ   #$00000001,D0   THIS TIME, WALK A 1 THROUGH ZEROS
WALK1
         MOVE.W  D0,(A2)         STORE D0 INTO MEMORY
         MOVE.W  (A2),D1         D1 CONTAINS RAM DATA
```

```
         CMP.W   D0,D1          WRITTEN VS. READ
         BNE.S   RAMERR         STOP ON ERROR
         ASL.W   #1,D0          ROLL A ONE TO THE NEXT POSITION
         BCC.S   WALK1          CONTINUE TILL DONE

         LEA     $0100(A2),A2   GO TO NEXT 256TH POSITION
         CMP.L   A1,A2          CHECK TO SEE IF DONE
         BLT.S   WALK3          CONTINUE

* -2ND TEST-
         MOVE.L  D3,A2          SET A2 TO POINT TO START OF MEMORY
         MOVEQ   #$00000000,D0  CLEAR D0
MTCLR
         MOVE.W  D0,(A2)+       CLEAR MEMORY
         CMP.L   A1,A2          DONE?
         BNE.S   MTCLR          NO... ZERO ALL OF MEMORY

         MOVEQ   #$FF,D2        SET D2 = FFFF
MTSTOR1
         MOVE.W  -(A2),D1       FIRST READ BACK MEMORY
         CMP.W   D0,D1          CHK AGAINST WHAT WAS WRITTEN
         BNE.S   RAMERR         STOP ON ERROR
         MOVE.W  D2,(A2)        STORE COMPLEMENT
         CMP.L   D3,A2          DONE?
         BNE.S   MTSTOR1        NO... COMPLEMENT ALL OF MEMORY
         MOVE.W  D2,D0          D0=WHAT WAS WRITTEN
         NOT.W   D2             SAVE COMPLEMENT FOR LATER

MTSTOR0
         MOVE.W  (A2),D1        READ BACK MEMORY
         CMP.W   D0,D1          CHK AGAINST WHAT WAS WRITTEN
         BNE.S   RAMERR         STOP ON ERROR
         MOVE.W  D2,(A2)+       STORE COMPLEMENT
         CMP.L   A1,A2          DONE?
         BNE     MTSTOR0        NO...KEEP LOOPING, YES...

RAMERR   MOVE.L  D3,A0          RESTORE A0
         RTS                    RETURN



*-------------------------------------------------------------------------
* File TM        TM  Transparent mode                              12/28/81

*    TM  [[EXIT CHAR]TRAILING CHAR]
*
*  In transparent mode the terminal is physically connected to the
*  host; at the same time the terminal is watched by TM software
*  for the EXIT character.  When the exit character is transmitted
*  by the terminal it goes directly to the host; and at the same
*  time the TM software reconfigures the hardware.  If a nonnull
*  trailing character is present it is now sent via port 2 to
*  the host.

*   TRANSPARENT MODE

TMCMD    LEA     P2CMD0(PC),A0
         BSR     FNEXTF         FIND NEXT FIELD
         MOVE.B  (A5)+,TMCHARS+1 QUIT CHARACTER
         BSR     FNEXTF         FIND NEXT FIELD
         MOVE.B  (A5)+,TMCHARS  OPTIONAL TRAILING CHAR
P2CMD0   MOVE.W  TMCHARS,D7
         BSR     GETSER1        ADDRESS FOR PORT1 INTO A0
         LEA     MSG006(PC),A5  "TRANSPARENT MODE"
         BSR     FIXDCRLF       SET UP FOR MESSAGE
         MOVE.L  D7,D0          EXIT CHARACTER
         BSR     PNT2HX         PRINT 2 HEX CHARACTERS
         MOVE.W  #' =',(A6)+
         MOVE.B  D7,D0          SEE IF CONTROL CHAR
         CMPI.B  #BLANK,D0
         BPL.S   P2CMD01
         MOVE.L  #' CTL',(A6)+
         ADDI.B  #64,D0         MAKE IT A PRINTABLE CHARACTER
P2CMD01  MOVE.B  #BLANK,(A6)+
         MOVE.B  D0,(A6)+
         MOVE.W  #$0D0A,(A6)+
         BSR     OUT1CR         GO PRINT BUFFER WITH CRLF
```

```
         MOVE.B  MD1CON.L,D0    PROGRAM ACIA FOR TRANSPARENT MODE
         ANDI.B  #$9F,D0
         ORI.B   #$40,D0        FORCE RTS HIGH
         MOVE.B  D0,(A0)
P2CMD2   BTST.B  #$0,(A0)      READ STATUS
         BEQ.S   P2CMD2
         MOVE.B  2(A0),D0       RECEIVE CHAR FROM PORT 1
         ANDI.B  #$7F,D0
         CMP.B   D7,D0          SEE IF QUIT CHARACTER (CTL A USUALLY)
         BNE.S   P2CMD2
         MOVE.B  MD1CON.L,D0    REPROGRAM FOR NON-TRANSPARENT
         ANDI.B  #$9F,D0
         MOVE.B  D0,(A0)

         ASR.W   #8,D7
         TST.B   D7
         BEQ.S   P2CMD6
* SPECIAL SECOND CHAR TO HOST SEQUENCE
         MOVE.L  #DELAYC1,D0
P2CMD4   SUBQ.L  #1,D0          DELAY; ALLOW HOST TO SYNC
         BNE.S   P2CMD4
         BSR     GETSER2
         MOVE.B  D7,2(A0)       SEND CHAR
P2CMD6   BRA     MACSBUG

MSG006   DC.B    '*TRANSPARENT* EXIT=$',EOT




         DC.B    0              PAD BYTE



*-------------------------------------------------------------------------
* File W         Software Abort, Hex print routines              06/05/82

*
*   HANDLE THE ABORT BUTTON
*
ABORTB   MOVE.W  #$2700,SR
         SAVEREGS
         BSR     FAULTSER       RESET SERIAL PORTS
         LEA     MSG012(PC),A5  'SOFTWARE ABORT'

         BSR     INITSER        RESET SERIAL PORTS (CLEAR TM MODE)
         BSR     FIXDCRLF       MOVE MESSAGE TO BUFFER
EVECT4   BSR     OUTPUT         MSG TO PORT1
ABORT335 BSR     TDISPLY        DISPLAY REGISTERS
         BRA     MACSBUG

MSG012   DC.B    LF,LF,'SOFTWARE ABORT',CR,LF,EOT




         DC.B    0              PAD BYTE

ABORTE MOVE.L    #'????',$30    UNKNOWN INTERRUPT


*    SAVE REGISTERS AND PRINT VECTOR MSG
*
EVECTL   SAVEREGS
         BSR     FAULTSER       RESET SERIAL PORTS
EVECT2   BSR     FIXBUF         PRINT MESSAGE "XXXX TRAP ERROR"
         MOVE.W  #$0D0A,(A6)+
         MOVE.L  AV12,(A6)+     TYPE OF ERROR
         LEA     MSG010(PC),A5  'TRAP ERROR'
         BSR     FIXDADD
         BRA     EVECT4

MSG010   DC.B    ' TRAP ERROR',CR,LF,EOT
```

```
*
*  PRINT HEX ROUTINES
*
*
* PRINT 8 HEX CHARACTERS
*
*  D0,D1,D2 DESTROYED
*
PNT8HX   SWAP    D0             FLIP REG HALVES
         BSR.S   PNT4HX         DO TOP WORD
         SWAP    D0             NOW DO LOWER WORD
         BRA.S   PNT4HX
* PRINT 6 HEX CHARACTERS
PNT6HX   SWAP    D0             FLIP REGISTER HALVES
         BSR.S   PNT2HX
         SWAP    D0             FLIP BACK REG HALVES
* PRINT 4 HEX CHARACTERS IN D0.W
PNT4HX   MOVE.W  D0,D1          SAVE IN TEMP
         ROR.W   #8,D0          GET BITS 15-8 INTO LOWER BYTE
         BSR.S   PNT2HX         PRINT IT
         MOVE.W  D1,D0          PULL IT BACK
* PRINT 2 HEX CHARACTERS IN D0.B
PNT2HX   MOVE.W  D0,D2          SAVE IN TEMP REG
         ROXR.W  #4,D0          FORM UPPER NIBBLE
         BSR.S   PUTHEX         PUT ASCII INTO PRINT BUFFER
         MOVE.W  D2,D0          GET BACK FROM TEMP
* CONVERT D0.NIBBLE TO HEX & PUT IT IN PRINT BUFFER
*
PUTHEX   ANDI.B  #$0F,D0        SAVE LOWER NIBBLE
         ORI.B   #$30,D0        CONVERT TO ASCII
         CMPI.B  #$39,D0        SEE IF IT IS>9
         BLE.S   SAVHEX
         ADD     #7,D0          ADD TO MAKE 10=>A
SAVHEX   MOVE.B  D0,(A6)+       PUT IT IN PRINT BUFFER
         RTS

* FORMAT RELATIVE ADDRESS  AAAAAA+Rn
*        ENTER     D0 = VALUE
*                  A6 = STORE POINTER
*
FRELADDR MOVEM.L D1/D5-D7/A0,-(A7)
         LEA     OFFSET,A0
         MOVEQ   #-1,D7         D7 = DIFF. BEST FIT
         CLR.L   D6             D6 = OFFSET POSITION

FREL10   MOVE.L  D0,D1
         TST.L   (A0)
         BEQ.S   FREL15         ZERO OFFSET
         SUB.L   (A0),D1        D1 = DIFF.
         BMI.S   FREL15         NO FIT

         CMP.L   D7,D1
         BCC.S   FREL15         OLD FIT BETTER

         MOVE.L  D1,D7          D7 = NEW BEST FIT
         MOVE.L  D6,D5          D5 = POSITION

FREL15   ADDQ.L  #4,A0
         ADDQ.L  #1,D6
         CMPI.W  #8,D6
         BNE     FREL10         MORE OFFSETS TO CHECK

         TST.L   D7
         BMI.S   FREL25         NO FIT
         TST     D6
         BNE.S   FREL20
         TST.L   OFFSET
         BEQ.S   FREL25         R0 = 000000; NO FIT

FREL20   MOVE.L  D7,D0
         BSR     PNT6HX         FORMAT OFFSET
         MOVE.B  #'+',(A6)+     +
         MOVE.B  #'R',(A6)+     R
         ADDI.B  #'0',D5        MAKE ASCII
         BRA.S   FREL30

FREL25   BSR     PNT6HX         FORMAT ADDRESS AS IS
         MOVE.B  #BLANK,D5
         MOVE.B  D5,(A6)+       THREE SPACES FOR ALIGNMENT
```

```
         MOVE.B  D5,(A6)+
FREL30   MOVE.B  D5,(A6)+

         MOVEM.L (A7)+,D1/D5-D7/A0
         RTS

*
*
*   PRINT HEX (ZERO SURPRESS)
*
PNTZHX   CLR.L   D4              IS ZERO WHEN SURPRESSING
         MOVE.L  D0,D1           SAVE IN TEMP
         BEQ.S   PNTZ81          IF ZERO
         BPL.S   PNTZ0
         NEG.L   D1              CHANGE TO POSITIVE VALUE
         BMI.S   PNTZ81          WATCH OUT SPECIAL CASE $80000000
         MOVE.B  #'-',(A6)+      PUT SIGN INTO BUFFER
PNTZ0    MOVEQ   #8,D2           8 POSSIBLE CHARACTERS
PNTZ1    MOVE.L  D1,D0           UNSAVE IT
         MOVE.L  D2,D3           COUNT DOWN FROM HERE
         SUBQ.L  #1,D3           BACK OFF ONE
         BEQ.S   PNTZ4           IF NO ROTATE SKIP THIS
PNTZ2    ASR.L   #4,D0           ROTATE LRIGHT
         ANDI.L  #$FFFFFFF,D0    CLEAR TOP NIBBLE
         SUBQ.L  #1,D3
         BNE     PNTZ2
PNTZ4    ANDI.B  #$F,D0          SAVE ONLY NIBBLE
         BNE.S   PNTZ3
         TST.B   D4              SEE IF STILL SURPRESSING
         BEQ.S   PNTZ8
PNTZ3    BSR     PUTHEX          PUT A HEX CHAR IN BUFFER
         MOVE.B  D0,D4           MARK AS NON-SURPRESSING MODE
PNTZ8    SUBQ.L  #1,D2           DO ANOTHER CHAR
         BNE     PNTZ1
         TST.B   D4              SEE IF ANYTHING PRINTED
         BNE.S   PNTZ9
PNTZ81   MOVE.B  #'0',(A6)+      MOVE AT LEAST ONE ZERO
PNTZ9    RTS

* FIND NEXT FIELD... (A5) POINTS TO FIELD, (A0) NULL FIELD RETURN
*   *-*-WARNING-*-* IF (A0) RETURN IS USED;  RETURN MUST BE AT
*                   SAME SUBROUTINE LEVEL OR STACK GETS SCREWED UP
*
*  SKIP LEADING SPACES
*  TERMINATE NULL FIELD ON COMA ,SEMICOLEN OR A5=A6

FNEXTF1  ADDQ.L  #1,A5

FNEXTF   MOVE.B  (A5),D0         ENTRY POINT
         CMP.L   A6,A5
         BCC.S   FNEXTF7         AT END OF BUFFER
         CMPI.B  #BLANK,D0
         BEQ     FNEXTF1         SPACE

         CMPI.B  #',',D0
         BEQ.S   FNEXTF7         COMMA - NULL FIELD
         CMPI.B  #';',D0
         BEQ.S   FNEXTF7         SEMICOLON - NULL FIELD
         RTS                     NORMAL FOUND FIELD RETURN

FNEXTF7  ADDQ.L  #4,A7           TRIM STACK
         JMP     (A0)            NULL FIELD RETURN

************************************************************
* THIS ROUTINE MAKES SURE YOU ARE ON THE CORRECT BOUNDRY.    *
*  D0 = ARGUMENT...   CHECKS ADDRESS AND WORD ALIGNMENTS.   *
************************************************************

CKWADR   ROR.L   #1,D0
         ROL.L   #1,D0
         BCS.S   CKADR39         NOT WORD ALIGNED

CKADDR   CMPI.L  #$1000000,D0    VALID ADDRESS?
         BCS.S   CKADR99         GOOD ADDRESS

CKADR39  LEA     MSG018(PC),A5
         BSR     FIXDATA
         BSR     PNT8HX
         BSR     OUT1CR
```

```
         BRA     MACSBUG

CKADR99  RTS

MSG018   DC.B    'INVALID ADDRESS=',EOT



         DC.B    0              PAD BYTE

* EVALUATE EXPRESSION
*  NUMBER PLUS OR MINUS NUMBER....
*
GETEXP   MOVE.L  D7,-(A7)       SAVE D7
         CLR.L   D7
GETEXP21 BSR.S   GETNUMA        GET NUMBER
         ADD.L   D0,D7          D7 = NUMBER BEING BUILT
GETEXP15 MOVE.B  (A5)+,D1       D1 = TERMINATING CHAR
         CLR.L   D0             D0 = NEXT NUMBER (=0 1ST TIME)
         CMPI.B  #'+',D1
         BEQ     GETEXP21       PLUS
         CMPI.B  #'-',D1
         BNE.S   GETEXP39       NOT MINUS
         BSR.S   GETNUMA        GET NEXT NUMBER
         SUB.L   D0,D7
         BRA     GETEXP15

GETEXP39 MOVE.L  D7,D0          D0 = VALUE BUILT
         SUBQ.L  #1,A5          A5 = CHAR AFTER EXPRESSION
         MOVE.L  (A7)+,D7       RESTORE D7
         RTS

*   ROUTINES TO CONVERT STRING TO BINARY VALUE
*
*        ACCEPTS PREFIXES:
*         % binary
*         @ octal
*         & decimal
*         $ hex
*
GETNUMD  BSR.S   GETDECNU       GET DECIMAL NUMBER
GETNUMD9 BNE     ERROR
         RTS

GETNUMA  BSR.S   GETHEXNU       GET HEX NUMBER
         BRA     GETNUMD9

GETDECNU MOVEM.L D1-D5/A0,-(A7) DECODE AS A DECIMAL NUMBER
         MOVEQ   #10,D3         D3 = BASE (DECIMAL)
         BRA.S   GETN20

GETHEXNU MOVEM.L D1-D5/A0,-(A7) DECODE AS A HEX NUMBER
         MOVEQ   #16,D3         D3 = BASE

GETN20   CLR.W   D4             D4 = FLAG FOR CHARACTER HIT
         CLR.L   D1             D1 = TEMP RESULTS
         CMPI.B  #BLANK,(A5)
         BEQ.S   GETN80         FIRST CHAR SPACE; ERROR

GETN30   CMP.L   A6,A5          SEE IF AT END OF BUFFER
         BEQ.S   GETN90
         CLR.L   D0
         MOVE.B  (A5)+,D0       D0 = CHARACTER

         LEA     GETNDATA(PC),A0 POINTER TO TERMINATE CHARS
GETN40   MOVE.B  (A0)+,D5       D5 = TERMINATION CHAR
         BEQ.S   GETN50         NOT TERM.CHAR
         CMP.B   D5,D0
         BEQ.S   GETN85         FOUND TERMINATION CHAR; EXIT
         BRA     GETN40

GETN50   CMPI.B  #'%',D0        SEE IF FLAGGED AS BINARY
         BNE.S   GETN60
         MOVEQ   #2,D3          BASE IS BINARY
GETN55   TST.W   D4             ANYTHING ENTERED
         BNE.S   GETN80         CONFLICT

GETN57   MOVEQ   #-1,D4         SET DATA ENTERED FLAG
```

```
         BRA     GETN30

GETN60   CMPI.B  #'@',D0        SEE IF FLAGGED AS OCTAL
         BNE.S   GETN65
         MOVEQ   #8,D3           BASE IS OCTAL
         BRA     GETN55

GETN65   CMPI.B  #'&',D0        SEE IF FLAGGED AS DECIMAL
         BNE.S   GETN70
         MOVEQ   #10,D3         BASE IS DECIMAL
         BRA     GETN55

GETN70   CMPI.B  #'$',D0        SEE IF FLAGGED AS HEX
         BNE.S   GETN75
         MOVEQ   #16,D3         BASE IS 16
         BRA     GETN55

GETN75   SUBI.B  #$30,D0
         BLT.S   GETN80         LESS THAN ZERO
         CMPI.B  #9,D0
         BLE.S   GETN77         VALUE 0 - 9
         CMPI.B  #$11,D0
         BLT.S   GETN80         LESS THAN A
         CMPI.B  #$16,D0
         BGT.S   GETN80         GREATER THAN F
         SUBQ.B  #7,D0          MAKE BINARY

GETN77   CMP.W   D3,D0          IS INPUT LARGER THAN BASE
         BCC.S   GETN80         LARGER THAN BASE
         MOVE.L  D1,D2          GET READY TO MULTIPLY D1*BASE
         SWAP    D1
         MULU    D3,D1          TOP PART
         SWAP    D1
         TST.W   D1             CHECK FOR OVERFLOW
         BNE.S   GETN80         VALUE IS TOO LARGE
         MULU    D3,D2          NOW WORK ON LOW WORD
         ADD.L   D2,D1          PUT IT BACK TOGETHER
         BVS.S   GETN80         OVERFLOW?
         ADD.L   D0,D1          ADD IN NEW STUFF
         BVC     GETN57
GETN80   MOVEQ   #-1,D0         ERROR RETURN (BNE)
         BRA.S   GETN95

GETN85   SUBQ.L  #1,A5          BACK UP TO POINT AT SPACE/PREVIOUS CHAR
GETN90   MOVE.L  D1,D0          SET UP RESULT FOR RETURN
         CLR.L   D1             NORMAL RETURN (BEQ)
GETN95   MOVEM.L (A7)+,D1-D5/A0 RESTORE SOME REGISTERS
         RTS

GETNDATA DC.B    ' (+,-.:;=^]',0 TERMINATE CHARS



*
*  ***GETHEX***  GET HEX (BINARY VALUE FROM ASCII)
*   D0.B HAS ASCII CHAR  RETURNS $0-$F BINARY
*
GETHEX CMPI.B    #$30,D0        IS IT LESS THAN ZERO
       BLT.S     PNMSG011
       CMPI.B    #$39,D0        IS IT GREATER THAN 9
       BLE.S     GTHX2          GOOD HEX

       CMPI.B    #$41,D0        IS IT LESS THAN 'A'
       BLT.S     PNMSG011
       CMPI.B    #$46,D0        IS IT GT 'F'
       BGT.S     PNMSG011
       SUBQ.B    #7,D0          MAKE IT SMALLER A=10
GTHX2  ANDI.L    #$F,D0
       RTS

PNMSG011 BSR     FIXBUF         PRINT NOT A HEX DIGIT
         MOVE.B  D0,(A6)+       PUT IN OFFENDING CHAR
         LEA     MSG011(PC),A5  '  IS NOT A HEX DIGIT'
         BSR     FIXDADD
         BRA     MSG            GO PRINT IT AND ENTER MACSBUG

MSG011   DC.B    '  IS NOT A HEX DIGIT',EOT
```

```
         DC.B    0              PAD BYTE



*-------------------------------------------------------------------------
* File X         COMMON I/O                                      05/17/82

*****************************
*
*    SEND LINE TO PORT1 WITH CR/LF
*
OUT1CR   MOVE.B  #CR,(A6)+
         MOVE.B  #LF,(A6)+
         TST.L   OUTTO          SEE IF ALTERNATE ADDRESS
         BEQ.S   OUT1CRX
         MOVE.L  OUTTO,-(A7)    PUSH ON STACK
         RTS                    GO DO IT

OUT1CRX  MOVE.L  OUTPORT1,-(A7) GO TO I/O ADDRESS
         RTS                    (THIS NORMALLY GOES TO OUT1CR0)
*
OUT1CR0  DS      0
*
*    OUTPUT BUFFER TO PORT1
*
OUTPUT   MOVEM.L D0-D3/A0-A1,-(A7) GET SOME WORKING ROOM
         CLR.L   D3             D3 = NO OFFSET INTO NULL CHARACTERS
         BSR.S   GETSER1        A0 = GET SERIAL PORT 1 ADDRESS
         BRA.S   OUTP2
*
*    SEND LINE TO PORT2 WITH CR
*
OUTPUT2  MOVE.B  #$D,(A6)+      TACK ON A "CR"
         TST.L   OUTTO          SEE IF ALTERNATE ADDRESS
         BEQ.S   OUTPUT2X
         MOVE.L  OUTTO,-(A7)    PUSH ON STACK
         RTS                    GO DO IT

OUTPUT2X MOVE.L  OUTPORT2,-(A7) GO TO IO ROUTINE
         RTS                    (THIS NORMALLY GOES TO OUTPUT20)
*
OUTPUT20 DS      0
*
*     OUTPUT BUFFER TO PORT2
*
OUTPUT21 MOVEM.L D0-D3/A0-A1,-(A7)
         MOVEQ   #1,D3          SIGNAL FOR PORT2
         BSR.S   GETSER2        RETURNS SERIAL PORT 2 BASE ADDRESS IN A0
*
*   SEND BUFFER TO PORT
*
OUTP2    CMP.L   A6,A5          SEE IF AT OR BEYOND END OF LINE
         BCS.S   OUTP3          NO... KEEP GOING, ELSE
         MOVEM.L (A7)+,D0-D3/A0-A1 RESTORE REGISTERS
         RTS                    END OF ROUTINE

OUTP3    MOVE.B  (A5)+,D0       GRAB BYTE TO OUTPUT
         BSR.S   OUTCH1         GO PRINT IT
         BRA.S   OUTP2          GO DO ANOTHER

*   SEND CHARACTER IN D0.B TO PORT1
*   WHOSE BASE ADDRESS IS IN A0
*   D3=0 FOR PORT1  D3=1 FOR PORT2
*   PUTS IN NULL PADDING AS NEEDED
*
OUTCH1   BSR     OUTCH          GO PRINT D0
         TST.B   D0             SEE IF NULL
         BEQ.S   OUTCHRTS       JUST END IF NULL
         CLR.L   D2             CLEAR UPPER BYTES OF NULL LOOP COUNTER
         LEA     NULLPADS,A1    FORM ADDRESS OF PADS
         ADD.L   D3,A1          D3=0 FOR PORT1 1=PORT2
         MOVE.B  (A1),D2        DEFAULT NULL PADS
         CMPI.B  #$D,D0         SEE IF CR
         BNE.S   OUTCH2
         LEA     CRPADS,A1      FORM ADDRESS OF CR PADS
```

```
         ADD.L   D3,A1          D3=0 FOR PORT1  1=PORT2
         MOVE.B  (A1),D2        NULLS AFTER CR
OUTCH2   TST.L   D2  SEE        IF ANY PADDS TO BE SEND
         BEQ.S   OUTCHRTS       0=NONE
         CLR.L   D0             0=NULL CHAR TO BE SEND
OUTCH3   BSR     OUTCH          SEND A NULL
         SUBQ.L  #1,D2          LOOP AROUND
         BNE     OUTCH3
OUTCHRTS RTS                    END OF OUTCH ROUTINE

*
*   GET BASE ADDRESS OF SERIAL PORT 1 IN  A0
*

GETSER1  LEA     SER1,A0        DEFAULT
         TST.L   ALTSER1        IF ALTERNATE IS ZERO
         BEQ.S   RETURN         THEN RETURN
         MOVE.L  ALTSER1,A0     ELSE USE ALTERNATE SERIAL PORT 1
RETURN   RTS     RETURN         (USED FROM A COUPLE OF PLACES)


*
*   GET BASE ADDRESS OF SERIAL PORT 2 IN A0
*

GETSER2  LEA     SER2,A0        DEFAULT SERIAL PORT 2
         TST.L   ALTSER2        IF ALTERNATE IS ZERO
         BEQ     RETURN         THEN RETURN
         MOVE.L  ALTSER2,A0     ELSE USE ALTERNATE SERIAL PORT 2
         RTS                    RETURN
*
*     FIX THE BUFFER A5 & A6 SET TO START OF BUFFER QUE
*
FIXBUF   LEA     BUFFER,A5
         MOVE.L  A5,A6
         RTS

*   INPUT A LINE FROM PORT1
*
PORTIN1  TST.L   INFROM         SEE IF ALTERNAME ADDRESS
         BEQ.S   PORTIN1X
         MOVE.L  INFROM,-(A7)   PUSH ON STACK
         RTS                    GO DO IT
PORTIN1X MOVE.L  INPORT1,-(A7)  GO TO IO ROUTINE
         RTS                    NORMALLY GOES TO PORTIN10

PORTIN1N MOVEM.L D0-D4/D7/A0-A2,-(A7)
         TST.B   XONOFF+2
         BNE.S   PORTIN1V       OVERRIDE "No Auto Line Feed"
         MOVEQ   #1,D7          "No Auto Line Feed" Entry point
         BRA.S   PORTIN1W

* ENTRY ALSO POSSIBLE FROM:  PORTIN2
PORTIN10 MOVEM.L D0-D4/D7/A0-A2,-(A7)
PORTIN1V CLR.L   D7             "Auto Line Feed" Entry point.
PORTIN1W CLR.L   D3             FLAG AS PORT1
         BSR     GETSER1        MOVE ADDRESS INTO A0

READBUF  BSR     INCHNE         GO GET SOME DATA  (NO ECHO)
         TST.B   D0             CHECK FOR NULLS
         BEQ.S   READBUF
         MOVE.W  D0,D4          SAVE FOR A WHILE
         BSR     OUTCH1         ECHO WHAT IS IN D0
         MOVE.W  D4,D0          RESTORE IT
         CMPI.B  #LF,D0         SEE IF LINE FEED
         BEQ.S   READBUF        DONT PUT IT IN BUFFER

         CMPI.B  #CTLX,D0       SEE IF CTRL-X "CANCEL LINE"
         BNE.S   CHKCTLH        No... Then go check for backspace
         MOVEQ   #CR,D0         Yes.. 1) Send a carriage return
         BSR     OUTCH1         *
         MOVEQ   #LF,D0         *     2) Send a line feed
         BSR     OUTCH1         *
         MOVE.L  A5,A6          *     3) Set BUFFER at start again
         BRA.S   READBUF        *     4) Restart this routine

CHKCTLH  CMPI.B  #CTLH,D0       CTRL-H? (Backspace)
         BEQ.S   BS4CTLH        Yes...  Go backspace 1 character
         CMPI.B  #DEL,D0        DEL?    (Also a Backspace)
```

```
         BNE.S   CHKCTLD        No....  Then go check for CTRL D
BKSPACE  CMP.L   A5,A6          Yes...  Usually need an extra "Backspace"
         BEQ     READBUF        If at start of buffer, don't back up any more.
         MOVE.B  #CTLH,D0       Backspace cursor.
         BSR     OUTCH1         *
         BRA.S   BLANKIT        Bypass check for start of buffer

BS4CTLH  DS      0              CTL-H already backed up 1 character.
         CMP.L   A5,A6          At beginning of buffer?
         BEQ     READBUF        Yes...  Then don't back up any more.
BLANKIT  MOVEQ   #$20,D0        Now blank out previous character.
         BSR     OUTCH1         *
         MOVE.B  -(A6),D0       Remove last character from buffer.
         MOVE.B  #CTLH,D0       Backup cursor again.
         BSR     OUTCH1         *
         BRA     READBUF        Go get next character.

CHKCTLD  CMPI.B  #CTLD,D0       CTL D?  (REPRINT)
         BNE.S   CHKCR          No....  Then go check if a "CR".
         MOVEQ   #CR,D0         Yes...  Print "CR"
         BSR     OUTCH1         *
         MOVEQ   #LF,D0         Print "LF"
         BSR     OUTCH1         *
         MOVE.L  A5,-(A7)       SAVE ON STACK FOR A MOMENT
         BSR     OUTPUT         GO PRINT BUFFER
         MOVE.L  (A7)+,A5       GET BACK FROM STACK
         BRA     READBUF

CHARSAVE MOVE.B  D0,(A6)+       SAVE DATA INTO BUFFER
         MOVE.L  A6,D1          CALCULATE ROOM USED
         SUB.L   A5,D1          DONT USE MORE THAN "BUFFSIZE"
         CMPI.W  #BUFFSIZE,D1
         BMI     READBUF        We're OK, go get next character.
         BRA     BKSPACE        If not, handle as though a "CNTL-H"

CHKCR    CMPI.B  #CR,D0         CR?  (End of command line).
         BNE.S   CHARSAVE       No.. Then go store this character.

         TST.B   D7             Check "Auto Line Feed" switch
         BNE.S   NOAUTOLF       Yes.. Bypass sending Line feed.
         MOVEQ   #LF,D0         No... Set one up and...
         BSR     OUTCH1         .     Send it!
NOAUTOLF DS      0

 MOVEM.L (A7)+,D0-D4/D7/A0-A2 Restore Regs.
         RTS                    RETURN TO CALLER




*---------------------------------------------------------------------------
* File Y         TUTOR     I/O Routine                            05/19/82

LTIME    SET     205000         LONG TIMER 5 SEC @ 8 MHZ
STIME    SET     41000          SHORT TIMER  100 MLS @ 8 MHZ

PDI1     SET     $010000        PARALLEL PORT ADDRESS
PITCDDR  SET     $010009        PORT C DATA DIRECTION REGISTER
PITPCDR  SET     $010019        PORT C DATA REGISTER
PITTCR   SET     $010021        TIMER CONTROL REGISTER
PSTATUS  SET     $B             PRINTER STATUS
PBDATA   SET     3              PRINTER CONTROL--BUSY,PAPER,SELECT
PDATA    SET     1              PRINTER DATA
SER1     SET     $010040        TERMINAL
SER2     SET     $010041        SERIAL PORT2 ADDRESS

*        PRINTER DRIVER
*
*    SEND BUFFER TO PRINTER
*
PRCRLF   DS      0

         MOVEM.L A5-A6,-(A7)    SAVE REGISTERS
*
*   SEND LINE
*
LIST     CMP.L   A6,A5          SEE IF AT END OF BUFFER
         BMI.S   LIST1
```

```
         MOVEM.L (A7)+,A5-A6    RESTORE REGISTERS
         RTS
*
LIST1    MOVE.B  (A5)+,D0       GRAB BYTE
         BSR.S   CHRPRINT       PRINT CHAR
         BRA     LIST

* OUTPUT CHAR IN D0 TO PRINTER
CHRPRINT MOVEM.L D0/D1/D7/A0/A5/A6,-(A7) SAVE SOME REGISTERS
LIST2    BSR     CHKBRK         CHECK FOR BREAK

         MOVE.L  PDIPORT,A0     A0 = ADDRESS OF PORT

         MOVE.B  3(A7),D0       D0 = CHAR TO BE SENT
* CHANGE CONTROL CHARS TO "."
         ANDI.B  #$7F,D0
         CMPI.B  #CR,D0
         BEQ.S   LIST25         OK CARRIAGE RETURN
         CMPI.B  #LF,D0
         BEQ.S   LIST25         OK LINE FEED
         CMPI.B  #$20,D0
         BLT.S   LIST24
         CMPI.B  #$7F,D0
         BLT.S   LIST25
LIST24   MOVE.B  #'.',D0        MAKE CHAR A PERIOD
LIST25   DS      0

         MOVE.B  D0,PDATA(A0)   SEND DATA
         MOVE.B  #$68,PDI1+13   STROBE PRINTER

         MOVE.B  #$60,PDI1+13

*
*
LIST3    BSR     CHKBRK         CHECK FOR BREAK
         MOVE.B  PBDATA(A0),D0
         ANDI.B  #3,D0          PAPER OUT? DESELECTED?
         SUBQ.B  #1,D0
         BEQ.S   LIST5
         MOVE.W  CRTPNT,D7
         CLR.W   CRTPNT
         LEA     MSG007(PC),A5
         BSR     FIXDATA
         BSR     OUTPUT
*
*   WAIT FOR BREAK OR PRINTER TO BE READY
*
LERR1    BSR.S   CHKBRK
         MOVE.B  PBDATA(A0),D0
         ANDI.B  #3,D0
         SUBQ.B  #1,D0
         BNE     LERR1          PRINTER NOT READY
         MOVE.W  D7,CRTPNT      RESTORE POSSIBLE "PA" SWITCH
         BRA     LIST2          TRY AGAIN

LIST5    BTST.B  #0,PSTATUS(A0) ACKNOWLEDGE?
         BEQ.S   LIST3

         MOVEM.L (A7)+,D0/D1/D7/A0/A5/A6  RESTORE REGISTERS
         RTS

MSG007   DC.B    CR,LF,'PRINTER NOT READY',CR,LF,EOT




         DS.B    0

*
*       SEND CHARACTER IN D0.B TO SERIAL PORT IN (A0) (NO NULL PADS)
*
OUTCH    BSR.S   CHKBRK         CHECK FOR BREAK
         MOVE.B  (A0),D1        READ STATUS AGAIN
         ANDI.B  #$2,D1         CHECK FOR READY TO SEND
         BEQ.S   OUTCH          STILL NOT READY
         MOVE.B  D0,2(A0)       SEND CHARACTER  ****************

* IF PRINT FLAG SET GOTO PRINTER
```

```
         BEQ.S   OUTCH21        NULL; IGNORE SENDING TO PRINTER
         TST.W   CRTPNT
         BEQ.S   OUTCH21        CRT ONLY
         BSR     CHRPRINT       GOTO PRINTER
OUTCH21  DS      0

*   CHECK FOR CONTROL W
         MOVE.B  (A0),D1        READ STATUS
         ANDI.B  #1,D1
         BEQ.S   CTLW9          CHAR NOT READY
         MOVE.B  2(A0),D1       READ CHARACTER
         CMPI.B  #CTLW,D1
         BNE.S   CTLW9          NOT CNTL/W
CTLWH    BSR.S   CHKBRK         CHECK FOR BREAK
         MOVE.B  (A0),D1        READ STATUS
         ANDI.B  #1,D1
         BEQ     CTLWH          WAIT FOR ANY CHAR TO CONTINUE
CTLW9    RTS
*
*   CHECK FOR BREAK ON PORT #1
*
CHKBRK   MOVE.L  A0,-(A7)       SAVE A0 * * *
         BSR     GETSER1        RETURNS ADDRESS IN A0
         MOVE.B  (A0),D1        READ STATUS
         ANDI.B  #$10,D1
         BNE.S   BREAK
         MOVE.L  (A7)+,A0       RESTORE A0 * * *
         RTS

*    WHAT TO DO WHEN THE BREAK IS PRESSED
*

BREAK2   CLR.B   2(A0)          SEND NULL TO ACIA TO RESET
BREAK1   BTST.B  #1,(A0)        CHECK IF "TRANSMIT READY"
         BEQ.S   BREAK1         WAIT FOR READY

         MOVE.B  2(A0),D0       READ TWO CHARS
         MOVE.B  2(A0),D0       *

         BTST.B  #4,(A0)        SEE IF BREAK BUTTON RELEASED
         BNE     BREAK2         NO... KEEP LOOPING
         RTS

BREAK    BSR     BREAK2
         LEA     MSG013(PC),A5  MSG:  "BREAK"

BREAK79  BSR     FIXDCRLF       *
         BSR     OUTPUT         PRINT MESSAGE
         BRA     MACSBUG        AND REENTER MACSBUG

MSG013   DC.B    LF,LF,'BREAK',CR,LF,LF,EOT



         DC.B    0              PAD BYTE

***  OUTPUT BUFFER TO TAPE  ***
TAPEOUT  MOVEM.L D0-D4/A0-A1,-(A7)  SAVE REGISTERS
         MOVE.L  A5,A0          REMEMBER WHERE BUFFER STARTS
         MOVEA.L #PDI1,A1
         CLR.B   $21(A1)
         CMPI.W  #'S0',(A0)     HEADER RECORD?
         BNE.S   TAPEOUT2       NO
         MOVE.B  #2,9(A1)       YES, PC0 INPUT, PC1 OUTPUT
         MOVE.W  #700,D3        OUTPUT NULLS (HEADER)
TAPEOUT1 CLR.B   D0
         BSR.S   TAPEO
         SUBQ.W  #1,D3
         BNE.S   TAPEOUT1
TAPEOUT2 CMP.L   A6,A5          SEE IF AT OR BEYOND END OF LINE
         BCS.S   TAPEOUT4       NO. KEEP GOING.
         MOVE.B  TAPENULS,D3    OUTPUT NULLS AFTER EACH RECORD
TAPEOUT5 CLR.B   D0             YES. OUTPUT A NULL
         BSR.S   TAPEO
         SUBQ.B  #1,D3
         BNE.S   TAPEOUT5
         MOVEM.L (A7)+,D0-D4/A0-A1  RESTORE REGISTERS
         RTS
TAPEOUT4 MOVE.B  (A5)+,D0       GRAB BYTE TO OUTPUT
```

```
         BSR.S   TAPEO          GO OUTPUT IT
         BRA.S   TAPEOUT2       GO DO ANOTHER
*
* OUTPUTS THE CHARACTER IN D0.B TO TAPE
* A LOGIC `0' IS RECORDED AS ONE SQUARE WAVE PERIOD OF
*        1 MILLISEC DURATION, 50% DUTY CYCLE
* A LOGIC `1' IS RECORDED AS ONE SQUARE WAVE PERIOD OF
*        500 MICROSEC DURATION, 50% DUTY CYCLE
*
TAPEO    ORI.B   #%10000,CCR    SET X BIT IN SR
         ROXL.B  #1,D0          DATA BIT INTO X
TAPEO1   ROXL.B  #1,D2          DATA BIT INTO D2
         BSR.S   TIMERTST       WAIT UNTIL LAST PULSE DONE
         BCLR.B  #0,$21(A1)     HALT TIMER
         MOVEQ   #30,D1         TIMER COUNT FOR A `1'
         BTST.L  #0,D2          SENDING A `1'?
         BNE.S   TAPEO2         YES.
         ADDI.L  #32,D1         NO. TIMER COUNT FOR 0
TAPEO2   MOVEP.L D1,$25(A1)     SET TIMER PRELOAD REGISTER
         BSET.B  #1,$19(A1)     SEND 1 TO TAPE
         BSET.B  #0,$21(A1)     START TIMER
         BSR.S   TIMERTST       WAIT UNTIL PULSE DONE
         BCLR.B  #0,$21(A1)     HALT TIMER
         BCLR.B  #1,$19(A1)     SEND 0 TO TAPE
         BSET.B  #0,$21(A1)     START TIMER
         ASL.B   #1,D0          SENT 8 BITS?
         BNE     TAPEO1         NO. CONTINUE
         RTS
*
* WAITS UNTIL PROGRAMMED TIME DELAY HAS ELAPSED
* (IF TIMER IS RUNNING)
* ALSO CHECKS FOR BREAK
* USES D1
*
TIMERTST BSR     CHKBRK         CHECK FOR BREAK
         BTST.B  #0,$21(A1)     IS TIMER RUNNING?
         BEQ.S   TIMERTS1       NO. RETURN
         BTST.B  #0,$35(A1)     HAS TIME DELAY ELAPSED?
         BEQ.S   TIMERTST       NO. WAIT
TIMERTS1 RTS

*   IF FAULT THEN INITIALIZE AN ACIA
*
FAULTSER MOVEM.L D0/A0,-(A7)  FREE UP SOME WORKING REGISTERS

* DELAY TO ALLOW ACIA TO FINISH TRANSMITTING
*  LONGEST TIME FOR TWO CHARACTERS; 110 BAUD, 16MHZ NO WAIT STATES
         MOVE.L  #50000,D0      DELAY A WHILE
FAULTAC4 SUBQ.L  #1,D0          LOOP AROUND
         BNE     FAULTAC4

         BSR     GETSER1        MOVE ADDRESS INTO A0
         MOVE.B  (A0),D0        READ STATUS
         ANDI.B  #$70,D0        SEE IF FAULT
         BEQ.S   FAULTAC2
         MOVE.B  #RESET,(A0)    MASTER RESET
         MOVE.B  MD1CON,(A0)    HOW TO PROGRAM IT

FAULTAC2 BSR     GETSER2        MOVE ADDRESS INTO A0
         MOVE.B  (A0),D0
         ANDI.B  #$70,D0
         BEQ.S   FAULTAC3
         MOVE.B  #RESET,(A0)    MASTER RESET
         MOVE.B  MD1CON+1,(A0)  HOW TO PROGRAM IT
FAULTAC3
         MOVEM.L (A7)+,A0/D0    RESTORE REGISTERS
         RTS

*   INITIALIZE BOTH ACIAs
*
INITSER  MOVEM.L D0/A0,-(A7)  FREE UP SOME WORKING REGISTERS

* DELAY TO ALLOW ACIA TO FINISH TRANSMITTING
*  LONGEST TIME FOR TWO CHARACTERS; 110 BAUD, 16MHZ NO WAIT STATES
         MOVE.L  #50000,D0      DELAY A WHILE
INITAC3  SUBQ.L  #1,D0          LOOP AROUND
         BNE     INITAC3

         BSR     GETSER1        MOVE ADDRESS INTO A0
```

```
         MOVE.B  #RESET,(A0)    MASTER RESET
         MOVE.B  MD1CON,(A0)    HOW TO PROGRAM IT
         BSR     GETSER2        MOVE ADDRESS INTO A0
         MOVE.B  #RESET,(A0)    MASTER RESET
         MOVE.B  MD1CON+1,(A0)  HOW TO PROGRAM IT
         MOVEM.L (A7)+,A0/D0    RESTORE REGISTERS
         RTS

*    INPUT CHARACTER FROM PORT1
*    ACIA ADDRESS IN (A0)
*

INCHNE   MOVE.B  (A0),D1        (INCH NO ECHO) LOAD STATUS SIDE
         ANDI.B  #$10,D1        .              CHECK FOR BREAK
         BNE     BREAK          .              GO PROCESS IT

         MOVE.B  (A0),D1        LOAD STATUS SIDE
         ANDI.B  #1,D1          SEE IF READY
         BEQ.S   INCHNE         IF NOT READY
         MOVE.B  2(A0),D0       READ DATA SIDE   *****************
         ANDI.B  #$7F,D0        DROP PARITY BIT
         RTS

*  INPUT A LINE FROM PORT2 (ACIA2)
*   [ECHO TO PORT1 (ACIA1)]
*                  XONOFF
*                   0000     NULL  NULL    DO NOTHING
*                   0817   BKSPACE CNTL/W
*                   1214     X-ON  X-OFF   READER ON OFF
*
PORTIN2  TST.L   INFROM         SEE IF ALTERNATE ADDRESS
         BEQ.S   PORTIN2X
         MOVE.L  INFROM,-(A7)
         RTS                    GO DO IT
PORTIN2X MOVE.L  INPORT2,-(A7)  GO TO IO ROUTINE
         RTS                    (THIS NORMALLY GOES TO PORTIN20)
*
PORTIN20 MOVEM.L D1-D3/A0/A3,-(A7)  SAVE WORKING REGISTERS

         BSR     GETSER2
         MOVE.L  A0,A3          A3 = ADDRESS OF PORT 2
         BSR     GETSER1        A0 = ADDRESS OF PORT 1

         MOVE.L  #LTIME,D2 D2 = 5 SEC DEADMAN @ 8MHZ

PORT2100 BSR.S   P2READY
         BEQ.S   PORT2120       PORT2 DATA IN NOT-READY

         BSR.S   RES            READ/ECHO/STORE CHAR
         BRA     PORT2100

PORT2120 MOVEQ   #1,D3          D3 = MLS TIMER (ONE TICK)
PORT2130 BSR.S   P2READY
         BNE.S   PORT2250       PORT2 DATA IN READY

         SUBQ.L  #1,D2
         BMI.S   PORT2300       SECOND TIME OUT

         SUBQ.L  #1,D3
         BNE     PORT2130       MLS TIMER

PORT2140 MOVE.B  XONOFF,D0
         BEQ.S   PORT2150       NULL; IGNORE READER ON
         MOVE.B  (A3),D1
         ANDI.B  #2,D1
         BEQ     PORT2120       PORT2 DATA OUT NOT-READY
         MOVE.B  D0,2(A3)       SEND START READER CHAR
PORT2150

         MOVE.L  #STIME,D3      D3 = 100 MLS @ 8MHZ
         BRA     PORT2130

PORT2250 BSR.S   P2READY
         BEQ.S   PORT2260       PORT2 DATA IN NOT-READY
         BSR.S   RES            READ/ECHO/STORE CHAR

PORT2260 SUBQ.L  #1,D3
         BMI     PORT2140       CHARACTER TIME OUT
         BRA     PORT2250
```

```
PORT2300 LEA     MSG030(PC),A5  TIMEOUT ERROR
         BRA     BREAK79

MSG030   DC.B    'TIMEOUT',EOT


P2READY  MOVE.B  (A0),D1        CHECK FOR ACTIVITY ON PORT1
         ANDI.B  #$10,D1        CHECK FOR BREAK
         BNE     BREAK
         MOVE.B  2(A0),D1       READ POSSIBLE CHAR PORT 1; IGNORE

         MOVE.B  (A3),D1        READ STATUS OF PORT2
         ANDI.B  #1,D1          SEE IF CHARACTER SENT
         RTS

RES      MOVE.B  2(A3),D1       D1 = CHAR READ FROM PORT2
         ANDI.B  #$7F,D1        DROP PARITY BIT

         TST.W   ECHOPT1        SEE IF ECHO ON
         BEQ.S   RES140
         MOVE.B  D1,2(A0)       SEND TO DATA SIDE (PORT1)
RES140

         CMPI.B  #CR,D1
         BEQ.S   RES190         END OF LINE

         CMPI.B  #$20,D1        IGNORE CONTROL CHARACTERS (NULLS)
         BLT.S   RES150
         MOVE.B  D1,(A6)        SAVE CHAR IN BUFFER

         MOVE.L  #LTIME/2,D3    D3 = 2 1/2 SEC @ 8MHZ (CHARATER  TIMER)

         MOVE.L  A6,D1          CHECK BUFFER FULL
         SUB.L   A5,D1
         CMPI.W  #BUFSIZE,D1
         BPL.S   RES150         FULL
         ADDQ.L  #1,A6          INCREMENT STORE POINTER
RES150   RTS


RES190   CMP.L   A5,A6
         BEQ     RES150         NULL RECORD; IGNORE

         ADDQ.L  #4,A7          TRIM STACK

         MOVE.B  XONOFF+1,D0
         BEQ.S   RES195         XOFF = NULL CHAR; IGNORE
RES194   MOVE.B  (A3),D1
         ANDI.B  #2,D1
         BEQ     RES194         PORT2 DATA OUT NOT-READY
         MOVE.B  D0,2(A3)       STOP READER
RES195

         MOVEM.L (A7)+,D1-D3/A0/A3   RESTORE THE REGISTERS
         RTS

*
* INPUT A LINE FROM AUDIO TAPE (PI/T)
* [ECHO TO PORT1 (ACIA1)]
*
TAPEIN   MOVEM.L D1-D4/A0-A3,-(A7)  SAVE WORKING REGISTERS
         BSR     GETSER1        ACIA1 ADDRESS INTO A0
         MOVEA.L #PDI1,A1
         MOVE.B  #2,PITCDDR     PC0 INPUT, PC1 OUTPUT.

* SYNCHRONIZE ON S CHARACTER
         CLR.B   D1
TAPEIN10 BSR.S   TAPEIN40       GET TAPE
         BCS.S   TAPEIN10       WAIT FOR LOW
TAPEIN11 BSR.S   TAPEIN40       GET TAPE
         BCC.S   TAPEIN11       WAIT FOR HIGH
         BSR.S   STARTIMR       START TIMER
TAPEIN12 ASL.B   #1,D1
         BSR.S   TAPEIN30       GET ONE BIT FROM TAPE
         CMPI.B  #'S',D1        S?
         BNE.S   TAPEIN12       NO. GET NEXT BIT
         BRA.S   TAPEIN51
```

```
* GET ONE CHARACTER FROM THE TAPE
TAPEIN20 MOVEQ   #2,D1          SET STOP BIT
TAPEIN21 BSR.S   TAPEIN30       GET 1 BIT FROM TAPE
         ASL.B   #1,D1          STOP IN CARRY
         BCC.S   TAPEIN21       NO
*FALL INTO LOAD BIT ROUTINE FOR LAST BIT
TAPEIN30 BSR.S   TAPEIN40       GET TAPE
         BCS.S   TAPEIN30       WAIT FOR LOW
TAPEIN31 BSR.S   TAPEIN40       GET TAPE
         BCC.S   TAPEIN31       WAIT FOR HIGH
         CLR.B   PITTCR         STOP TIMER
         MOVEP.L $2D(A1),D3     GET PERIOD MEASUREMENT
         BSR.S   STARTIMR       START TIMER
         SUBI.L  #$FFFFFF-94,D3 IS IT A LOGIC 1?
         BLO.S   TAPEIN32       NO
         ADDQ.B  #1,D1          YES. STORE LOGIC 1
TAPEIN32 RTS

* READ THE TAPE LEVEL INTO THE CARRY AND CHECK FOR BREAK
TAPEIN40 MOVE.B  (A0),D2        CHECK FOR ACTIVITY ON PORT1
         ANDI.B  #$10,D2        CHECK FOR BREAK
         BNE     BREAK
         MOVE.B  (A0),D2        SEE IF CHARACTER SENT
         ANDI.B  #1,D2
         BEQ.S   TAPEIN41       NONE SENT.
         MOVE.B  2(A0),D2       READ WHAT WAS SENT
TAPEIN41 MOVE.B  PITPCDR,D2     READ PI/T
         ASR.B   #1,D2          DATA INTO CARRY
         RTS

*STARTS PROGRAMMABLE TIMER
STARTIMR MOVE.L  #$00FFFFFF,D4  INIT. COUNT. PRELOAD REG.
         MOVEP.L D4,$25(A1)
         MOVE.B  #1,PITTCR      START TIMER

         RTS

TAPEIN53 CMPI.B  #$20,D1
         BLT.S   TAPEIN50       IGNORE CONTROL CHARACTERS
         MOVE.B  D1,(A6)        SAVE CHARACTER IN BUFFER
         MOVE.L  A6,D1          CHECK BUFFER FULL
         SUB.L   A5,D1
         CMPI.W  #BUFSIZE,D1
         BPL.S   TAPEIN50       FULL
         ADDQ.L  #1,A6          INCREMENT BUFFER POINTER

TAPEIN50 BSR     TAPEIN20       GET ONE CHARACTER FROM TAPE
TAPEIN51 ANDI.B  #$7F,D1        DROP PARITY BIT
         TST.W   ECHOPT1        SEE IF ECHO ON
         BEQ.S   TAPEIN52
         MOVE.B  D1,2(A0)       SEND TO PORT1
TAPEIN52 CMPI.B  #LF,D1         END OF LINE?
         BNE.S   TAPEIN53       NO.
         MOVEM.L (A7)+,D1-D4/A0-A3  RESTORE REGISTERS
         RTS

*     SCAN COMMAND LINE FOR PORT NUMBER
*     DU LO VE MD
*
SCANPORT CLR.L   INFROM         DEFAULT IS WHATEVER IS CALLED
         CLR.L   OUTTO
         MOVE.B  (A5),D0        NO....GET CHARACTER
         CMPI.B  #BLANK,D0      .     IS IT A BLANK?
         BEQ.S   SCANP6         .     YES..EXIT

         CMPI.B  #'1',D0        SEE IF PORT 1 OVERRIDE
         BNE.S   SCANP4
         MOVE.L  INPORT1,INFROM
         MOVE.L  OUTPORT1,OUTTO
         BRA.S   SCANP7

SCANP4   CMPI.B  #'2',D0        SEE IF PORT 2 OVERRIDE
         BNE.S   SCANP5
         MOVE.L  INPORT2,INFROM
         MOVE.L  OUTPORT2,OUTTO
         BRA.S   SCANP7

SCANP5   CMPI.B  #'3',D0        SEE IF PORT3 OVERRIDE
         BNE.S   SCANP8
```

```
         MOVE.L  INPORT3,INFROM
         MOVE.L  OUTPORT3,OUTTO
         BRA.S   SCANP7

SCANP8   CMPI.B  #'4',D0        SEE IF PORT4 OVERRIDE
         BNE.S   SCANP6
         MOVE.L  INPORT4,INFROM
         MOVE.L  OUTPORT4,OUTTO

SCANP7   ADDQ.L  #1,A5          ADJUST GET CHAR POINTER
SCANP6   RTS

*       *  P A  *      "PRINTER ATTACH" COMMAND          *
*                                                        *
* IF THE "PRINTER" HAS BEEN ATTACHED, ALL CHARACTERS     *
* ENTERED AND TRANSMITTED VIA THE CONSOLE WILL PRINT ON  *
* THE HARD COPY PRINTER. -- TO TERMINATE, ENTER "NOPA"   *
*                                                        *
* THIS COMMAND MAKES USE OF THE "NO" OPTION.  THERE IS A *
* 4 BYTE HARD BRANCH INFRONT OF THE REGULAR ENTRY POINT. *
**********************************************************

         BRA.L   NOPACMD        ENTRY FOR "NOPA"

*
PACMD    MOVEQ   #-1,D0         ENTRY FOR "PA" (D0=-1[CRT & PRINT])
         BRA.S   SETCRTPR       GO ACTUALLY SET THE SWITCH
NOPACMD  CLR.L   D0             D0 = ZEROS... "CRT" ONLY
SETCRTPR MOVE.W  D0,CRTPNT      SET THE "CRT" AND "PRINTER" SWITCH
         BRA     MACSBUG        GO BE "READY"




*------------------------------------------------------------------------
* File CODE68K   68K ONE LINE ASSEMBLER                          07/23/82

* EVALUATE EXPRESSION
*  NUMBER PLUS OR MINUS NUMBER....
*
EV       DS      0
         MOVE.L  D7,-(A7)       SAVE D7
         CLR.L   D7
EV21     BSR.S   GETFIELD       GET NUMBER
         ADD.L   D0,D7          D7 = NUMBER BEING BUILT
EV15     MOVE.B  (A5)+,D1       D1 = TERMINATING CHAR
         CLR.L   D0             D0 = NEXT NUMBER (=0 1ST TIME)
         CMPI.B  #'+',D1
         BEQ     EV21           PLUS
         CMPI.B  #'-',D1
         BNE.S   EV39           NOT MINUS
         BSR.S   GETFIELD       GET NEXT NUMBER
         SUB.L   D0,D7
         BRA     EV15

EV39     MOVE.L  D7,D0          D0 = VALUE BUILT
         SUBQ.L  #1,A5          A5 = CHAR AFTER EXPRESSION
         MOVE.L  (A7)+,D7       RESTORE D7
         RTS

GETFIELD DS      0
         CMPI.B  #'*',(A5)
         BNE.S   GETF305

         MOVE.L  D4,D0          D0 = PROGRAM COUNTER
         ADDQ.L  #1,A5
         BRA.S   GETF333

GETF305  CMPI.B  #$27,(A5)
         BNE.S   GETF325        NOT LITERAL

         ADDQ.L  #1,A5
         CLR.L   D0

         MOVE.W  TLENGTH(A1),D1  D1 = SIZE
         BEQ.S   GETF308        .B = 0
         LSR.W   #5,D1          .W = 1
         SUBQ.L  #1,D1          .L = 3
GETF308
```

```
GETF311  LSL.L   #8,D0
         MOVE.B  (A5)+,D0
         CMPI.B  #$27,(A5)
         BEQ.S   GETF312         CLOSING QUOTE
         DBF     D1,GETF311
         BRA.S   ER1             OVERFLOW

GETF312  ADDQ.L  #1,A5           MOVE PAST CLOSING QUOTE
         BRA.S   GETF314

GETF313  LSL.L   #8,D0
GETF314  DBF     D1,GETF313      LEFT NORNALIZE
         BRA.S   GETF333

GETF325  BSR     GETDECNU        GET DECIMAL NUMBER
         BNE.S   ER1             MESSED UP NUMBER

GETF333  RTS

XBASE    DS        0

* FIND AND SET SIZE
* BIT 5432109876543210
*     ........00......  = BYTE
*     ........01......  = WORD
*     ........10......  = LONG
*
FSIZE    OR.W    TLENGTH(A1),D2  SET SIZE BITS
         RTS

*  D0 = VALUE 0 - 7
*  D1 = 0 IF D@     = 1 IF A@
GETREGD  CLR.L   D1
         CMPI.B  #'D',(A5)+
         BNE.S   ER1
GET41    CLR.L   D0
         MOVE.B  (A5)+,D0
         SUBI.B  #'0',D0
         BMI.S   ER1
         CMPI.B  #$7,D0
         BGT.S   ER1
         RTS

GETREGA  CLR.L   D1
         MOVE.B  #8,D1
         CMPI.B  #'A',(A5)+
         BNE.S   ER1
         BRA     GET41

GETREGAD CLR     D1
         MOVE.B  (A5)+,D0
         CMPI.B  #'D',D0
         BEQ     GET41
         MOVE.B  #8,D1
         CMPI.B  #'A',D0
         BEQ     GET41
ER1      BRA     ER

EADA     MOVE.W  #$1FD,D7        DATA ALTERABLE ONLY
         BRA.S   EA

EAC      MOVE.W  #$7E4,D7        CONTROL ONLY
         BRA.S   EA

EAM      MOVE.W  #$1FC,D7        MEMORY ALTERABLE ONLY
         BRA.S   EA

EAZ      MOVE.W  #$800,D7        IMMEDIATE ONLY
         BRA.S   EA

EADADDR  MOVE.W  #$FFD,D7        DATA ADDRESSING
         BRA.S   EA

EAA      MOVE.W  #$1FF,D7        ALTERABLE ADDRESSING
         BRA.S   EA

EAALL    MOVE.W  #$FFF,D7        ALL MODES
```

```
*  ...............1  D@               DATA REGISTER
*  ..............1.  A@
*  .............1..  (A@)
*  ............1...  -(A@)
*  ...........1....  (A@)+
*  ..........1.....  DATA(A@)
*  .........1......  DATA(A@,R@)
*  ........1.......  DATA  (SHORT)
*  .......1........  DATA  (LONG)
*  ......1.........  DATA(PC)
*  .....1..........  DATA(PC,R@)
*  ....1...........  #DATA
*  1...............  SPECIAL CASE JMP.L

* D0 = VALUE CALCULATED
* D2 = MASK WORD (1ST WORD OF INSTRUCTION)
* D3 = OFFSET FOR DATA STORE (TDATA+..)
* D4 = EXTENSION WORD
* D5 = <DATA>
* D6 = MODE AS BUILT   .........XXXXXX
* D7 = MODES ALLOWED
*
* A4 = BASE ADDRESS FOR DATA STORE (TDATA+..)[A4,D3]
EA       DS      0
         CLR.L   D5              ZERO VALUE
         CLR.L   D6              MODE = 000000
         MOVE.B  (A5),D0
         CMPI.B  #'#',D0
         BNE.S   EA10

* IMMEDIATE MODE

         BTST    #11,D7
         BEQ     ER1

         MOVE.B  #$3C,D6         D6 = MODE  111100
         ADDQ.L  #1,A5

         BSR     EV              EVALUATE EXPRESSION
         MOVE.L  D0,D5           D5 = VALUE

         TST.B   TLSPEC(A1)
         BEQ.S   EA0633          .SIZE NOT SPECIFIED (.W ASSUMED)

         MOVE.W  TLENGTH(A1),D0
         BEQ.S   EA0635          .BYTE

         TST.B   D0
         BMI.S   EA0637          .LONG

EA0633   BSR     EA16BIT         .WORD     -32K TO +64K
EA0634   MOVE.W  D5,(A4,D3)
         ADDQ.B  #2,TNB(A1)      BYTE COUNT
         ADDQ.L  #2,D3           OFFSET
         RTS

EA0635   BSR     EA8BIT          -127 TO +255
         BNE     ER1
         BRA     EA0634

EA0637   MOVE.L  D5,(A4,D3)
         ADDQ.B  #4,TNB(A1)
         ADDQ.L  #4,D3
         RTS

EA10     DS      0
         CMPI.B  #'-',(A5)
         BNE.S   EA11

         CMPI.B  #'(',1(A5)
         BNE     EA41            MAY BE "-<DATA>

         ADDQ.L  #2,A5
         MOVE.W  #$0020,D6       MODE = -(A@)    100AAA

         BTST    #4,D7
         BEQ     ER1             THIS MODE NOT ALLOWED

         BSR     GETREGA
```

```
         OR.W    D0,D6

         CMPI.B  #')',(A5)+
         BNE.S   ER3             NO CLOSING PAREN
         RTS

EA11     CMPI.B  #'A',D0
         BNE.S   EA21

         MOVE.B  #$08,D6         MODE = 001...
         BTST    #1,D7
         BEQ.S   ER3             MODE NOT ALLOWED

         BSR     GETREGA
         OR.W    D0,D6
         RTS

EA21     CMPI.B  #'D',D0
         BNE.S   EA31

         BTST    #0,D7
         BEQ.S   ER3             MODE NOT ALLOWED

         BSR     GETREGD         MODE = D@   000AAA
         OR.W    D0,D6
         RTS

EA31     CMPI.B  #'(',D0
         BNE.S   EA41

* POSSIBLE
*  (A@)
*  (A@)+
*  (A@,R@)  IMPLIED ZERO DISPLACEMENT
*
         ADDQ.L  #1,A5
         BSR     GETREGA
         OR.W    D0,D6

         MOVE.B  (A5)+,D0
         CMPI.B  #',',D0
         BEQ     EA5116          MODE = (A@,R@)  ;IMPLIED D5 = 0 DATA

         CMPI.B  #')',D0         LOOK FOR CLOSING )
         BNE.S   ER3

         CMPI.B  #BLANK,(A5)     LOOK FOR BLANK
         BEQ.S   EA35            MODE = (A@)

         CMPI.B  #'+',(A5)
         BNE.S   EA35
         ADDQ.L  #1,A5

         ORI.W   #$18,D6         MODE = 011...    (A@)+

         BTST    #3,D7
         BEQ.S   ER3             MODE NOT ALLOWED

EA34     RTS

EA35     ORI.W   #$10,D6         MODE = 010...   (A@)

         BTST    #2,D7
         BNE     EA34            MODE ALLOWED
ER3      BRA     ER              MODE NOT ALLOWED

* POSSIBLE
*   <DATA>   SHORT
*   <DATA>   LONG
*   <DATA>(A@)
*   <DATA>(A@,R@)
*   <DATA>(A@,R@.W)
*   <DATA>(A@,R@.L)
*   <DATA>(PC)
*   <DATA>(PC,R@)
*   <DATA>(PC,R@.W)
*   <DATA>(PC,R@.L)
*
EA41     BSR     EV              EVALUATE EXPRESSION
```

```
         MOVE.L  D0,D5          D5 = <DATA>

         MOVE.B  (A5),D0
         CMPI.B  #',',D0
         BEQ.S   EA4102
         CMPI.B  #BLANK,D0
         BNE.S   EA4120

*  <DATA>         ONLY
* CHECK IF NEGATIVE NUMBER

EA4102   MOVE.L  D5,D0
         BPL.S   EA4105         POSITIVE NUMBER
         NOT.L   D0
EA4105   ANDI.W  #$8000,D0
         TST.L   D0
         BNE.S   EA4135         .LONG

*  <DATA>.W
         BTST    #7,D7
         BNE.S   EA4127         SHORT ALLOWED
         BTST    #15,D7
         BEQ     ER3            MODE NOT ALLOWED
         BRA.S   EA4135         SPECIAL CASE (JMP.L)

EA4127   ORI.W   #$38,D6        EA = ABSOULTE SHORT
         MOVE.W  D5,(A4,D3)     D5 = DATA
         ADDQ.B  #2,TNB(A1)     BYTE COUNT
         ADDQ.L  #2,D3
         RTS

*EA4134  CMPI.B  #'L',D0
*        BNE     ER3

*  <DATA>.L
EA4135   ORI.W   #$39,D6        EA = ABSOLUTE LONG
         MOVE.L  D5,(A4,D3)
         ADDQ.B  #4,TNB(A1)     BYTE COUNT
         ADDQ.L  #4,D3
         BTST    #8,D7
         BEQ     ER3            MODE NOT ALLOWED
         RTS

EA4120   ADDQ.L  #1,A5
         CMPI.B  #'(',D0
         BNE     ER3

         CMPI.B  #'P',(A5)
         BEQ     EA61

* <DATA>(A@.......
         BSR     EA16BITS       -32K TO +32K

         BSR     GETREGA
         OR.W    D0,D6

         MOVE.B  (A5),D0
         CMPI.B  #')',D0
         BNE.S   EA5115

* <DATA>(A@)
         ADDQ.L  #1,A5

         BTST    #5,D7
         BEQ     ER4            MODE NOT ALLOWED

         ORI.W   #$0028,D6      MODE = 101AAA

         CMPI.L  #$10000,D5
         BPL     ER4

         MOVE.W  D5,(A4,D3)
         ADDQ.B  #2,TNB(A1)
         ADDQ.L  #2,D3
         RTS

EA5115   BSR     COMMA

*  <DATA>(A@,-----    ADDRESS REGISTER WITH INDEX
```

```
EA5116   EXT.L   D5
         BSR     EA8BITS        -128 TO +127
         BNE.S   ER4
         ANDI.W  #$00FF,D5
         ORI.W   #$0030,D6      MODE  110---

         BTST    #6,D7
         BEQ.S   ER4            MODE NOT ALLOWED

         BSR     GETREGAD
         OR.W    D1,D0
         ROR.W   #4,D0
         OR.W    D0,D5          EXTENSION WORD

* BIT 11 EXTENSION WORD
*   0 = SIGN EXTENDED, LOW ORDER INTEGER IN INDEX REGISTER
*   1 = LONG VALUE IN INDEX REGISTER  (DEFAULT)
*
         MOVE.B  (A5)+,D0
         CMPI.B  #')',D0
         BEQ.S   EA5119         DEFAULT   .W

         CMPI.B  #'.',D0
         BNE.S   ER4

         MOVE.B  (A5)+,D0
         CMPI.B  #'W',D0
         BEQ.S   EA5118

         CMPI.B  #'L',D0
         BNE.S   ER4            NEITHER .W NOR .L

         ORI.W   #$0800,D5      EXTENSION WORD, W/L BIT = .L

EA5118   CMPI.B  #')',(A5)+
         BNE.S   ER4            NO CLOSING ")"

EA5119   MOVE.W  D5,(A4,D3)
         ADDQ.B  #2,TNB(A1)
         ADDQ.L  #2,D3
EA5119E  RTS

*  <DATA>(P-----
EA61     ADDQ.L  #1,A5
         CMPI.B  #'C',(A5)+
         BNE     ER

         SUB.L   PCOUNTER(A1),D5  D5 = D5 - PC
         SUBQ.L  #2,D5          D5 = D5 - (PC + 2)

         MOVE.B  (A5)+,D0
         CMPI.B  #')',D0
         BNE.S   EA71

*  <DATA>(PC)
         ORI.W   #$3A,D6        MODE = 111010

         BSR.S   EA16BITS       -32K TO +32K
         MOVE.W  D5,(A4,D3)
         ADDQ.B  #2,TNB(A1)
         ADDQ.L  #2,D3

         BTST    #9,D7
         BNE     EA5119E
ER4      BRA     ER

*  <DATA>(PC----          PROGRAM COUNTER WITH INDEX
EA71     MOVE.W  #$003B,D6      MODE = 111011

         CMPI.B  #',',D0
         BNE     ER4

         BTST    #10,D7
         BEQ     ER4            MODE NOT ALLOWED

         BSR.S   EA8BITS        -128 TO +127
         BNE     ER4

         ANDI.W  #$00FF,D5      D5 = VALUE
```

```
         BSR     GETREGAD
         OR.W    D1,D0

         ROR.W   #4,D0
         OR.W    D0,D5          D5 = EXTENSION WORD

         MOVE.B  (A5)+,D0
         CMPI.B  #')',D0
         BEQ.S   EA7115         DEFAULT  .W

         CMPI.B  #'.',D0
         BNE     ER4

         MOVE.B  (A5)+,D0
         CMPI.B  #'W',D0
         BEQ.S   EA7113

         CMPI.B  #'L',D0
         BNE     ER4
         ORI.W   #$0800,D5      EXTENSION WORD W/L = .L

EA7113   CMPI.B  #')',(A5)+
         BNE     ER4            NO CLOSING ")"

EA7115   MOVE.W  D5,(A4,D3)
         ADDQ.B  #2,TNB(A1)
         ADDQ.L  #2,D3
         RTS

*******************************************************************************
*
*        ROUTINES TO TEST FOR VALID EFFECTIVE ADDRESSES
*
*        EA16BIT   tests that -32768 <= D5 <= 65535.  (signed or unsigned)
*        EA16BITS  tests that -32768 <= D5 <= 32767.  (signed only)
*        EA8BIT    tests that   -128 <= D5 <=   255.  (signed or unsigned)
*        EA8BITS   tests that   -128 <= D5 <=   127.  (signed only)
*
*        The 16-bit tests branch to ER if invalid, else return.
*        The  8-bit tests return condition codes <EQ> if valid, else <NE>.
*        D5 is preserved unless a branch to ER results.
*        D1 is destroyed.
*
*******************************************************************************

EA16BIT:
         BSR.S   EA16BITC       CHECK RANGE -32768 TO 32767.  IF
         MOVE.L  D5,D1          INVALID, CHECK WHETHER THE HIGH 16
         SWAP    D1             BITS ARE 0 (WHICH IMPLIES THAT
         TST.W   D1             D5 <= 65535).  IF NOT, FALL THRU TO
         BEQ.S   EASEX          THE 16-BIT SIGNED TEST--WE WILL
*                               FAIL AND GO TO ER.

EA16BITS:
         PEA     ER(PC)         SET UP TO RETURN TO ER IF INVALID.

EA16BITC MOVE.L  #$7FFF,D1      D1  <--  2^16-1.
         BRA.S   EAS            GO TO THE COMMON TEST ROUTINE.


EA8BIT:
         BSR.S   EA8BITC        CHECK RANGE -128 TO 127.  IF INVALID,
         MOVE.L  D5,D1          CHECK WHETHER THE HIGH 24 BITS ARE
         LSR.L   #8,D1          0 (WHICH IMPLIES THAT D5 <= 255).
         RTS

EA8BITS:
         BSR.S   EA8BITC        JUST CHECK FOR -127 <= D5 <= 128.
         RTS                    (BSR PUTS NEEDED ADDRESS ON STACK!)

EA8BITC  MOVEQ   #$7F,D1        D1  <--  2^8 - 1.

*                 *** NOTE: THIS ROUTINE PLAYS WITH THE STACK ***
EAS      CMP.L   D1,D5          IF D5 > 2^N-1, RETURN WITH <NE> (INVAL).
         BGT.S   EASEX
         NOT.L   D1             IF D5 < -2^N,  RETURN WITH <NE> (INVAL).
         CMP.L   D1,D5
         BLT.S   EASEX
```

```
         ADDQ.L  #4,A7          POP THE RETURN ADDRESS OFF THE STACK,
         CLR.L   D1             SET <EQ> (VALID), AND RETURN.

EASEX    RTS

ADR      MACRO
         DC.W    M\1-XBASE
         ENDM

TBLKEYS  DS      0              INDEX
         ADR     ABCD            0  ABCD SBCD
         ADR     ADD             1  ADD  SUB
         ADR     ADDA            2  ADDA CMPA SUBA
         ADR     ADDI            3  ADDI CMPI SUBI
         ADR     ADDQ            4  ADDQ SUBQ
         ADR     ADDX            5  ADDX SUBX
         ADR     AND             6  AND  EOR OR
         ADR     ASL             7  ASL  LSL ROL ROXL
         ADR     DBCC            8  DBCC
         ADR     BCHG            9  BCHG
         ADR     BRA            10  BRA  BSR  BCC
         ADR     BSET           11  BSET
         ADR     CHK            12  CHK  DIVS DIVU MILS MULU
         ADR     CLR            13  CLR NEG NEGX NOT TST
         ADR     CMPM           14  CMPM
         ADR     MOVEQ          15  MOVEQ
         ADR     EXG            16  EXG
         ADR     EXT            17  EXT
         ADR     JMP            18  JMP  JSR
         ADR     LEA            19  LEA
         ADR     LINK           20  LINK
         ADR     MOVE           21  MOVE
         ADR     CMMD2          22  NOP RESET RTE RTR RTS TRAPV
         ADR     STOP           23  STOP
         ADR     SWAP           24  SWAP
         ADR     TRAP           25  TRAP
         ADR     UNLK           26  UNLK
         ADR     MOVEM          27  MOVEM
         ADR     ANDI           28  ANDI EORI ORI
         ADR     SCC            29  NBCD SCC TAS
         ADR     BCLR           30  BCLR
         ADR     BTST           31  BTST
         ADR     MOVEA          32  MOVEA
         ADR     MOVEP          33  MOVEP
         ADR     CMP            34  CMP
         ADR     EOR            35  EOR
         ADR     PEA            36  PEA
         ADR     DC             37  DC.W


* \1,\2 = MNEUMONIC (\2 SIGN BIT SET AS LAST CHARACTER)
* \3    = INDEX TO TABLKEYS
* \4,\5 = FIRST WORD MASK
* \6    = NO OPERAND ALLOWED IF SIGN SET
* \7    = .S ALLOWED (.W NOT ALLOWED)
OPC      MACRO
         DC.B    '\1','\2'+128
         DC.B    \3+\6+\7,$\4,$\5
         ENDM

NOC      EQU     $80            (BIT 7 SET) NO OPERAND
NW       EQU     $40            (BIT 6 SET) .W NOT ALLOWED

TBLOPC   OPC     ABC,D,0,C1,00,0,0     ABCD
         OPC     ADD,A,2,D0,C0,0,0     ADDA
         OPC     ADD,I,3,06,00,0,0     ADDI
         OPC     ADD,Q,4,50,00,0,0     ADDQ
         OPC     ADD,X,5,D1,00,0,0     ADDX
         OPC     AD,D,1,D0,00,0,0      ADD
         OPC     AND,I,28,02,00,0,0    ANDI
         OPC     AN,D,6,C0,00,0,0      AND
         OPC     AS,L,7,E1,00,0,0      ASL
         OPC     AS,R,07,E0,00,0,0     ASR
         OPC     BR,A,10,60,00,0,NW    BRA
         OPC     BH,I,10,62,00,0,NW    BHI
         OPC     BL,S,10,63,00,0,NW    BLS
         OPC     BC,C,10,64,00,0,NW    BCC
         OPC     BC,S,10,65,00,0,NW    BCS
         OPC     BN,E,10,66,00,0,NW    BNE
```

```
OPC     BE,Q,10,67,00,0,NW    BEQ
OPC     BV,C,10,68,00,0,NW    BVC
OPC     BV,S,10,69,00,0,NW    BVS
OPC     BP,L,10,6A,00,0,NW    BPL
OPC     BM,I,10,6B,00,0,NW    BMI
OPC     BG,E,10,6C,00,0,NW    BGE
OPC     BL,T,10,6D,00,0,NW    BLT
OPC     BG,T,10,6E,00,0,NW    BGT
OPC     BL,E,10,6F,00,0,NW    BLE
OPC     BCH,G,9,01,40,0,0     BCHG
OPC     BCL,R,30,01,80,0,0    BCLR      DYNAMIC
OPC     BSE,T,11,01,C0,0,0    BSET
OPC     BS,R,10,61,00,0,NW    BSR
OPC     BTS,T,31,01,00,0,0    BTST
OPC     B,T,10,60,00,0,NW     BT
OPC     CH,K,12,41,80,0,0     CHK
OPC     CL,R,13,42,00,0,0     CLR
OPC     CMP,A,2,B0,C0,0,0     CMPA
OPC     CMP,I,3,0C,00,0,0     CMPI
OPC     CMP,M,14,B1,08,0,0    CMPM
OPC     CM,P,34,B0,00,0,0     CMP
OPC     DB,T,8,50,C8,0,NW     DBT
OPC     DB,F,8,51,C8,0,NW     DBF
OPC     DBR,A,8,51,C8,0,NW    DBRA
OPC     DBH,I,8,52,C8,0,NW    DBHI
OPC     DBL,S,8,53,C8,0,NW    DBLS
OPC     DBC,C,8,54,C8,0,NW    DBCC
OPC     DBC,S,8,55,C8,0,NW    DBCS
OPC     DBN,E,8,56,C8,0,NW    DBNE
OPC     DBE,Q,8,57,C8,0,NW    DBEQ
OPC     DBV,C,8,58,C8,0,NW    DBVC
OPC     DBV,S,8,59,C8,0,NW    DBVS
OPC     DBP,L,8,5A,C8,0,NW    DBPL
OPC     DBM,I,8,5B,C8,0,NW    DBMI
OPC     DBG,E,8,5C,C8,0,NW    DBGE
OPC     DBL,T,8,5D,C8,0,NW    DBLT
OPC     DBG,T,8,5E,C8,0,NW    DBGT
OPC     DBL,E,8,5F,C8,0,NW    DBLE
OPC     DC.,W,37,00,00,0,0    DC.W  (WORD ONLY)
OPC     DIV,S,12,81,C0,0,0    DIVS
OPC     DIV,U,12,80,C0,0,0    DIVU
OPC     EOR,I,28,0A,00,0,0    EORI
OPC     EO,R,35,B1,00,0,0     EOR
OPC     EX,G,16,C1,00,0,0     EXG
OPC     EX,T,17,48,00,0,0     EXT
OPC     JM,P,18,4E,C0,0,NW    JMP
OPC     JS,R,18,4E,80,0,NW    JSR
OPC     LE,A,19,41,C0,0,0     LEA
OPC     LIN,K,20,4E,50,0,0    LINK
OPC     LS,L,7,E3,08,0,0      LSL
OPC     LS,R,07,E2,08,0,0     LSR
OPC     MOVE,A,32,00,04,0,0   MOVEA
OPC     MOVE,M,27,48,80,0,0   MOVEM
OPC     MOVE,P,33,01,08,0,0   MOVEP
OPC     MOVE,Q,15,70,00,0,0   MOVEQ
OPC     MOV,E,21,00,00,0,0    MOVE
OPC     MUL,S,12,C1,C0,0,0    MULS
OPC     MUL,U,12,C0,C0,0,0    MULU
OPC     NBC,D,29,48,0,0,0     NBCD
OPC     NEG,X,13,40,00,0,0    NEGX
OPC     NE,G,13,44,00,0,0     NEG
OPC     NO,P,22,4E,71,NOC,0   NOP
OPC     NO,T,13,46,00,0,0     NOT
OPC     OR,I,28,00,00,0,0     ORI
OPC     O,R,6,80,00,0,0       OR
OPC     PE,A,36,48,40,0,0     PEA
OPC     RESE,T,22,4E,70,NOC,0 RESET
OPC     RO,L,7,E7,18,0,0      ROL
OPC     RO,R,07,E6,18,0,0     ROR
OPC     ROX,L,7,E5,10,0,0     ROXL
OPC     ROX,R,07,E4,10,0,0    ROXR
OPC     RT,E,22,4E,73,NOC,0   RTE
OPC     RT,R,22,4E,77,NOC,0   RTR
OPC     RT,S,22,4E,75,NOC,0   RTS
OPC     SBC,D,0,81,00,0,0     SBCD
OPC     S,F,29,51,C0,0,0      SF
OPC     SH,I,29,52,C0,0,0     SHI
OPC     SL,S,29,53,C0,0,0     SLS
OPC     SC,C,29,54,C0,0,0     SCC
```

```
         OPC     SC,S,29,55,C0,0,0     SCS
         OPC     SN,E,29,56,C0,0,0     SNE
         OPC     SE,Q,29,57,C0,0,0     SEQ
         OPC     SV,C,29,58,C0,0,0     SVC
         OPC     SV,S,29,59,C0,0,0     SVS
         OPC     SP,L,29,5A,C0,0,0     SPL
         OPC     SM,I,29,5B,C0,0,0     SMI
         OPC     SG,E,29,5C,C0,0,0     SGE
         OPC     SL,T,29,5D,C0,0,0     SLT
         OPC     SG,T,29,5E,C0,0,0     SGT
         OPC     SL,E,29,5F,C0,0,0     SLE
         OPC     STO,P,23,4E,72,0,0    STOP
         OPC     S,T,29,50,C0,0,0      ST
         OPC     SUB,A,2,90,C0,0,0     SUBA
         OPC     SUB,I,3,04,00,0,0     SUBI
         OPC     SUB,Q,4,51,00,0,0     SUBQ
         OPC     SUB,X,5,91,00,0,0     SUBX
         OPC     SU,B,1,90,00,0,0      SUB
         OPC     SWA,P,24,48,40,0,0    SWAP
         OPC     TA,S,29,4A,C0,0,0     TAS
         OPC     TRAP,V,22,4E,76,NOC,0 TRAPV
         OPC     TRA,P,25,4E,40,0,0    TRAP
         OPC     TS,T,13,4A,00,0,0     TST
         OPC     UNL,K,26,4E,58,0,0    UNLK

         DC.B    0             PAD BYTE

TBLOPCE  DS      0

* WITHOUT LABEL FIELD
* 012345678901234567890123456789012345678901234567890
* AAAAAA DDDDDDDDDDDDDDDDDDDD OPCODE  OPERAND
*        FDATA                FOC     FOP

* 012345678901234567890123456789012345678901234567890
* AAAAAA DDDDDDDDDDDDDDDDDDDD LLLLLLLL OPCODE  OPERAND
*        FDATA                FOL      FOC     FOP

* A3 = STORE POINTER
* A4 = PROGRAM COUNTER
* A5 = SOURCE PTR BEGINNING
* A6 = SOURCE PTR END+1
*
OUTBSIZE EQU     80        BUFFER SIZE
FDATA    EQU     10        OFFSET TO DATA
FOL      EQU     31        OFFSET TO LABEL
FOC      EQU     31        OFFSET TO OP-CODE (NO LABEL FIELD)
FOP      EQU     39        OFFSET TO OPERAND (NO LABEL FIELD)

CODE68K  LINK    A1,#0-(ESKE-ESKB)
         MOVE.L  A1,LINK(A7)    SAVE LINKAGE
         MOVE.L  A7,A1          A1 = BASE REGISTER TO DATA

         MOVE.B  #BLANK,(A6)    INSURE LAST CHAR IS SPACE

         MOVE.L  A3,A0
         MOVEQ   #OUTBSIZE-1,D0
M300     MOVE.B  #BLANK,(A0)+   SPACE FILL BUFFER
         DBRA    D0,M300
         SUBQ.L  #2,A0
         MOVE.L  A0,PTRBUFE(A1) PTR TO END OF BUFFER

         MOVE.L  A4,PCOUNTER(A1)  FORMAT PC
         MOVE.L  A4,D4          D4 = P-COUNTER
         MOVE.L  A4,D0

         MOVE.L  A6,-(A7)       SAVE A6
         MOVE.L  A3,A6
         BSR     FRELADDR       FORM RELATIVE ADDRESS
         MOVE.L  (A7)+,A6       RESTORE A6

         MOVEQ   #1,D7          POSSIBLE ERROR CODE
         MOVE.L  A4,D0
         ROR.L   #1,D0
         BMI     ERDONE         PC ODD ADDRESS

         MOVEQ   #FOL,D7        POSSIBLE ERROR CODE
         CMPI.B  #BLANK,(A5)+
         BNE     ERDONE         1ST CHAR NOT SPACE
```

```
         MOVE.B  #2,TNB(A1)      INZ # OF BYTES
         MOVE.W  #$40,TLENGTH(A1)  SIZE = .W (DEFAULT)
         CLR.B   TLSPEC(A1)      DEFAULT (SIZE NOT SPECIFIED)

         MOVE.L  A3,A0           A0 = STORE ADDRESS
         ADD.L   #FOC,A0

M340     BSR     GETCHARF        GET PAST SPACES
         CMPI.B  #BLANK,D0
         BEQ     M340

         SUBQ.L  #1,A5           FORMAT OP-CODE
M350     BSR     GETCHARF
         MOVE.B  D0,(A0)+
         CMPI.B  #'.',D0
         BNE.S   M352

         MOVE.B  (A5),TLSPEC(A1)  NOT DEFAULT
         CMPI.B  #'W',(A5)
         BEQ.S   M352
         CMPI.B  #'S',(A5)
         BEQ.S   M352            .SHORT = .WORD
         CLR.W   TLENGTH(A1)
         CMPI.B  #'B',(A5)
         BEQ.S   M352            SIZE = .W
         MOVE.W  #$80,TLENGTH(A1)
         CMPI.B  #'L',(A5)
         BNE     ERF

M352     CMPI.B  #BLANK,D0
         BNE     M350            NOT SPACE CONTINUE

* SEARCH OP-CODE TABLE
         LEA     TBLOPC(PC),A0   A0 = PTR TO TABLE OF CODES
M410     MOVE.L  A3,A2           A3 = START OF STORE BUFFER
         ADD.L   #FOC,A2         A2 = PTR TO OP-CODE JUST FORMATTED

M415     MOVE.B  (A0)+,D0        XXXXXXDD
         EXT.W   D0              XXXXSSDD  SIGN EXTENDED
         ANDI.B  #$7F,D0
         CMP.B   (A2)+,D0
         BNE.S   M420            NON-MATCH
         TST.W   D0
         BPL     M415            SIGN RESET; CONTINUE
         BRA.S   M430            MATCH COMPLETE

M420     TST.W   D0              SEQUENCE TO NEXT CODE
         BMI.S   M426
M424     MOVE.B  (A0)+,D0
         BPL     M424            FINISH THIS OP-CODE
M426     ADDQ.L  #3,A0           ADJUST PTR TO TABLE
         LEA     TBLOPCE(PC),A2
         CMP.L   A0,A2
         BNE     M410

M428     MOVEQ   #FOC,D7         ERROR CODE
         BRA     ERDONE

* GET GOTO INDEX
* GET FIRST WORD MASK
M430     MOVE.B  (A2),D0         MUST TERMINATE OP-CODE
         CMPI.B  #BLANK,D0         WITH SPACE OR PERIOD
         BEQ.S   M432
         CMPI.B  #'.',D0
         BNE     M428            ERROR
M432

         CLR.L   D0
         MOVE.B  (A0)+,D0        D0 =  KEYS  INDEX
         MOVE.B  D0,D1           D1 =  KEYS (INDEX)
         ANDI.B  #$3F,D0         D0 =        INDEX
         ASL.L   #1,D0           INDEX * TWO
         MOVE.B  (A0)+,D2
         LSL.W   #8,D2
         MOVE.B  (A0)+,D2        D2 = FIRST WORD MASK
         MOVE.W  D2,(A1)         *D2,TDATA(A1)

* INSURE .S .W MATCH OP-CODE
```

```
         MOVE.B  TLSPEC(A1),D3  D3 = .n SPECIFIED
         BEQ.S   M4326          NOT SPECIFIED
         BTST    #6,D1
         BEQ.S   M4324          .W ALLOWED (.S NOT ALLOWED)
         CMPI.B  #'W',D3
         BEQ     M428           .W NOT ALLOWED
         CMPI.B  #'B',D3
         BEQ     M428           .B NOT ALLOWED
         BRA.S   M4326

M4324    CMPI.B  #'S',D3
         BEQ     M428           .S NOT ALLOWED
M4326

* CALCULATE GOTO ADDRESS

         LEA     TBLKEYS(PC),A0 A0 = PTR TO KEYS
         MOVE.W  (A0,D0),D0     D0 = 16 BIT OFFSET
         LEA     XBASE(PC),A2   A2 = BASE ADDRESS
         ADD.L   D0,A2          A2 = COMPUTED GO TO  ADDRESS

* FORMAT OPERAND IF REGUIRED

         TST.B   D1             LOOK AT KEY
         BMI.S   M440           OPERAND NOT REQUIRED

         MOVE.L  A3,A0
         ADD.L   #FOP,A0        STORE POINTER
         MOVE.L  A0,PTROP(A1)   POINTER TO OPERAND (FORMATED)
M435     BSR.S   GETCHARF       D0 = CHAR
         CMPI.B  #BLANK,D0
         BEQ     M435           SKIP SPACES

M437     MOVE.B  D0,(A0)+       MOVE REST OF SOURCE LINE
         BSR.S   GETCHARF       D0 = CHAR
         CMP.L   A5,A6
         BPL     M437
         MOVE.L  A0,PTRBUFE(A1) POINTER TO END FORMATED SOURCE
         MOVE.L  A0,A6          A6 = POINTER TO END OF SOURCE
M440

         MOVE.L  PTROP(A1),A5   A5 = PTR TO OPERAND
         LEA     TDATA+2(A1),A4 A4 = BASE ADDR FOR DATA STORE
         CLR.L   D3             D3 = OFFSET FOR DATA STORE
         JMP     (A2)           GOTO ROUTINE
*                                D2 = MASK
*                                D4 = P-COUNTER

COMMA    CMPI.B  #',',(A5)+
         BNE.S   ER             NOT COMMA
         RTS

GETCHAR  CMP.L   A5,A6
         BMI.S   ER
         MOVE.B  (A5)+,D0
         RTS

GETCHARF CMP.L   A5,A6
         BMI.S   ERF
         MOVE.B  (A5)+,D0
         RTS

ERF      MOVE.L  A0,A5

ER       MOVE.L  A5,D7          D7 = ERROR FLAG
         SUB.L   A3,D7          ..& POSITION OF ERROR
ERDONE   CLR.L   D6             D6 = (ZERO) BYTE COUNT
         BRA.S   CMMD35

CMMD2    CMPI.B  #BLANK,(A5)
         BNE     ER             OPERAND DID NOT END WITH SPACE

MCMMD2   DS      0              NO OPERAND SEQUENCE
         MOVE.W  D2,(A1)        *D2,TDATA(A1)

         MOVE.B  TNB(A1),D3     FORMAT DATA
         MOVE.L  D3,D6          D7 = NUMBER OF BYTES
         LEA     (A1),A2        A2 = PTR TO HEX DATA  *TDATA(A1),A2
```

```
         MOVE.L  A3,A6          D3 = NUMBER OF BYTES
         ADD.L   #FDATA,A6      A6 = STORE PTR
FPR315   MOVE.B  (A2)+,D0
         BSR     PNT2HX
         SUBQ.L  #1,D3
         BNE     FPR315         MORE BYTES

         CLR.L   D7             RESET ERROR FLAG

CMMD35   MOVEM.L (A1),D0-D2       D0-D2 = DATA   *TDATA(A1),D0-D2

         MOVE.L  PTRBUFE(A1),A6   A6 = POINTER TO END OF BUFFER

         MOVE.L  PCOUNTER(A1),A4  A4 = ORGINAL PC

         MOVE.L  LINK(A1),A1
         UNLK    A1
         RTS                    RETURN TO REQUESTOR
*                               A3 = POINTER TO START OF BUFFER
*                               D6 = NUMBER OF BYTES ASSEMBLED
*                               D7 = ERROR FLAG (POSITION)


*  SIZE = BYTE
*    DY,DX
*    -(AY),-(AX)
*    ....RX@.SS...RY@
MABCD    DS      0              (INDEX 0) ABCD SBCD
         TST.B   TLSPEC(A1)
         BEQ.S   MABCD9         DEFAULT SIZE = BYTE
         CMPI.W  #$00,TLENGTH(A1)
         BNE     ER             NOT .B
MABCD9

         MOVE.W  #$11,D7
         BSR     EA

         BSR     COMMA

         MOVE.L  D6,D0
         ANDI.W  #7,D0
         OR.W    D0,D2

         BTST    #5,D6
         BEQ.S   MABCD55        D@,D@ MODE

         ORI.W   #$0008,D2      -(A@),-(A@) MODE

         MOVE.W  #$10,D7
         BSR     EA

         ANDI.W  #7,D6
         ROR.W   #7,D6
         OR.W    D6,D2
         BRA     CMMD2

MABCD55  BSR     GETREGD        D@,D@
         ROR.W   #7,D0
         OR.W    D0,D2
CMMD2S10 BRA     CMMD2


MCMP     DS      0              (INDEX 34)
         BSR     EAALL
         MOVE.L  D6,D4          D4 = SOURCE MODE

         BSR.S   COMMAS20

         CMPI.B  #'A',(A5)
         BEQ     MADDA21        DESTINATION = A@

         CMPI.B  #$3C,D4
         BEQ.S   MCMP56         SOURCE  ....I  #<DATA>,...

         BSR     FSIZE

         MOVE.W  #$009,D7
         BSR     EA             D@ + (A@)+
         MOVE.L  D6,D0          MMMRRR
```

```
         ANDI.W  #$38,D0        MMM...

         BEQ.S   MCMP39         DESTINATION  D@

         CMPI.B  #$18,D0
         BNE     ER

         ANDI.W  #$F,D6         (AY)+,(AX)+
         ROR.W   #7,D6
         OR.W    D6,D2          ....AX@.........
         ORI.W   #$0100,D2      ...1AX@.........

         MOVE.L  D4,D0
         ANDI.W  #$38,D0
         CMPI.W  #$18,D0
         BNE     ER             NOT (A@)+
         ANDI.W  #$F,D4         ............1AY@
         OR.W    D4,D2
         BRA     CMMD2

*  <EA>,D@
MCMP39   ROR.W   #7,D6
         OR.W    D6,D2

         OR.W    D4,D2
         BRA.S   CMMD2S11

MCMP56   MOVE.W  #$0C00,D2      #<DATA>,<EA>      MASK = CMPI

         BSR     FSIZE

         BSR     EADA
         OR.W    D6,D2
CMMD2S11 BRA     CMMD2S10

COMMAS20 BRA     COMMA

MADD     DS      0              (INDEX 1)
         BSR     EAALL
         MOVE.L  D6,D4          D4 = SOURCE MODE

         BSR     COMMAS20

         CMPI.B  #'A',(A5)
         BEQ     MADDA21        DESTINATION = A@

         CMPI.B  #$3C,D4
         BEQ.S   MADD56         SOURCE  ....I  #<DATA>,...

         BSR     FSIZE

         MOVE.W  #$1FF,D7
         BSR     EA             ALTERABLE + D@
         MOVE.L  D6,D0          MMMRRR
         ANDI.W  #$38,D0        MMM...
         BNE.S   MADD46         DESTINATION NOT  D@

*  <EA>,D@
         ROR.W   #7,D6
         OR.W    D6,D2

         OR.W    D4,D2
         BRA     CMMD2S11

MADD46   DS      0              D@,<EA>
         ORI.W   #$100,D2

         ROR.W   #7,D4
         OR.W    D4,D2          <EA> AS DESTINATION

         OR.W    D6,D2          D@  AS SOURCE
         BRA     CMMD2S11

MADD56   MOVE.L  D2,D0          #<DATA>,<EA>
         MOVE.W  #$0600,D2      MASK = ADDI

         CMPI.W  #$D000,D0
         BEQ.S   MADD58
         MOVE.W  #$400,D2       MASK = SUBI
```

```
MADD58

         BSR     FSIZE

         BSR     EADA            DATA ALTERABLE ONLY
         OR.W    D6,D2
         BRA     CMMD2S11

MADDI    MOVE.L  PTROP(A1),A5    (INDEX 3) CMPI
         BSR     FSIZE

         BSR     EAZ

         BSR     COMMAS20

         BSR     EADA            DATA ALTERABLE ONLY
         OR.W    D6,D2
         BRA     CMMD2S11

*  SIZE =  BYTE, WORD, LONG
*  #<DATA>,SR
*  #<DATA>,<EA>    DATA ALTERABLE ONLY
MANDI    DS      0               (INDEX 28) ANDI EORI ORI
         BSR     FSIZE

         BSR     EAZ

         BSR     COMMAS20

         CMPI.B  #'S',(A5)
         BEQ.S   MANDI23

         BSR     EADA
         OR.W    D6,D2
         BRA     CMMD2S11

MANDI23  CMPI.B  #'R',1(A5)      #<DATA>,SR
         BNE     ER
         CMPI.W  #$0080,TLENGTH(A1)
         BEQ     ER              .L NOT ALLOWED WITH SR
         ORI.W   #$003C,D2
         ADDQ.L  #2,A5           POINTER TO END OF OPERAND
         BRA     CMMD2

MADDA    DS      0               (INDEX 2)
         BSR     EAALL
         OR.W    D6,D2

         BSR     COMMA

MADDA21  OR.W    D6,D2
         MOVE.W  TLENGTH(A1),D0
         BEQ     ER              .BYTE NOT ALLOWED
         LSL.W   #1,D0           .W = 011......
         ORI.W   #$00C0,D0       .L = 111......
         OR.W    D0,D2

         BSR     GETREGA
         ROR.W   #7,D0
         OR.W    D0,D2
         BRA     CMMD2

MADDQ    DS      0               (INDEX 4)
         BSR     FSIZE

         BSR     GETIMM

         TST.L   D0
         BEQ     ER              DATA = ZERO
         CMPI.B  #9,D0
         BPL     ER              VALUE TOO BIG
         ANDI.W  #$7,D0          MAKE 8 = 0
         ROR.W   #7,D0           SHIFT DATA TO BITS 9-11
         OR.W    D0,D2

         BSR     COMMA

         BSR     EAA             ALTERABLE ADDRESSING
```

```
         OR.W    D6,D2
         MOVE.L  D2,D0
         ANDI.W  #$C0,D0
         BNE.S   MADDQ25

* BYTE SIZE; ADDRESS REGISTER DIRECT NOT ALLOWED
         MOVE.L  D2,D0
         ANDI.W  #$38,D0
         CMPI.B  #$08,D0
         BEQ     ER
MADDQ25  BRA     CMMD2

* SIZE = BYTE, WORD, LONG
*    DY,DX
*    -(AY),-(AX)
*    ....RX@.SS...RY@
MADDX    DS      0              (INDEX 5)
         BSR     FSIZE

         MOVE.W  #$11,D7
         BSR     EA

         BSR     COMMA

         MOVE.L  D6,D0
         ANDI.W  #7,D0
         OR.W    D0,D2

         BTST    #5,D6
         BEQ.S   MADDX5         D@,D@ MODE

         ORI.W   #$0008,D2      -(A@),-(A@) MODE

         MOVE.W  #$10,D7
         BSR     EA

         ANDI.W  #7,D6
         ROR.W   #7,D6
         OR.W    D6,D2
         BRA     CMMD2

MADDX5   BSR     GETREGD        D@,D@
         ROR.W   #7,D0
         OR.W    D0,D2
         BRA     CMMD2

*  SIZE = BYTE, WORD, LONG
*         <EA>,D@
*         D@,<EA>
MAND     BSR     FSIZE          (INDEX 6)

         CMPI.B  #'#',(A5)
         BEQ.S   MAND90

         BSR     A5TODEST       MOVE A5 TO DESTINATION

         MOVE.B  (A5),D0        D0 = 1ST CHAR OF DESTINATION
         MOVE.L  PTROP(A1),A5   A5 = POINTER TO OPERAND
         CMPI.B  #'D',D0
         BEQ.S   MAND77

         ORI.W   #$0100,D2       D@,<EA>

         BSR     GETREGD
         ROR.W   #7,D0
         OR.W    D0,D2

         BSR     COMMA

         BSR     EAM            ALTERABLE MEMORY
         OR.W    D6,D2
         BRA     CMMD2

MAND77   BSR     EADADDR        <EA>,D@
         OR.W    D6,D2

         BSR     COMMA

         BSR     GETREGD
```

```
         ROR.W   #7,D0
         OR.W    D0,D2
         BRA     CMMD2

MAND90   ANDI.W  #$F000,D2
         CMPI.W  #$C000,D2
         BEQ.S   MAND97          AND
         MOVE.W  #$0000,D2      CHANGE TO "ORI"
MAND91   BRA     MANDI
MAND97   MOVE.W  #$0200,D2      CHANGE TO "ADDI"
         BRA     MAND91

MEOR     BSR     FSIZE          (INDEX 35)

         CMPI.B  #'#',(A5)
         BEQ.S   MEOR90

         BSR     GETREGD
         ROR.W   #7,D0
         OR.W    D0,D2

         BSR     COMMA

         BSR     EADA           DATA ALTERABLE ADDRESSING
         OR.W    D6,D2
         BRA     CMMD2

MEOR90   MOVE.L  PTROP(A1),A5   A5 = POINTER TO OPERAND
         MOVE.W  #$0A00,D2      CHANGE TO "EORI"
         BRA     MANDI

A5TODEST CLR.L   D1             MOVE A5 TO DESTINATION
A5300    BSR     GETCHAR
         CMPI.B  #'(',D0
         BNE.S   A5305
         MOVE.L  D0,D1
A5305    CMPI.B  #')',D0
         BEQ     A5TODEST
         CMPI.B  #',',D0
         BNE     A5300
         TST     D1
         BNE     A5300
         RTS

MASL     DS      0              (INDEX 7)

*         ASL     LSL     ROL     ROXL
*  MASKS  E000    E008    E018    E010
*         E0C0    E2C0    E6C0    E4C0   SHIFT MEMORY

         BSR     FSIZE

         MOVE.B  (A5)+,D0
         CMPI.B  #'#',D0
         BNE.S   MSL200

*  #<COUNT>,D@
         BSR     EV
         CMPI.L  #8,D0
         BGT     ER             GREATER THAN 8

         ANDI.B  #$7,D0         MAKE 8 INTO 0
MSL150   ROR.W   #7,D0
         ANDI.W  #$F1FF,D2
         OR.W    D0,D2          COUNT/REG

         BSR     COMMA

         BSR     GETREGD
         OR.W    D0,D2
         BRA     CMMD2

MSL200   DS      0              D@,D@
         CMPI.B  #'D',D0
         BNE.S   MSL300

*        D@,D@
         ORI.W   #$20,D2
         SUBQ.L  #1,A5
```

```
         BSR     GETREGD
         BRA     MSL150

MSL300   DS      0               <EA>      SHIFT MEMORY
         SUBQ.L  #1,A5
         ORI.W   #$00C0,D2       SIZE = MEMORY

         ANDI.W  #$FFC0,D2       REMOVE "SHIFT MEMORY" BITS

         CMPI.W  #$0040,TLENGTH(A1)
         BNE.S   ER2             NOT .WORD

         BSR     EAM
         OR.W    D6,D2
         BRA     CMMD2

MSCC     BSR     SIZEBYTE        (INDEX 29) NBCD SCC TAS

         BSR     EADA            DATA ALTERABLE ONLY
         OR.W    D6,D2
         BRA     CMMD2

MBCHG    DS      0               (9)
         CMPI.B  #'#',(A5)
         BNE.S   MB200

         MOVE.W  #$0840,D2       NEW OP-CODE MASK

MB100    ADDQ.L  #1,A5
         BSR     EV              EVALUATE EXPRESSION
         CMPI.L  #33,D0
         BGT.S   ER2             (MODULO 32)
         MOVE.W  D0,TDATA+2(A1)
         ADDQ.L  #2,D3           STORE POINTER

         ADDQ.B  #2,TNB(A1)

MB105    BSR     COMMA

         BSR     EADA            DESTINATION
         OR.W    D6,D2

         TST.B   TLSPEC(A1)      ..
         BEQ.S   MB185           DEFAULT

         ANDI.W  #$0038,D6
         BNE.S   MB145           DESTINATION <EA> WAS NOT D@

         CMPI.W  #$80,TLENGTH(A1)  DESTINATION <EA> WAS D@
         BEQ.S   MB185           LENGTH IS .L
ER2      BRA     ER

MB145    TST.W   TLENGTH(A1)
         BNE     ER2             NOT BYTE LENGTH

MB185    BRA     CMMD2

MB200    BSR     GETREGD         DYNAMIC
         ROR.W   #7,D0
         OR.W    D0,D2

         BRA     MB105

MBSET    CMPI.B  #'#',(A5)       (INDEX 11) BCLR BSET
         BNE     MB200

         MOVE.W  #$08C0,D2
         BRA     MB100

MBCLR    CMPI.B  #'#',(A5)       (INDEX 30)
         BNE     MB200

         MOVE.W  #$0880,D2
         BRA     MB100

MBTST    CMPI.B  #'#',(A5)       (INDEX 31)
         BNE     MB200

         MOVE.W  #$0800,D2
```

```
         BRA     MB100

MDBCC    DS      0               (INDEX 8)
         BSR     GETREGD
         OR.W    D0,D2

         BSR     COMMA
         BSR.S   EVSR
         BRA.S   MBRA23

*  SIZE   .S  =  .W   (DEFAULT)
*         .L  =  .L
MBRA     DS      0               (INDEX 10)
         BSR.S   EVSR

         CMPI.W  #$0080,TLENGTH(A1)
         BEQ.S   MBRA23          FORCED LONG

         BSR     EA8BITS         -128 TO +127
         BNE.S   MBRA23          NOT 8 BIT VALUE

         OR.B    D5,D2
         BRA     CMMD2           .SHORT

EVER     BRA     ER              ERROR HANDLER

MBRA23   TST.B   TLSPEC(A1)
         BEQ.S   MBRA27
         CMPI.W  #$0040,TLENGTH(A1)
         BEQ     EVER            .S SPECIFIED
MBRA27

         MOVE.L  D4,D5           RESTORE D5
         BSR     EA16BITS        -32K TO +32K
         MOVE.W  D5,TDATA+2(A1)
         ADDQ.B  #2,TNB(A1)
         BRA     CMMD2

EVSR     BSR     EV
         CMPI.B  #BLANK,(A5)
         BNE     EVER            DID NOT TERMINATE WITH SPACE

         MOVE.L  D0,D5
         ASR.L   #1,D0
         BCS     EVER            ODD VALUE
         MOVE.L  PCOUNTER(A1),D4
         ADDQ.L  #2,D4           D4 = PC + 2
         SUB.L   D4,D5
         BEQ     EVER            ZERO; SPECIAL CASE ERROR
         MOVE.L  D5,D4
         RTS

MCHK     DS      0               (INDEX 12) CHK DIV MUL
         BSR     SIZEWORD

         BSR     EADADDR         DATA ADDRESSING ONLY
         OR.W    D6,D2

         BSR     COMMA

         BSR     GETREGD
         ROR.W   #7,D0
         OR.W    D0,D2

         BRA     CMMD2

MCLR     DS      0               (INDEX 13)
         BSR     FSIZE

         BSR     EADA            DATA ALTERABLE ONLY
         OR.W    D6,D2
         BRA     CMMD2

* SIZE = BYTE, WORD, LONG
MCMPM    DS      0               (INDEX 14)
         BSR     FSIZE

         MOVE.W  #$0008,D7
         BSR     EA              -(A@)   ONLY
```

```
         ANDI.W  #7,D6
         OR.W    D6,D2

         BSR     COMMA

         MOVE.W  #$0008,D7
         BSR     EA
         ANDI.W  #7,D6
         ROR.W   #7,D6
         OR.W    D6,D2
         BRA     CMMD2

MEXG     DS      0               (INDEX 16)
         BSR     SIZELONG

         BSR     GETREGAD
         MOVE.L  D0,D4           D4 = REG NUMBER
         MOVE.L  D1,D5           D5 = REG TYPE

         BSR     COMMA           TEST FOR COMMA

         BSR     GETREGAD

         CMP.L   D1,D5
         BEQ.S   MEXG35          BOTH REGS THE SAME

*  DX,AY  OR  AY,DX
         ORI.W   #$88,D2         MODE
         TST.B   D1
         BNE.S   MEXG25

         EXG     D0,D4           SWAP SOURCE & DESTINATION

MEXG25   OR.W    D0,D2           .......MMMMMYYY
         ROR.W   #7,D4
         OR.W    D4,D2           ....XXXMMMMMYYY
         BRA     CMMD2

MEXG35   ORI.W   #$0040,D2       OP-MODE
         TST.B   D1
         BEQ     MEXG25          DX,DY

         ORI.W   #$0048,D2       AX,AY
         BRA     MEXG25


MEXT     DS      0               (INDEX 17)
         TST.W   TLENGTH(A1)
         BEQ     ER              BYTE SIZE NOT ALLOWED

         BSR     FSIZE           .W = ........10......
         ADDI.W  #$0040,D2       .L = ........11......

         BSR     GETREGD
         OR.W    D0,D2
         BRA     CMMD2

MMOVEM   DS      0               (INDEX 27)
         MOVE.W  TLENGTH(A1),D0 SIZE BITS  76 TO 6
         BEQ     ER              BYTE       00  ERROR
         LSR.W   #1,D0           WORD       01    0
         ANDI.W  #$0040,D0       LONG       10    1
         OR.W    D0,D2

         ADDQ.B  #2,TNB(A1)      NUMBER OF BYTES
         ADDQ.L  #2,D3           FORCE STORE PTR PAST MASK

         CMPI.B  #'A',(A5)
         BEQ.S   MMM44
         CMPI.B  #'D',(A5)
         BEQ.S   MMM44


*    <EA>,<REGISTER LIST>       MEMORY TO REGISTER
         ORI.W   #$0400,D2       DIRECTION BIT

         MOVE.W  #$7EC,D7        MODES ALLOWED
         BSR     EA
         OR.W    D6,D2
```

```
         BSR     COMMA

         BSR.S   MMM48
         BRA     CMMD2


*   <REGISTER LIST>,<EA>        REGISTER TO MEMORY
MMM44    DS      0

* EVALUATE DESTINATION FIRST
MMM46    BSR     GETCHAR
         CMPI.B  #',',D0         FIND COMMA
         BNE     MMM46

         MOVE.W  #$1F4,D7        MODES ALLOWED
         BSR     EA
         OR.W    D6,D2
         MOVE.L  A5,PENDOP(A1)   END OF OPERAND
         MOVE.L  PTROP(A1),A5
         BSR.S   MMM48           EVALUATE REGISTERS
         MOVE.L  PENDOP(A1),A5   POINTER TO END OF OPERAND
         BRA     CMMD2

*        D6 = CORRESPONDENCE MASK
*        D4 = CONTROL  $FF '-BLANK/'
MMM48    CLR.L   D6              MASK
         MOVEQ   #-1,D4          CONTROL = $FF

RL111    BSR     GETCHAR
         CMPI.B  #',',D0
         BEQ.S   RL114           DONE; FOUND COMMA
         CMPI.B  #BLANK,D0
         BNE.S   RL115           NOT SPACE
RL114    RTS     DONE

RL115    CMPI.B  #'/',D0
         BNE.S   RL444

         TST.B   D4              CONTROL
         BMI     ER
         MOVE.L  D0,D4           CONTROL = '/'
RL333    BSR     GETREGAD
         OR.B    D0,D1           D1 = BIT POSITION
         MOVE.B  D1,D5           D5 = LAST REGISTER ENTERED
         BSR.S   SETBIT
         BRA     RL111

RL444    CMPI.B  #'-',D0
         BNE.S   RL666

         CMPI.B  #'/',D4         CONTROL
         BNE     ER
         MOVE.L  D0,D4           CONTROL = '-'
         BSR     GETREGAD
         OR.B    D0,D1
         MOVE.L  D1,D7           D7 = NOW REGISTER
         MOVE.B  D5,D0           D5 = LAST REG
         EOR.B   D1,D0
         ANDI.B  #$38,D0
         BNE     ER              NOT MATCED SET
         CMP.B   D1,D5
         BPL     ER

RL555    ADDQ.L  #1,D5
         MOVE.L  D5,D1
         BSR.S   SETBIT
         CMP.B   D7,D5
         BMI     RL555
         BRA     RL111

RL666    TST.B   D4
         BPL     ER
         MOVE.B  #'/',D4         CONTROL = '/'
         SUBQ.L  #1,A5
         BRA     RL333

SETBIT   LEA     MTBL(PC),A0     SET BIT IN CORRESPONDENCE MASK
         MOVE.L  D2,D0
```

```
         ANDI.W  #$38,D0
         CMPI.W  #$20,D0
         BNE.S   RL30            NOT PREDECREMENT
         MOVE.B  (A0,D1),D1      D1 = BIT  (FOR SURE)
RL30     BSET    D1,D6

         MOVE.W  D6,TDATA+2(A1) SAVE CORRESPONDENCE MASK
         RTS

MTBL     DC.B    15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0



*   D@,<DATA>(A@)
*   <DATA>(A@),D@
*         (A@),D@            FORCED TO 0(A@),D0
*         D@,(A@)            FORCED TO D0,0(A@)
*
*  SIZE = WORD, LONG
MMOVEP   DS      0              (INDEX 33)
         MOVE.W  TLENGTH(A1),D0
         BEQ     ER             .BYTE NOT ALLOWED
         LSR.W   #1,D0
         ANDI.W  #$0040,D0
         OR.W    D0,D2          SIZE

         MOVE.W  #$25,D7
         BSR     EA             D6 = MODE

         BSR     COMMA

         MOVE.L  D6,D0
         ANDI.W  #$38,D0
         CMPI.B  #$0,D0
         BEQ.S   MMP344         D@,<DATA>(A@)

*    <DATA>(A@),D@
         BSR     GETREGD
         ROR.W   #7,D0
         OR.W    D0,D2          D@
         BRA.S   MMP348

MMP344   ORI.W   #$0080,D2      REGISTER TO MEMORY

         ROR.W   #7,D6
         OR.W    D6,D2          D@

         MOVE.W  #$24,D7
         BSR     EA
MMP348   MOVE.L  D6,D0
         ANDI.W  #7,D6
         OR.W    D6,D2          A@

         ANDI.W  #$38,D0
         CMPI.B  #$10,D0
         BNE.S   MMP368         <DATA>(A@)

         CLR.W   TDATA+2(A1)    <DATA> FORCED TO ZERO;  "(A@)"
         ADDQ.B  #2,TNB(A1)     NUMBER OF BYTES
         ADDQ.L  #2,D3          STORE POINTER
MMP368   BRA     CMMD2

MMOVEQ   DS      0              (INDEX 34)
         BSR     GETIMM
         MOVE.L  D0,D5

         BSR     EA8BITS        -128 TO +127
         BNE     ER
         OR.B    D5,D2          D5 = VALUE

         BSR     COMMA

         BSR     GETREGD        D@
         ROR.W   #7,D0

MMQ20    OR.W    D0,D2
         BSR.S   SIZELONG
         BRA     CMMD2
```

```
SIZELONG TST.B   TLSPEC(A1)     MUST BE .LONG
         BEQ.S   SI201          DEFAULT SIZE OK
         CMPI.W  #$0080,TLENGTH(A1)
         BNE.S   ER10           NOT .LONG
SI201    RTS

SIZEWORD CMPI.W  #$0040,TLENGTH(A1)  MUST BE .WORD
         BEQ     SI201          [RTS]
ER10     BRA     ER

SIZEBYTE TST.B   TLSPEC(A1)
         BEQ.S   SI222          DEFAULT SIZE OK
         TST.W   TLENGTH(A1)
         BNE     ER10
SI222    RTS

MMOVE    DS      0              (INDEX 21)
         CMPI.B  #'S',(A5)
         BNE.S   MMM40
         MOVE.W  #$40C0,D2      SR,<EA>
         ADDQ.L  #1,A5
         CMPI.B  #'R',(A5)+
         BNE     ER10

         BSR     COMMA

         BSR     EADA           DATA ALTERABLE ONLY (DESTINATION)

MM315    OR.W    D6,D2
         BSR     SIZEWORD
         BRA     CMMD2

MMM40    CMPI.B  #'U',(A5)
         BNE.S   MMM50
         ADDQ.L  #1,A5
         CMPI.B  #'S',(A5)+
         BNE     ER10
         CMPI.B  #'P',(A5)+
         BNE     ER10

         BSR     COMMA

         MOVE.W  #$4E68,D2      USP,A@
         BSR     GETREGA
         BRA     MMQ20

* GET EXCEPTIONS FROM DESTINATION
MMM50    DS      0

         BSR     A5TODEST       MOVE A5 TO DESTINATION

         MOVE.B  (A5)+,D0
         CMPI.B  #'C',D0
         BNE.S   MMM60
         CMPI.B  #'C',(A5)+
         BNE     ER10
         CMPI.B  #'R',(A5)+
         BNE     ER10
         MOVE.W  #$44C0,D2      <EA>,CCR

MM508    MOVE.L  A5,PENDOP(A1)  SAVE POINTER
         MOVE.L  PTROP(A1),A5   A5 = POINTER TO OPERAND

         BSR     EADADDR        DATA ADDRESSING ONLY (SOURCE)
         MOVE.L  PENDOP(A1),A5
         BRA     MM315

MMM60    CMPI.B  #'S',D0
         BNE.S   MM70
         MOVE.W  #$46C0,D2      <EA>,SR
         CMPI.B  #'R',(A5)+
         BNE     ER
         BRA     MM508

MM70     CMPI.B  #'U',D0
         BNE.S   MM80
         MOVE.W  #$4E60,D2      A@,USP
         CMPI.B  #'S',(A5)+
```

```
         BNE     ER
         CMPI.B  #'P',(A5)+
         BNE     ER

         MOVE.L  A5,PENDOP(A1)
         MOVE.L  PTROP(A1),A5
         BSR     GETREGA
         MOVE.L  PENDOP(A1),A5  RESTORE A5
         BRA     MMQ20

MM80     MOVE.L  PTROP(A1),A5   A5 = POINTER TO SOURCE FIELD
         BSR     FSIZE          GET SIZE (BITS  7 - 6)
         LSL.W   #6,D2          ADJUST TO(BITS 13-12)
         BTST    #13,D2
         BNE.S   MM804          .L 10 TO 10
         ADDI.W  #$1000,D2      .W 01 TO 11
         ORI.W   #$1000,D2      .B 00 TO 01
MM804    BSR     EAALL          SOURCE; ALL MODES ALLOWED
         OR.W    D6,D2

* IF BYTE SIZE; "ADDRESS REGISTER DIRECT" NOT ALLOWED

         MOVE.L  D2,D0
         ANDI.W  #$3000,D0
         CMPI.W  #$1000,D0
         BNE.S   MM806          NOT .B SIZE
         ANDI.B  #$38,D6
         CMPI.B  #$08,D6
         BEQ     ER
MM806    DS      0

         BSR     COMMA

         MOVE.W  #$1FF,D7       DATA ALTERABLE + A@
         BSR     EA

         MOVE.L  D6,D0          DESTINATION
         ANDI.W  #$0038,D0
         CMPI.W  #$0008,D0
         BEQ.S   MMOVEA1        A@ MAKE MOVEA

* POSITION REGISTER AND MODE OF DESTINATION

MM825    ROR.L   #3,D6          RRR..... .....MMM
         ROR.W   #3,D6                  MMM.....
         SWAP    D6             MMM..... RRR.....
         ROL.W   #3,D6                  .....RRR
         ROL.L   #1,D6          MM..... .....RRRM
         ROL.L   #8,D6          ........... ....RRRMMM......
         OR.W    D6,D2
         BRA     CMMD2

MMOVEA1  CLR.L   D3
         MOVE.B  #2,TNB(A1)

MMOVEA   DS      0              (INDEX 32)
         MOVE.L  PTROP(A1),A5   A5 = POINTER TO OPERAND

         MOVE.W  TLENGTH(A1),D2 D0 = SIZE
         BEQ     ER             .BYTE NOT ALLOWED

         LSL.W   #6,D2          .SIZE
         BTST    #12,D2
         BEQ.S   MMA225         .L = ..10
         ORI.W   #$3000,D2      .W = ..11
MMA225

         BSR     EAALL          ALL MODES ALLOWED
         OR.W    D6,D2

         BSR     COMMA

         MOVE.W  #$0002,D7      A@ ONLY
         BSR     EA
         BRA     MM825


MJMP     DS      0              (INDEX 18)
         TST.B   TLSPEC(A1)
```

```
         BEQ.S   MJMP32          DEFAULT (ALLOW EITHER .S OR .L)
         MOVE.W  TLENGTH(A1),D0
         BEQ     ER              .B NOT ALLOWED
         MOVE.W  #$6E4,D7        D7 = MODES ALLOWED
         CMPI.W  #$40,D0
         BEQ.S   MJMP22          .S SPECIFIED (.W ACCEPTED)
         MOVE.W  #$8764,D7       MODE FOR .L
MJMP22   BSR     EA
         BRA.S   MJMP42

MJMP32   BSR     EAC             CONTROL ADDRESSING ONLY
MJMP42   OR.W    D6,D2
         BRA     CMMD2

* SIZE = LONG
MLEA     DS      0               (INDEX 19)
         BSR     SIZELONG

         BSR     EAC             CONTROL ADDRESSING ONLY
         OR.W    D6,D2

         BSR     COMMA

         BSR     GETREGA
         ROR.W   #7,D0
         OR.W    D0,D2
         BRA     CMMD2


* SIZE = LONG
MPEA     DS      0               (INDEX 36)
         BSR     SIZELONG

         BSR     EAC             CONTROL ADDRESSING ONLY
         OR.W    D6,D2
         BRA     CMMD2

MSWAP    DS      0               (INDEX 24)
* SIZE WORD
         CMPI.W  #$0040,TLENGTH(A1)
         BNE     ER              NOT .W

         BSR     GETREGD         D@ ONLY
         OR.W    D0,D2
         BRA     CMMD2

GETIMM   CMPI.B  #'#',(A5)+
         BNE     ER

         BSR     EV              EVALUATE EXPRESSION
         RTS                     D0 = VALUE

MLINK    BSR     GETREGA         (INDEX 20)
         OR.W    D0,D2

         BSR     COMMA

         BSR     GETIMM
         MOVE.L  D0,D5
         BSR     EA16BITS        -32K TO +32K
         MOVE.W  D0,TDATA+2(A1)

         ADDQ.B  #2,TNB(A1)
         BRA     CMMD2

MSTOP    DS      0               (INDEX 23)
* UNSIZED
         BSR     GETIMM
         CMPI.L  #$00010000,D0
         BCC     ER
         MOVE.W  D0,TDATA+2(A1)
         ADDQ.B  #2,TNB(A1)
         BRA     CMMD2

MTRAP    DS      0               (INDEX 25)
         BSR     GETIMM
         CMPI.L  #16,D0
         BCC     ER
         OR.W    D0,D2
```

```
         BRA     CMMD2

MUNLK    DS      0               (INDEX 26)
* UNSIZED
         BSR     GETREGA
         OR.W    D0,D2
         BRA     CMMD2

MDC      DS      0               (INDEX 37) .W ONLY ALLOWED
         BSR     EV
         MOVE.L  D0,D5
         BSR     EA16BIT         ONLY .W ALLOWED     -32K TO +64K
         MOVE.W  D5,D2
         BRA     CMMD2




*-------------------------------------------------------------------------
* File DCODE68K  68K ONE LINE DISASSEMBLER                        07/28/82

*        CALLING SEQUENCE:
*   D0,D1,D2 = CODE TO BE DISASSEMBLED
*   A4 = VALUE OF PROGRAM COUNTER FOR THE CODE
*   A5 = POINTER TO STORE DATA (BUFSIZE = 80 ASSUMED)
*        JSR       DCODE68K

*        RETURN:
*   A4 = VALUE OF PROGRAM COUNTER FOR NEXT INSTRUCTION
*   A5 = POINTER TO LINE AS DISASSEMBLED
*   A6 = POINTER TO END OF LINE
*
* 0123456789012345678901234567890123456789012345678901234567890123456789
* AAAAAA FDATA.DDDDDDDDDDDDDD FOC.... FOP.....

*FDATA   EQU     10              DATA FIELD
*FOC     EQU     31              OP-CODE FIELD
*FOP     EQU     39              OPERAND FIELD

* CAUSES ORGIN MODULO 4
LONG     MACRO
         DS      0
         DS.B    (*-X)&2
         ENDM


X        DS      0               BASE ADDRESS THIS MODULE
         LONG

*  MOVEM REGISTERS TO EA
*
*        01001D001S......
*        ..........XXXXXX       EFFECTIVE ADDRESS
*        .........0......       WORD
*        .........1......       LONG
*        .....0..........       REGISTER TO MEMORY
*        .....1..........       MEMORY TO REGISTER
*

IMOVEMFR DS      0
         BSR     MOVEMS          SIZE

         MOVEQ   #$0038,D6
         AND.W   (A4),D6
         CMPI.W  #$0020,D6
         BEQ.S   IM7788          PREDECREMENT MODE

         MOVEQ   #1,D6           D6 = INCREMENTER (BIT POSITION)
         MOVEQ   #0,D1           D1 = BIT POSITION
         BRA.S   IM7799

IM7788   MOVEQ   #-1,D6          D6 = DECREMENTER (BIT POSITION)
         MOVEQ   #15,D1          D1 = BIT POSITION
IM7799   BSR     MOVEMR          BUILD MASK WORD

         MOVE.B  #',',(A6)+      STORE COMMA

         ADDQ.L  #2,D3
         MOVE.W  (A4),D4
```

```
         MOVE.W  #$1F4,D7       CONTROL + PREDECREMENT
         BSR     EEA
         BRA.S   CS16           COMMON

         LONG
* MOVEM  EA  TO REGISTERS
*
IMOVEMTR BSR     MOVEMS         SIZE
         ADDQ.L  #2,D3
         MOVE.W  #$7EC,D7       CONTROL + POSTINCREMENT
         BSR     EEA

         MOVE.B  #',',(A6)+     STORE COMMA

         MOVEQ   #1,D6          D6 = BIT POSITION INCREMENTER
         MOVEQ   #0,D1          D1 = BIT POSITION
         BSR     MOVEMR

CS16     BRA.S   CS15           COMMON


         LONG
ISTOP    DS      0
         MOVE.W  2(A4),D0
         MOVE.B  #'#',(A6)+     IMMEDIATE
         MOVE.B  #'$',(A6)+     HEX
         BSR     PNT4HX         VALUE
         BRA     COMMON4

         LONG
IMMED    DS      0              ADD  AND  CMP #  EOR  OR  SUB
         BSR     FORMSIZE
         ADDQ.L  #2,D3          SIZE = 4
         MOVE.B  #'#',(A6)+     IMMEDIATE

         CLR.L   D0
         MOVE.W  2(A4),D0       D0 = EXTENSION WORD
         MOVE.W  (A4),D1
         LSR.W   #6,D1
         ANDI.W  #3,D1
         BEQ.S   IMMED65        .BYTE

         CMPI.B  #1,D1
         BEQ.S   IMMED75        .WORD

         ADDQ.L  #2,D3          .LONG    SIZE = 6
         MOVE.L  2(A4),D0       D0 = LONG EXTENSION WORD

IMMED45  BSR     HEX2DEC        DECIMAL

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE    (A4),D0
         ANDI.W  #$003F,D0
         CMPI.W  #$003C,D0      DESTINATION ADDRESS MODE 111100  "SR"
         BNE.S   IMMED55        NOT FOUND

         MOVE.W  (A4),D0        "SR"  ILLEGAL FOR
         ANDI.W  #$4000,D0      ADDI   SUBI  CMPI
         BNE     FERROR         0600   0400  0C00

         MOVE.W  (A4),D1
         ANDI.W  #$00C0,D1
         CMPI.W  #$0080,D1
         BEQ     FERROR         .LONG NOT ALLOWED

         MOVE.B  #'S',(A6)+     #,SR FOR ANDI, EORI, ORI
         MOVE.B  #'R',(A6)+
CS15     BRA.S   CS14           COMMON

IMMED55  BSR     EEA
         BRA.S   CS14           COMMON

IMMED65  MOVE.L  D0,D1          D1 = XXXXXXXX........
         LSR.W   #8,D1          D1 = 00000000XXXXXXXX
         BEQ.S   IMMED75
         MOVE.L  D0,D1
         ASR.W   #7,D1
         ADDQ.W  #1,D1          CHECK FOR NEGATIVE
```

```
         BNE     FERROR

IMMED75  EXT.L   D0
         BRA     IMMED45

*  BIT   5432109876543210
*        ....RRRMMM......       DESTINATION REGISTER MODE
*        ..........MMMRRR       SOURCE MODE REGISTER
*        0001............       .BYTE
*        0011............       .WORD
*        0010............       .LONG
*
* IF BYTE SIZE; DESTINATION ADDRESS DIRECT NOT ALLOWED
         LONG
IMOVE    DS      0
         BRA     IMOVEA1

         LONG
ILINK    DS      0
         BSR.S   FORMREGA

         MOVE.B  D5,(A6)+       COMMA SERARATOR

         MOVE.B  #'#',(A6)+
         MOVE.W  2(A4),D0
         EXT.L   D0
         BSR     HEX2DEC        DECIMAL DISPLACEMENT
         BRA     COMMON4

         LONG
FORM1    DS      0              CLR  NEG  NEGX  NOT TST
         BSR.L   FORMSIZE


*                               NBCD TAS
FORM1A   BSR     EEA            DATA ALTERABLE ONLY
CS14     BRA.S   CS13           COMMON

         LONG
FORM3    DS      0              EXT  SWAP
         BSR.S   FORMREGD
         BRA.S   CS13           COMMON

         LONG
FORM4    DS      0              TRAP
         MOVE.B  #'#',(A6)+
         MOVE.W  (A4),D0
         ANDI.L  #$0F,D0
         BSR     HEX2DEC        DECIMAL
         BRA.S   CS13           COMMON

         LONG
FORM5    DS      0              UNLNK
         BSR.S   FORMREGA
         BRA.S   CS13           COMMON

*  BIT   5432109876543210
*        ....RRR.........       ADDRESS REGISTER
*        ..........XXXXXX       EFFECTIVE ADDRESS
*
         LONG
FORM6A   DS      0              LEA
         MOVE.W  #$7E4,D7       CONTROL ADDRESSING
         BSR.S   EEA10

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.W  (A4),D4
         ROL.W   #7,D4
         BSR.S   FORMREGA
         BRA.S   CS13           COMMON

*  BIT   5432109876543210
*        ....DDD.........       DATA REGISTER
*        ..........XXXXXX       EFFECTIVE ADDRESS
*
         LONG
FORM6D   DS      0              CHK  DIVS  DIVU  MULS  MULU
         MOVE.W  #$FFD,D7       DATA ADDRESSING
```

```
         BSR.S   EEA10

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.W  (A4),D4
         ROL.W   #7,D4
         BSR.S   FORMREGD
         BRA.S   CS13           COMMON

FORMREGA MOVE.B  #'A',(A6)+     FORMAT A@
FORMREG5 ANDI.B  #$07,D4
         ORI.B   #'0',D4
         MOVE.B  D4,(A6)+
         RTS

FORMREGD MOVE.B  #'D',(A6)+     FORMAT D@
         BRA     FORMREG5

*  BIT   5432109876543210
*        ....DDD......DDD       DATA REGISTERS
*
         LONG
FORM7    DS      0              EXG
         ROL.W   #7,D4
         BSR     FORMREGD

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.W  (A4),D4
         BSR     FORMREGD
         BRA.S   CS13           COMMON

*  BIT   5432109876543210
*        ....AAA......AAA       ADDRESS REGISTERS
*
         LONG
FORM8    DS      0              EXG
         ROL.W   #7,D4
         BSR     FORMREGA

FORM815  MOVE.B  #',',(A6)+     COMMA SEPARATOR

         MOVE.W  (A4),D4
         BSR     FORMREGA
CS13     BRA     CS12           COMMON

*  BIT   5432109876543210
*        ....DDD.........       DATA REGISTER
*        ............AAA       ADDRESS REGISTER
*
         LONG
FORM9    DS      0              EXG
         ROL.W   #7,D4
         BSR     FORMREGD       DATA REGISTER
         BRA     FORM815

EEA10    BRA     EEA

*  BIT   5432109876543210
*        ..........AAAAAA         EFFECTIVE ADDRESS
*        .......MMM......         OP-MODE
*        ....RRR.........         D-REGISTER
*        .......011......         WORD  EA,A@
*        .......111......         LONG  EA,A@
*        .......000......         EA,D@ BYTE (ADDRESS REGISTER DIRECT NOT ALLOWED)
*        .......0........         EA,D@
*        .......1........         D@,EA
*        ........00......         BYTE
*        ........01......         WORD
*        ........10......         LONG
*
         LONG
*                               ADD <EA>,A@   CMP <EA>,A@   SUB <EA>,A@
FORM10EX DS      0              ADD  CMP  SUB
         MOVE.W  #$FFF,D7       ALL MODES ALLOWED
         MOVE.L  D4,D0
         ANDI.W  #$01C0,D0
         BEQ.S   FORM103        .......000......
         CMPI.W  #$01C0,D0
```

```
         BEQ.S   FORM10E3       .......111......
         CMPI.W  #$00C0,D0
         BNE.S   FORM10E6

         MOVE.B  #'.',(A5)+     .......011......       STORE PERIOD
         MOVE.B  #'W',(A5)+
         BRA.S   FORM10E4

FORM10E3 MOVE.B  #'.',(A5)+
         MOVE.B  #'L',(A5)+

FORM10E4 BSR     EEA10

         MOVE.B  D5,(A6)+       STORE COMMA SEPARATOR

         MOVE.W  (A4),D4
         ROL.W   #7,D4
         BSR     FORMREGA       <EA>,A@
         BRA.S   CS12           COMMON

FORM10E6 BTST.B  #0,(A4)
         BNE.S   FORM105        .......1........    D@,<EA>
         BRA.S   FORM104        .......0........    <EA>,D@

*  BIT   5432109876543210
*        ..........AAAAAA       EFFECTIVE ADDRESS
*        .......MMM......       OP-MODE
*        ....RRR.........       D-REGISTER
*        .......0........       EA,D@
*        .......1........       D@,EA
*        .........00......       BYTE
*        .........01......       WORD
*        .........10......       LONG
         LONG
FORM10   DS      0              AND  EOR  OR
         BTST.B  #0,(A4)
         BNE.S   FORM105

FORM103  MOVE.W  #$FFD,D7       DATA ADDRESSING
FORM104  BSR     FORMSIZE
         BSR     EEA10          <EA>,D@

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.B  (A4),D4
         LSR.B   #1,D4
         BSR     FORMREGD
         BRA.S   CS12           COMMON

FORM105  BSR     FORMSIZE       D@,<EA>
         MOVE.B  (A4),D4
         LSR.B   #1,D4
         BSR     FORMREGD

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.W  (A4),D4
         MOVE.W  #$1FD,D7       ALTERABLE MEMORY ADDRESSING
         BSR     EEA10
CS12     BRA     COMMON

         LONG
*                               PEA     (JMP  JSR)
FORM11   MOVE.W  #$7E4,D7       CONTROL ADDERSSING
         BSR     EEA10
         BRA     CS12           COMMON

         LONG
*                               JMP  JSR
FORM11SL MOVE.L  D4,D0          LOOK FOR .S  OR  .L
         ANDI.W  #$3F,D0
         CMPI.W  #$38,D0
         BNE.S   FORM112        NOT .S
         MOVE.B  #'.',(A5)+     PERIOD
         MOVE.B  #'S',(A5)+     S
FORM112  CMPI.W  #$39,D0
         BNE.S   FORM114
         MOVE.B  #'.',(A5)+     PERIOD
         MOVE.B  #'L',(A5)+     L
```

```
FORM114  BRA     FORM11

*  BIT   5432109876543210
*        ....XXX.....0...       DATA DESTINATION REGISTER
*        ....XXX.....1...       ADDRESS REGISTER
*        ....XXX.00......       BYTE
*        ........01......       WORD
*        ........10......       LONG
*        ............0...       DATA REGISTER TO DATA REGISTER
*        ............1...       MEMORY TO MEMORY
*        ............0XXX       DATA SOURCE REGISTER
*        ............1XXX       ADDRESS SOURCE REGISTER
*
         LONG
FORM12   DS      0              ABCD  ADDX  SBCD  SUBX
         BSR     FORMSIZE

         BTST    #3,D4
         BNE.S   FORM125

         BSR     FORMREGD       D@,D@;   FORMAT SOURCE

         MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.B  (A4),D4
         LSR.B   #1,D4
         BSR     FORMREGD       FORMAT DESTINATION
         BRA.S   CS11           COMMON

FORM125  MOVE.B  #'-',(A6)+     -
         MOVE.B  #'(',(A6)+     (
         BSR     FORMREGA       A@    SOURCE

         MOVE.L  #'(-,)',D0     ),-(
         BSR.S   SCHR           STORE CHARS

         MOVE.B  (A4),D4
         LSR.B   #1,D4
         BSR     FORMREGA       A@   DESTINATION
         MOVE.B  #')',(A6)+
         BRA.S   CS11           COMMON

*  BIT   5432109876543210
*        ....XXX.....1...       ADDRESS REGISTER    DESTINATION
*        ....XXX.00......       BYTE
*        ........01......       WORD
*        ........10......       LONG
*        ............1...       MEMORY TO MEMORY
*        ............1XXX       ADDRESS SOURCE REGISTER
*
         LONG
FORM12A  DS      0              CMPM
         BSR     FORMSIZE

         MOVE.B  #'(',(A6)+     (
         BSR     FORMREGA       A@

         MOVE.L  #'(,+)',D0     )+,(
         BSR.S   SCHR           STORE CHARS

         MOVE.B  (A4),D4
         LSR.B   #1,D4
         BSR     FORMREGA       A@
         MOVE.B  #')',(A6)+
         MOVE.B  #'+',(A6)+
CS11     BRA     COMMON

         LONG
IQUICK   BRA     IQUICKA        ADDQ  SUBQ

*  BIT   5432109876543210
*        0111...0........       FIXED
*        ....RRR.........       DATA REGISTER
*        ........DDDDDDDD       SIGN EXTENDED DATA
*
         LONG
IMOVEQ   DS      0
         MOVE.B  #'#',(A6)+     IMMEDIATE
```

```
         MOVE.W  (A4),D0
         EXT.W   D0
         EXT.L   D0
         BSR     HEX2DEC         DECIMAL

         MOVE.B  D5,(A6)+        COMMA SEPARATOR

         ROL.W   #7,D4
         BSR     FORMREGD
         BRA     CS11            COMMON

SCHR     MOVE.B  D0,(A6)+        OUTPUT STRING
         LSR.L   #8,D0
         BNE     SCHR            MORE TO OUTPUT
         RTS

* MOVE FROM SR  (STATUS REGISTER)
*
         LONG
IMVFSR   MOVE.L  #',RS'+0,D0   SR,

         BSR     SCHR
         BSR     EEA             DATA ALTERABLE
         BRA     CS11            COMMON

* MOVE FROM USP (USER STACK POINTER)
*
         LONG
IMVFUSP  MOVE.L  #',PSU',D0     USP,
         BSR     SCHR
         BSR     FORMREGA
         BRA     CS11            COMMON

* MOVE TO SR (STATUS REGISTER)
*
         LONG
IMVTSR   MOVE.W  #$FFD,D7        DATA ADDRESSING
         BSR     EEA
         MOVE.L  #'RS,'+0,D0   ,SR
IMVT44   BSR     SCHR
         BRA     CS11            COMMON

* MOVE TO USP (USER STACK POINTER)
*
         LONG
IMVTUSP  BSR     FORMREGA
         MOVE.L  #'PSU,',D0     ,USP
         BRA     IMVT44

*  MOVE TO CCR (CONDITION CODE REGISTER)
*
         LONG
IMVTCCR  MOVE.W  #$FFD,D7        DATA ADDRESSING
         BSR     EEA
         MOVE.L  #'RCC,',D0     ,CCR
         BRA     IMVT44

*  BIT   5432109876543210
*        0000...1..001...        FIXED
*        ....XXX.........        DATA REGISTER
*        .........0.......       MEMORY TO REGISTER
*        .........1.......       REGISTER TO MEMORY
*        ..........0......       WORD
*        ..........1......       LONG
*        ..............XXX       ADDRESS REGISTER
*
         LONG
IMOVEP   DS      0
         MOVE.B  #'.',(A5)+     D@,#(A@)
         MOVE.W  #'LW',D0
         BTST    #6,D4
         BEQ.S   IMOVEP11        USE "W"
         LSR.W   #8,D0           USE "L"
IMOVEP11 MOVE.B  D0,(A5)+        LENGTH

         MOVE.B  (A4),D4
         LSR.B   #1,D4

         BTST.B  #7,1(A4)
```

```
         BEQ.S   IMOVEP35

         BSR     FORMREGD        D@,$HHHH(A@)

         MOVE.B  D5,(A6)+        COMMA SEPARATOR

         MOVE.W  (A4),D4
         BSR.S   IMOVEP66
CS20     BRA     COMMON4

IMOVEP35 BSR.S   IMOVEP66        $HHHH(A@),D@

         MOVE.B  D5,(A6)+        COMMA SEPARATOR

         MOVE.B  (A4),D4
         LSR.B   #1,D4
         BSR     FORMREGD
         BRA     CS20            COMMON4

IMOVEP66 MOVE.B  #'$',(A6)+      FORMAT DISPLACEMENT
         MOVE.W  2(A4),D0
         BSR     PNT4HX

         MOVE.B  #'(',(A6)+

         MOVE.W  (A4),D4
         BSR     FORMREGA
         MOVE.B  #')',(A6)+
         RTS

         LONG
SCOMMON  BRA     COMMON          NOP RESET RTE RTR RTS TRAPV

         LONG
ISCC     BSR     ICCCC           GET REST OF OP-CODE
         BSR     EEA             DATA ALTERABLE
         BRA     SCOMMON


         LONG
IDBCC    DS      0               DB--
         MOVE.W  (A4),D4
         BSR     FORMREGD

         MOVE.B  D5,(A6)+        COMMA SEPARATOR
         MOVE.B  #'$',(A6)+      HEX FIELD TO FOLLOW

         BSR     ICCCC
         BRA.S   ICC55

*  BIT   5432109876543210
*        0110............        FIXED
*        ....CCCC........        CONDITION
*        ........DDDDDDD0        DISPLACEMENT
*        ................1       ERROR (ODD BOUNDRY DISPLACEMENT)
*
         LONG
ICC      DS      0               B--
         BSR     ICCCC

IBSR     MOVE.B  #'$',(A6)+      BSR   BRA

         TST.B   D4
         BEQ.S   ICC55           16 BIT DISPLACEMENT

         MOVE.B  #'.',(A5)+
         MOVE.B  #'S',(A5)+
         EXT.W   D4              8 BIT DISPLACEMENT

ICC35    EXT.L   D4              SIGN-EXTENDED DISPLACEMENT
         ADD.L   HISPC(A1),D4    + PROGRAM COUNTER
         ADDQ.L  #2,D4           + TWO
         MOVE.L  D4,D0

         ASR.L   #1,D4
         BCS     FERROR          ODD BOUNDRY DISPLACEMENT

         BSR     PNT6HX
         BRA     SCOMMON
```

```
ICC55    ADDQ.L  #2,D3          SIZE
         MOVE.W  2(A4),D4
         MOVE.B  #'.',(A5)+
         MOVE.B  #'L',(A5)+     .L FOR 16 BIT DISPLACEMENT
         BRA     ICC35

         LONG
*                               BCHG  BCLR  BSET  BTST
ISETD    DS      0              DYNAMIC BIT
         ROL.W   #7,D4
         BSR     FORMREGD       DATA REGISTER

ISETD12  MOVE.B  D5,(A6)+       COMMA SEPARATOR

         MOVE.W  (A4),D4
         BSR     EEA            DATA ALTERABLE
CS18     BRA     SCOMMON

         LONG
*                           BCHG  BCLR  BSET  BTST
*  1ST WORD     .... .... ..XX XXXX    EA   DATA ALTERABLE ONLY
*  2ND WORD     0000 0000 000Y YYYY    BIT NUMBER
*
ISETS    DS      0              STATIC BIT
         ADDQ.L  #2,D3     SIZE
         MOVE.B  #'#',(A6)+     IMMEDIATE

         CLR.L   D0
         MOVE.W  2(A4),D0       GET BIT POSITION FROM 2ND WORD
         MOVE.L  D0,D1
         LSR.L   #5,D1
         BNE     FERROR
         BSR     HEX2DEC        DECIMAL

         BRA     ISETD12

*   BIT  5432109876543210
*        ....XXX.........       IMMEDIATE COUNT/REGISTER
*        .......0........       RIGHT SHIFT
*        .......1........       LEFT SHIFT
*        ........00......       BYTE
*        ........01......       WORD
*        ........10......       LONG
*        ....0...11......       WORD (MEMORY)
*        ....0...11AAAAAA       EFFECTIVE ADDRESS
*        ..........0.....       SHIFT IMMEDIATE COUNT
*        ..........1.....       SHIFT COUNT (MODULO 64) IN DATA REGISTER
*
         LONG
ISHIFT   DS      0              AS-  LS-  RO-  ROX-
         MOVE.W  #'LR',D0
         BTST    #8,D4          DIRECTION BIT
         BEQ.S   ISHIFT13       RIGHT
         LSR.W   #8,D0          LEFT
ISHIFT13 MOVE.B  D0,(A5)+       DIRECTION; "L" OR "R"

         MOVE.W  (A4),D0
         ANDI.W  #$00C0,D0
         CMPI.W  #$00C0,D0
         BEQ.S   ISHIFTM1       MEMORY SHIFT

         BSR     FORMSIZE

         ROL.W   #7,D4
         BTST    #12,D4         I/R BIT
         BNE.S   ISHIFT33       COUNT IN REGISTER

         ANDI.B  #$07,D4        IMMEDIATE COUNT
         BNE.S   ISHIFT23
         ORI.B   #$08,D4        CHANGE ZERO TO EIGHT
ISHIFT23 ORI.B   #'0',D4
         MOVE.B  #'#',(A6)+
         MOVE.B  D4,(A6)+
         BRA.S   ISHIFT44

ISHIFT33 BSR     FORMREGD

ISHIFT44 MOVE.B  D5,(A6)+       COMMA SEPARATOR
```

```
         MOVE.W  (A4),D4
         BSR     FORMREGD
CS17     BRA     CS18            COMMON

ISHIFTM1 MOVE.B  #'.',(A5)+      PERIOD
         MOVE.B  #'W',(A5)+      .WORD

         BTST    #11,D4
         BNE     FERROR          BIT 11 MUST BE ZERO

         MOVE.W  #$1FC,D7        MEMORY ALTERABLE ADDRESSING
         BSR     EEA
         BRA     CS17            COMMON

ICCCC    MOVEQ   #$0F,D0         APPEND CONDITION CODE
         AND.B   (A4),D0         D0 = CCC
         LSL.L   #1,D0           D0 = CCC*2

         MOVE.W  BRTBL(PC,D0.W),D1  GET BRANCH MNEMONIC
         MOVE.B  D1,(A5)+           (REVERSED) FROM THE TABLE
         LSR.W   #8,D1              AND ADD THE NONBLANK PORTION
         CMPI.B  #BLANK,D1          TO THE BUFFER.
         BEQ.S   ICCCC9
         MOVE.B  D1,(A5)+
ICCCC9   RTS

BRTBL    DC.B    ' T'       'T '  BRA ACCEPTED
         DC.B    ' F'       'F '
         DC.B    'IH'       'HI'
         DC.B    'SL'       'LS'
         DC.B    'CC'       'CC'
         DC.B    'SC'       'CS'
         DC.B    'EN'       'NE'
         DC.B    'QE'       'EQ'
         DC.B    'CV'       'VC'
         DC.B    'SV'       'VS'
         DC.B    'LP'       'PL'
         DC.B    'IM'       'MI'
         DC.B    'EG'       'GE'
         DC.B    'TL'       'LT'
         DC.B    'TG'       'GT'
         DC.B    'EL'       'LE'

*   BIT  5432109876543210
*        ....RRRMMM......    DESTINATION REGISTER MODE
*        ..........MMMRRR    SOURCE MODE REGISTER
*
* IF BYTE SIZE; ADDRESS DIRECT NOT ALLOWED AS SOURCE
*
IMOVEA1  DS      0
         MOVE.W  #$FFF,D7        ALL MODES
         BSR     EEA

         MOVE.B  D5,(A6)+        COMMA SEPARATOR

         MOVE.W  (A4),D4         ....RRRMMM......
         LSR.W   #1,D4           .....RRRMMM.....
         LSR.B   #5,D4           .....RRR.....MMM
         ROR.W   #8,D4           .....MMM.....RRR
         LSL.B   #5,D4           .....MMMRRR.....
         LSR.W   #5,D4           ..........MMMRRR

* IF .BYTE DESTINATION A@ NOT ALLOWED
         MOVE.W  #$1FF,D7        DATA ALTERABLE + A@
         MOVE.B  (A4),D0
         CMPI.B  #$01,D0
         BNE.S   IMOVE19         NOT BYTE SIZE

         MOVE.W  #$1FD,D7        DATA ALTERABLE
IMOVE19

         BSR     EEA
         BRA.S   CS19            COMMON

*  IF BYTE; ADDRESS REGISTER DIRECT NOT ALLOWED
IQUICKA  DS      0               ADDQ  SUBQ
         BSR.S   FORMSIZE
```

```
         MOVE.B  #'#',(A6)+
         ROL.W   #7,D4
         ANDI.B  #7,D4
         BNE.S   IQUICK21
         ORI.B   #8,D4           MAKE ZERO INTO EIGHT
IQUICK21 ORI.B   #'0',D4         MAKE ASCII
         MOVE.B  D4,(A6)+

         MOVE.B  D5,(A6)+        COMMA SEPARATOR

         MOVE.W  (A4),D4

         MOVE.W  (A4),D0
         ANDI.W  #$00C0,D0
         BEQ.S   IQUICK31        DATA ALTERABLE
         MOVE.W  #$1FF,D7        ALTERABLE ADDRESSING
IQUICK31 BSR     EEA
CS19     BRA     COMMON

*  BIT   5432109876543210
*        ........00......       BYTE
*        ........01......       WORD
*        ........10......       LONG
*        ........11......       ERROR
*
FORMSIZE DS      0
         MOVE.W  (A4),D2
         MOVE.B  #'.',(A5)+      STORE PERIOD
         LSR.W   #6,D2
         ANDI.W  #$03,D2
         BNE.S   FORM91
         MOVE.B  #'B',(A5)+      STORE "B"
         BRA.S   FORM95

FORM91   MOVE.B  #'W',D0
         CMPI.B  #1,D2
         BEQ.S   FORM93
         MOVE.B  #'L',D0
         CMPI.B  #2,D2
         BNE.S   FE10            FERROR
FORM93   MOVE.B  D0,(A5)+        STORE "W" OR "L"
FORM95   RTS

EA000    BSR     FORMREGD
         BTST    #0,D7
         BEQ.S   FE10            FERROR
         RTS

EA001    BSR     FORMREGA
         BTST    #1,D7
         BEQ.S   FE10            FERROR  THIS MODE NOT ALLOWED
         RTS

EA010    MOVE.B  #'(',(A6)+
         BSR     FORMREGA
         MOVE.B  #')',(A6)+
         BTST    #2,D7
         BEQ.S   FE10            FERROR  THIS MODE NOT ALLOWED
         RTS

EA011    MOVE.B  #'(',(A6)+
         BSR     FORMREGA
         MOVE.B  #')',(A6)+
         MOVE.B  #'+',(A6)+
         BTST    #3,D7
         BEQ.S   FE10            FERROR  THIS MODE NOT ALLOWED
EA011RTS RTS

EA100    MOVE.B  #'-',(A6)+
         MOVE.B  #'(',(A6)+
         BSR     FORMREGA
         MOVE.B  #')',(A6)+
         BTST    #4,D7
         BNE     EA011RTS
FE10     BRA     FERROR          THIS MODE NOT ALLOWED

*  ENTER       A4 = POINTER TO FIRST WORD
*              D3 = OFFSET TO EXTENSION
*              D4 = VALUE TO PROCESS
```

```
*              D7 = MODES ALLOWED MASK
*
EEA      DS      0
         MOVE.L  D4,D0
         LSR.W   #3,D0
         ANDI.W  #$7,D0
         BEQ     EA000

         CMPI.B  #1,D0
         BEQ     EA001

         CMPI.B  #2,D0
         BEQ     EA010

         CMPI.B  #3,D0
         BEQ     EA011

         CMPI.B  #4,D0
         BEQ     EA100

         CMPI.B  #5,D0
         BEQ.S   EA101

         CMPI.B  #7,D0
         BEQ.S   EA111

*    EXTENSION WORD
*   BIT  5432109876543210
*        0...............    DATA REGISTER
*        1...............    ADDRESS REGISTER
*        .RRR............    REGISTER
*        ....0...........    SIGN EXTENDED, LOW ORDER INTEGER IN INDEX REG
*        ....1...........    LONG VALUE IN INDEX REGISTER
*        .....000........
*        ........DDDDDDDD    DISPLACEMENT INTEGER
*
* EA110            ADDRESS REGISTER INDIRECT WITH INDEX

         BTST    #6,D7
         BEQ     FE10           FERROR  THIS MODE NOT ALLOWED

         MOVE.W  (A4,D3),D1
         ANDI.W  #$0700,D1
         BNE     FE10           FERROR  BITS 10-8 MUST BE ZERO

         MOVE.W  (A4,D3),D0     D0 = DISPLACEMENT
         EXT.W   D0
         EXT.L   D0
         BSR     HEX2DEC        DECIMAL
         MOVE.B  #'(',(A6)+     (

         BSR     FORMREGA       XX(A@

         MOVE.B  #',',(A6)+     XX(A@,

         MOVE.B  (A4,D3),D4
         ASR.B   #4,D4
         BPL.S   EA1105
         BSR     FORMREGA
         BRA.S   EA1107

EA1105   BSR     FORMREGD
EA1107   MOVE.B  #'.',(A6)+     XX(A@,X@.

         MOVE.W  (A4,D3),D4     D4 = R@
         MOVE.B  #'W',D0        ..........W
         BTST    #11,D4
         BEQ.S   EA1109
         MOVE.B  #'L',D0        ..........L
EA1109   MOVE.B  D0,(A6)+
         MOVE.B  #')',(A6)+     ...........)
         ADDQ.L  #2,D3
         RTS

* ADDRESS REGISTER INDIRECT WITH DISPLACEMENT
*
EA101    BTST    #5,D7          101000;   DIS(A@)
         BEQ.S   FE11           FERROR;  THIS MODE NOT ALLOWED
```

```
        MOVE.W  (A4,D3),D0
        EXT.L   D0
        BSR     HEX2DEC         DECIMAL
        ADDQ.L  #2,D3           SIZE
        BRA     EA010

*  111000       ABSOLUTE SHORT
*  111001       ABSOLUTE LONG
*  111010       PROGRAM COUNTER WITH DISPLACEMENT
*  111011       PROGRAM COUNTER WITH INDEX
*  111100       IMMEDIATE OR STATUS REG
*
EA111
        ANDI.W  #7,D4
        BNE.S   EA1112

        BTST    #7,D7
        BEQ.S   FE11            FERROR;  THIS MODE NOT ALLOWED

        MOVE.W  (A4,D3),D0      111000;   ABSOLUTE SHORT
        EXT.L   D0
        MOVE.B  #'$',(A6)+
        BSR     PNT8HX          SIGN EXTENDED VALUE
        ADDQ.L  #2,D3           SIZE + 2
        RTS

EA1112  CMPI.B  #1,D4
        BNE.S   EA1113

        BTST    #8,D7
        BEQ.S   FE11            FERROR;  THIS MODE NOT ALLOWED

        MOVE.B  #'$',(A6)+      HEX
        MOVE.L  (A4,D3),D0      111001;     ABSOLUTE LONG
        BSR     PNT8HX
*-      MOVE.B  #'.',(A6)+      FORCE LONG FORMAT
*-      MOVE.B  #'L',(A6)+      IE   .L
        ADDQ.L  #4,D3
        RTS

EA1113  CMPI.B  #2,D4
        BNE.S   EA1114

        BTST    #9,D7
        BNE.S   EA1113A
FE11    BRA     FERROR          THIS MODE NOT ALLOWED
EA1113A

        MOVE.W  (A4,D3),D0      111010;  PC + DISPLACEMENT  DESTINATION(PC)
        EXT.L   D0
        ADD.L   HISPC(A1),D0
        ADDQ.L  #2,D0
        MOVE.B  #'$',(A6)+      HEX "$"
        BSR     PNT8HX          DESTINATION
        MOVE.L  #')CP(',D0      (PC)
        BSR     SCHR            STORE WORD
        ADDQ.L  #2,D3           SIZE
        RTS

EA1114  CMPI.B  #3,D4
        BNE.S   EA1115

* PROGRAM COUNTER WITH INDEX    DESTINATION(PC,R@.X)
*
*       5432109876543210        SECOND WORD
*       0...............        DATA REGISTER
*       1...............        ADDRESS REGISTER
*       .XXX............        REGISTER
*       ....0...........        SIGN-EXTENDED, LOW ORDER WORD INTEGER
*                               ..IN INDEX REGISTER
*       ....1...........        LONG VALUE IN INDEX REGISTER
*       .....000........
*       ........XXXXXXXX        DISPLACEMENT INTEGER
*
        BTST    #10,D7
        BEQ     FE11            FERROR  THIS MODE NOT ASLLOWED

        MOVE.W  (A4,D3),D1
        ANDI.W  #$0700,D1
```

```
         BNE     FE11           FERROR;  BITS 10-8 MUST BE ZERO

         MOVE.B  1(A4,D3),D0    111100;   DESTINATION(PC,R@.X)
         EXT.W   D0
         EXT.L   D0
         ADD.L   HISPC(A1),D0
         ADDQ.L  #2,D0
         MOVE.B  #'$',(A6)+     HEX "$"
         BSR     PNT8HX         DESTINATION

         MOVE.L  #',CP(',D0
         BSR     SCHR           DES(PC,

         MOVE.W  (A4,D3),D4
         ROL.W   #4,D4
         BTST    #3,D4
         BEQ.S   EAF25
         BSR     FORMREGA
         BRA.S   EAF27
EAF25    BSR     FORMREGD       DES(PC,R@
EAF27

         MOVE.B  #'.',(A6)+     DES(PC,R@.

         MOVE.W  (A4,D3),D4
         MOVE.W  #'LW',D0
         BTST    #11,D4
         BEQ.S   EAF35
         LSR.W   #8,D0
EAF35    MOVE.B  D0,(A6)+       DES(PC,R@.X

         MOVE.B  #')',(A6)+     DES(PC,R@.X)
         ADDQ.L  #2,D3
         RTS

*   BIT  5432109876543210
*        ..........111100       FIRST WORD;  #<IMMEDIATE>
*
EA1115   CMPI.B  #4,D4
         BNE     FE11           FERROR

         BTST    #11,D7
         BEQ     FE11           FERROR;  THIS MODE NOT ALLOWED

         MOVE.B  #'#',(A6)+     IMMEDIATE

         MOVE.B  -1(A5),D1
         CMPI.B  #'L',D1
         BEQ.S   EA11155        LONG

         MOVE.W  (A4,D3),D0

         CMPI.B  #'B',D1
         BNE.S   EA11153        .WORD

* BYTE SIZE; DATA ALLOWED
*  0000 0000 XXXX XXXX
*  1111 1111 1XXX XXXX
         MOVE.L  D0,D1
         LSR.W   #8,D1
         BEQ.S   EA11153
         MOVE.L  D0,D1
         ASR.W   #7,D1
         ADDQ.W  #1,D1
         BNE     FE11           FERROR

EA11153  EXT.L   D0
         BSR     HEX2DEC
         ADDQ.L  #2,D3
         RTS

EA11155  MOVE.L  (A4,D3),D0
         BSR     HEX2DEC
         ADDQ.L  #4,D3          SIZE
         RTS

MOVEMS   MOVE.B  #'.',(A5)+     PERIOD
         MOVE.W  #'LW',D0
         BTST    #6,D4
```

```
         BEQ.S   MOVEMS2
         LSR.W   #8,D0
MOVEMS2  MOVE.B  D0,(A5)+       SIZE
         RTS

* MOVEM - REGISTER EXPANSION
*
MOVEMR   DS      0
         MOVE.W  2(A4),D2       D2 = SECOND WORD
         MOVEQ   #$20,D0        D0 = SPACE
         MOVEQ   #$2F,D7        D7 = /
         SUBQ.L  #1,A6          ADJUST STORE POINTER
         MOVEQ   #$30,D5        D5 = REGISTER #
         MOVE.W  #'AD',D4       D4 = REG CLASS

MOVEMR11 BTST    D1,D2
         BEQ.S   MOVEMR77       BIT RESET

         CMP.B   (A6),D0        BIT SET
         BNE.S   MOVEMR44       NOT SPACE

MOVEMR33 MOVE.B  D4,1(A6)       REG TYPE
         MOVE.B  D5,2(A6)       REG #
         MOVE.B  #'-',3(A6)     -
         ADDQ.L  #3,A6
         BRA.S   MOVEMR88

MOVEMR44 CMPI.B  #',',(A6)
         BEQ     MOVEMR33       COMMA SEPARATOR

         CMP.B   (A6),D7        / SEPARATOR
         BEQ     MOVEMR33

         MOVE.B  D4,1(A6)       REG TYPE
         MOVE.B  D5,2(A6)       REG #
         MOVE.B  #'-',3(A6)     - SEPARATOR
         BRA.S   MOVEMR88

MOVEMR77 CMPI.B  #',',(A6)
         BEQ.S   MOVEMR88       COMMA

         CMP.B   (A6),D0
         BEQ.S   MOVEMR88       SPACE
         CMP.B   1(A6),D0
         BEQ.S   MOVEMR79       SPACE
         ADDQ.L  #3,A6
MOVEMR79 MOVE.B  D7,(A6)        / SEPARATOR

MOVEMR88 ADDQ.L  #1,D5
         ADD.L   D6,D1          D1 = BIT POSITION
         CMPI.B  #'8',D5
         BNE     MOVEMR11

         CMP.B   (A6),D0        SPACE
         BEQ.S   MOVEMR94

         CMP.B   1(A6),D0       SPACE
         BEQ.S   MOVEMR94
         ADDQ.L  #3,A6
         MOVE.B  D7,(A6)        /   SEPARATOR

MOVEMR94 MOVE.B  #'0',D5        RESET REG TO ZERO
         LSR.W   #8,D4          CHANGE REG TYPE
         BNE     MOVEMR11       MORE

         MOVE.B  D0,(A6)        SPACE
         RTS

DCODE68K DS      0
         LINK    A1,#-LOCVARSZ       CREATE A FRAME FOR THE
         MOVEM.L D0-D2/A4,DDATA(A1)  CODE AND ITS PC.  A4
         LEA     DDATA(A1),A4        POINTS TO THE CODE.

         MOVE.L  A5,A3          A3 = START OF OUTPUT BUFFER
         MOVEQ   #BUFSIZE,D0
         MOVE.L  A3,A6
DEC311   MOVE.B  #BLANK,(A6)+   SPACE FILL BUFFER
         SUBQ.L  #1,D0
         BNE     DEC311
```

```
         MOVE.L  A3,A6          FORMAT ADDRESS
         MOVE.L  HISPC(A1),D0
         BSR     FRELADDR

* CHECK FOR KNOWN ILLEGAL CODES
         MOVE.W  (A4),D0

         LEA     KI(PC),A5
         MOVE.L  A5,A6
         ADD.L   #KIEND-KI,A6
DEC404   CMP.W   (A5)+,D0
         BEQ.S   FE12           FERROR;  ILLEGAL CODE
         CMP.L   A6,A5
         BNE     DEC404

* LOOK FOR MATCH OF OP-CODE
*
         LEA     TBL(PC),A5     A5 = POINTER TO DECODE TABLE
         LEA     TBLE(PC),A6    A6 = POINTER TO END OF TABLE
DEC411   MOVE.W  (A4),D0        FIRST WORD
         AND.W   (A5)+,D0       MASK
         CMP.W   (A5)+,D0
         BEQ.S   DEC425         FOUND MATCH
         ADDQ.L  #2,A5          UPDATE POINTER
         CMP.L   A6,A5
         BNE     DEC411         MORE TABLE
FE12     BRA.S   FERROR         ILLEGAL INSTRUCTION

DEC425   CLR.L   D6
         MOVE.B  (A5)+,D6       D6 = (GOTO OFFSET)/4
         LSL.L   #2,D6

         CLR.L   D7
         MOVE.B  (A5)+,D7       D7 = INDEX TO OP-CODE

* MOVE OP-CODE TO BUFFER
*
         LEA     OPCTBL(PC),A0
DEC510   TST     D7
         BEQ.S   DEC530         AT INDEX
DEC515   TST.B   (A0)+
         BPL     DEC515         MOVE THROUGH FIELD
         SUBQ.L  #1,D7
         BRA     DEC510

DEC530   MOVEQ   #FOC,D0
         LEA.L   (A3,D0),A5     A5 = STORE POINTER  OP-CODE
DEC535   MOVE.B  (A0)+,D0
         BCLR    #7,D0
         BNE.S   DEC537         END OF MOVE
         MOVE.B  D0,(A5)+
         BRA     DEC535
DEC537   MOVE.B  D0,(A5)+

* CALCULATE GOTO AND GO
*
         MOVEQ   #2,D3          D3= SIZE
         LEA     X(PC),A0
         ADD.L   D6,A0

         MOVEQ   #FOP,D0
         LEA.L   (A3,D0),A6     A6 = POINTER FOR OPERAND

         MOVE.W  (A4),D4        D4 = FIRST WORD

         MOVE.B  #',',D5        D5 = CONTAINS ASCII COMMA

         MOVE.W  #$1FD,D7       D7 = DATA ALTERABLE MODES ALLOWED

         JMP     (A0)
*
*  A4 = POINTER TO DATA IN FRAME CREATED BY 'LINK A1,...'
*  A5 = POINTER STORE OP-CODE
*  A6 = POINTER STORE OPERAND
*  D3 = SIZE = 2 BYTES
*  D4 = FIRST WORD
*  D7 = ADDRESS MODES ALLOWED ($1FD) DATA ALTERABLE
```

```
COMMON4  ADDQ.L  #2,D3          SIZE = 4

COMMON   MOVE.L  D3,D6          D6 = SIZE
         MOVE.B  #BLANK,(A6)+   SPACE AS LAST CHAR

         MOVE.L  A6,A5          SAVE END OF BUFFER POINTER
         MOVEQ   #FDATA,D0
         LEA.L   (A3,D0),A6

COMMON35 MOVE.W  (A4)+,D0       GET NEXT WORD OF DATA.
         ADDQ.L  #2,HISPC(A1)   ADJUST PROG COUNTER.
         BSR     PNT4HX         FORMAT DATA. (A6)+
         SUBQ.B  #2,D3
         BNE     COMMON35

         MOVE.L  A5,A6          A6 = RESTORE END POINTER

         MOVE.L  A3,A5          A5 =  BEGINNING OF BUFFER

         MOVE.L  HISPC(A1),A4   MOVE THE UPDATED PC
         UNLK    A1               TO A4 AND UNDO FRAME.

         RTS


FERROR   DS      0
* ILLEGAL INSTRUCTION
*
         MOVEQ   #FOC,D0
         LEA.L   (A3,D0),A6
         LEA     MSG111(PC),A5
FERROR35 MOVE.B  (A5)+,D0
         CMPI.B  #EOT,D0
         BEQ.S   FERROR39
         MOVE.B  D0,(A6)+
         BRA     FERROR35
FERROR39 DS      0

         MOVE.W  (A4),D0
         BSR     PNT4HX

         MOVEQ   #2,D3          SIZE

         BRA     COMMON

MSG111   DC.B    'DC.W    $',EOT



KI       DC.W    $4AFB          KNOWN ILLEGAL CODES
KIEND    DS      0


*  \1   MASK
*  \2   OP-CODE PATTERN
*  \3   GOTO OFFSET
*  \4   INDEX TO OP-CODE
C68      MACRO
         DC.W    $\1
         DC.W    $\2
         DC.B    (\3-X)>>2
         DC.B    \4
         ENDM

TBL      DS      0
         C68     FEC0,E6C0,ISHIFT,56          RO
         C68     FEC0,E4C0,ISHIFT,57          ROX
         C68     FEC0,E2C0,ISHIFT,55          LS
         C68     FEC0,E0C0,ISHIFT,54          AS
         C68     F018,E018,ISHIFT,56          RO
         C68     F018,E010,ISHIFT,57          ROX
         C68     F018,E008,ISHIFT,55          LS
         C68     F018,E000,ISHIFT,54          AS
         C68     F0C0,D0C0,FORM10EX,4         ADD       <EA>,A@
         C68     F130,D100,FORM12,53          ADDX
         C68     F000,D000,FORM10EX,4           ADD
         C68     F1F8,C188,FORM9,50           EXG
         C68     F1F8,C148,FORM8,50           EXG
         C68     F1F8,C140,FORM7,50           EXG
```

```
        C68     F1F0,C100,FORM12,49          ABCD
        C68     F1C0,C1C0,FORM6D,48          MULS
        C68     F1C0,C0C0,FORM6D,47          MULU
        C68     F000,C000,FORM10,2           AND
        C68     F0C0,B0C0,FORM10EX,6         CMP     <EA>,A@
        C68     F138,B108,FORM12A,46          CMPM
        C68     F100,B100,FORM10,5           EOR
        C68     F000,B000,FORM10EX,6           CMP
        C68     F0C0,90C0,FORM10EX,44        SUB       <EA>,A@
        C68     F130,9100,FORM12,45          SUBX
        C68     F000,9000,FORM10EX,44          SUB
        C68     F1F0,8100,FORM12,43          SBCD
        C68     F1C0,81C0,FORM6D,42          DIVS
        C68     F1C0,80C0,FORM6D,41          DIVU
        C68     F000,8000,FORM10,40          OR
        C68     F100,7000,IMOVEQ,39          MOVEQ
        C68     FF00,6100,IBSR,51            BSR
        C68     FF00,6000,IBSR,65            BRA
        C68     F000,6000,ICC,38             B
        C68     F0F8,50C8,IDBCC,37             DB
        C68     F0C0,50C0,ISCC,36            S
        C68     F100,5100,IQUICK,35          SUBQ
        C68     F100,5000,IQUICK,34          ADDQ
        C68     F1C0,41C0,FORM6A,33          LEA
        C68     F1C0,4180,FORM6D,32          CHK
        C68     FFC0,4EC0,FORM11SL,31        JMP
        C68     FFC0,4E80,FORM11SL,30        JSR
        C68     FFFF,4E77,SCOMMON,29          RTR
        C68     FFFF,4E76,SCOMMON,28          TRAPV
        C68     FFFF,4E75,SCOMMON,27          RTS
        C68     FFFF,4E73,SCOMMON,26          RTE
        C68     FFFF,4E72,ISTOP,25            STOP
        C68     FFFF,4E71,SCOMMON,24          NOP
        C68     FFFF,4E70,SCOMMON,23          RESET
        C68     FFF8,4E68,IMVFUSP,60         MOVE FROM USP
        C68     FFF8,4E60,IMVTUSP,60         MOVE TO USP
        C68     FFF8,4E58,FORM5,22           UNLINK
        C68     FFF8,4E50,ILINK,21           LINK
        C68     FFF0,4E40,FORM4,20           TRAP
        C68     FF80,4C80,IMOVEMTR,15        MOVEM FROM REGISTERS
        C68     FFC0,4AC0,FORM1A,19          TAS
        C68     FF00,4A00,FORM1,18           TST
        C68     FFF8,48C0,FORM3,17           EXT.L
        C68     FFF8,4880,FORM3,16           EXT.W
        C68     FF80,4880,IMOVEMFR,15        MOVEA TO REGISTERS
        C68     FFF8,4840,FORM3,14           SWAP
        C68     FFC0,4840,FORM11,13          PEA
        C68     FFC0,4800,FORM1A,12          NBCD
        C68     FFC0,46C0,IMVTSR,59          MOVE TO SR
        C68     FF00,4600,FORM1,11           NOT
        C68     FFC0,44C0,IMVTCCR,59         MOVE TO CCR
        C68     FF00,4400,FORM1,10           NEG
        C68     FF00,4200,FORM1,9            CLR
        C68     FFC0,40C0,IMVFSR,59          MOVE.W  FROM  SR
        C68     FF00,4000,FORM1,8            NEGX
        C68     F000,3000,IMOVE,59           MOVE.W
        C68     F000,2000,IMOVE,60           MOVE.L
        C68     F000,1000,IMOVE,58           MOVE.B
        C68     FF00,0C00,IMMED,6            CMP       #
        C68     FF00,0A00,IMMED,5            EOR       #
        C68     FF00,0600,IMMED,4            ADD       #
        C68     FF00,0400,IMMED,3            SUB       #
        C68     FF00,0200,IMMED,2            AND       #
        C68     FF00,0000,IMMED,1            OR        #
        C68     F138,0108,IMOVEP,0           MOVEP
        C68     FFC0,08C0,ISETS,64           BSET
        C68     FFC0,0880,ISETS,63           BCLR
        C68     FFC0,0840,ISETS,62           BCHG
        C68     FFC0,0800,ISETS,61           BTST
        C68     F1C0,01C0,ISETD,64           BSET
        C68     F1C0,0180,ISETD,63           BCLR
        C68     F1C0,0140,ISETD,62           BCHG
        C68     F1C0,0100,ISETD,61           BTST
TBLE    DS      0

N68     MACRO
        DC.B    '\1',128+'\2'        \1\2
        ENDM
```

```
OPCTBL   DS      0
         N68     MOVE,P    0
         N68     O,R       1
         N68     AN,D      2
         N68     SU,B      3
         N68     AD,D      4
         N68     EO,R      5
         N68     CM,P      6
         N68     MOV,E     7
         N68     NEG,X     8
         N68     CL,R      9
         N68     NE,G      10
         N68     NO,T      11
         N68     NBC,D     12
         N68     PEA.,L    13
         N68     SWAP.,W   14
         N68     MOVE,M    15
         N68     EXT.,W    16
         N68     EXT.,L    17
         N68     TS,T      18
         N68     TAS.,B    19
         N68     TRA,P     20
         N68     LIN,K     21
         N68     UNL,K     22
         N68     RESE,T    23
         N68     NO,P      24
         N68     STO,P     25
         N68     RT,E      26
         N68     RT,S      27
         N68     TRAP,V    28
         N68     RT,R      29
         N68     JS,R      30
         N68     JM,P      31
         N68     CHK.,W    32
         N68     LEA.,L    33
         N68     ADD,Q     34
         N68     SUB,Q     35
         DC.B    128+'S'   36
         N68     D,B       37
         DC.B    128+'B'   38
         N68     MOVEQ.,L  .....39
         N68     O,R       40
         N68     DIVU.,W   41
         N68     DIVS.,W   42
         N68     SBC,D     43
         N68     SU,B      44
         N68     SUB,X     45
         N68     CMP,M     46
         N68     MULU.,W   47
         N68     MULS.,W   48
         N68     ABC,D    49
         N68     EX,G      50
         N68     BS,R      .....51
         N68     NUL,L     .....52
         N68     ADD,X     53
         N68     A,S       54
         N68     L,S       55
         N68     R,O       56
         N68     RO,X      57
         N68     MOVE.,B   58
         N68     MOVE.,W   59
         N68     MOVE.,L   60
         N68     BTS,T     61
         N68     BCH,G     62
         N68     BCL,R     63
         N68     BSE,T     64
         N68     BR,A      65

         DC.B    0              PAD BYTE



*-------------------------------------------------------------------------
* File TRAP14    Trap 14 handler of "TUTOR"                      06/25/82

*        CALLING SEQUENCE
*                  D7 = XXXXXXFF   WHERE "FF" IF FUNCTION NUMBER
*                  TRAP      #14
```

```
TRAP14
         MOVEM.L D1/D7/A1-A2,-(A7)

         MOVE.L  CTLINK,A1
T100     MOVE.B  (A1),D1         D1 = FUNCTION FROM TABLE
         CMPI.B  #$FF,D1
         BEQ.S   T500            END OF TABLE

         CMPI.B  #$FE,D1
         BEQ.S   T600            LINK IN LIST

         CMP.B   D7,D1
         BEQ.S   T400            FOUND MATCH

         ADDQ.L  #4,A1
         BRA     T100

T400     MOVE.L  (A1),D1         FFAAAAAA
         ASL.L   #8,D1           AAAAAA..
         LSR.L   #8,D1           00AAAAAA  GO TO ADDRESS

         BTST.B  #5,16(A7)
         BEQ.S   T450            CALL FROM USER MODE

*  CALL FROM SUPERVISOR MODE
* STACK (WORDS)
*   +0   D1 HIGH
*   +2   D1 LOW
*   +4   D7 HIGH
*   +6   D7 LOW
*   +8   A1 HIGH
*   +10  A1 LOW
*   +12  A2 HIGH              STATUS REG
*   +14  A2 LOW               GOTO HIGH
*   +16  STATUS REG           GOTO LOW
*   +18  RETURN HIGH        RETURN HIGH
*   +20  RETURN LOW         RETURN LOW
*
         MOVE.L  16(A7),12(A7)  SR
         MOVE.L  D1,14(A7)      GOTO ADDRESS
         MOVEM.L (A7)+,D1/D7/A1 RESTORE REGISTERS
         RTE                    GOTO (AND TRIM STACK)

* STACK (WORDS)
*   +0   D1 HIGH
*   +2   D1 LOW
*   +4   D7 HIGH
*   +6   D7 LOW
*   +8   A1 HIGH
*   +10  A1 LOW
*   +12  A2 HIGH                                  (USER STACK)
*   +14  A2 LOW
*   +16  STATUS REG         STATUS REG
*   +18  RETURN HIGH        GOTO HIGH            RETURN HIGH
*   +20  RETURN LOW         GOTO LOW             RETURN LOW
*
* CALL FROM USER MODE
T450     MOVE.L  18(A7),D7      RETURN PC
         MOVE.L  D1,18(A7)      GOTO ADDRESS
         MOVE.L  USP,A1         POINTER TO USER STACK
         MOVE.L  D7,-(A1)       RETURN PC TO USER STACK
         MOVE.L  A1,USP         UPDATED USER STACK POINTER
         MOVEM.L (A7)+,D1/D7/A1/A2
         RTE

T500     MOVEM.L (A7)+,D1/D7/A1/A2
         SAVEREGS
         LEA     MSGT14(PC),A5
         BSR     FIXDATA
         BSR     OUT1CR
         BSR     TDISPLY
         BRA     MACSBUG

MSGT14   DC.B    'UNDEFINED TRAP 14',EOT
```

```
T600     MOVE.L  (A1),A1
         BRA     T100

T700     DS      0               253 APPEND NEW TABLE
         MOVE.L  A0,D1           ..AAAAAA
         MOVE.L  CTLINK,A0       A0 = LINK TO BE RETURNED
         ROL.L   #8,D1           AAAAAA..
         MOVE.B  #$FE,D1         AAAAAAFE
         ROR.L   #8,D1           FEAAAAAA
         MOVE.L  D1,CTLINK
         RTS


*  CREATE ENTRY TO FUNCTION/ADDRESS TABLE
*    FFAAAAAA
*    FF......      FUNCTION NUMBER
*    ..AAAAAA      ADDRESS OF FUNCTION

FADDR    MACRO   \1,\2
         DC.L    (\1<<24)+\2
         ENDM


CT       FADDR   253,T700        APPEND NEW TABLE
         FADDR   252,FIXDADD     APPEND DATA (A5) TO BUFFER (A6)+
         FADDR   251,FIXBUF      SET A5 & A6 AS POINTERS TO BUFFER
         FADDR   250,FIXDATA     MOVE DATA (A5) TO BUFFER; A5=BUFFER A6
         FADDR   249,FIXDCRLF
         FADDR   248,F100        OUTPUT CHAR PORT1  D0=CHAR
         FADDR   247,F110        INPUT CHAR PORT1  D0=CHAR
         FADDR   244,CHRPRINT    OUTPUT CHAR PORT3 D0=CHAR
         FADDR   243,OUTPUT      OUTPUT STRING PORT1 (A5) (A6)
         FADDR   242,OUTPUT21    OUTPUT STRING PORT2 (A5) (A6)
         FADDR   241,PORTIN1     INPUT STRING PORT1  (A5) (A6)
         FADDR   240,PORTIN20    INPUT STRING PORT2  (A5) (A6)
         FADDR   239,TAPEOUT     OUTPUT STRING TO PORT4 (A5) (A6)
         FADDR   238,TAPEIN      INPUT STRING FROM PORT4 (A5) (A6)
         FADDR   237,PRCRLF      OUTPUT STRING TO PORT3 (A5) (A6)
         FADDR   236,HEX2DEC     CONVERT HEX D0 TO DECIMAL (A6)+
         FADDR   235,GETHEX      GET HEX CHAR INTO D0 FROM (A5)+
         FADDR   234,PUTHEX      FORMAT HEX CHAR FROM D0 TO (A6)+
         FADDR   233,PNT2HX      FORMAT 2 HEX CHAR FROM D0 TO (A6)+
         FADDR   232,PNT4HX      FORMAT 4 HEX CHAR FROM D0 TO (A6)+
         FADDR   231,PNT6HX      FORMAT 6 HEX CHAR FROM D0 TO (A6)+
         FADDR   230,PNT8HX      FORMAT 8 HEX CHAR FROM D0 TO (A6)+
         FADDR   229,START       RESTART TUTOR INITIALIZE EVERYTHING
         FADDR   228,MACSBUG     GOTO TUTOR;   PRINT PROMPT
         FADDR   227,F120        OUTPUT STRING,CR,LF PORT1 (A5) (A6)
         FADDR   226,GETNUMA     GET HEX NUMBER (A5)+ INTO D0
         FADDR   225,GETNUMD     GET DECIMAL NUMBER (A5)+ INTO D0
         FADDR   224,PORTIN1N    INPUT STRING PORT1 (NO AUTO LF)

         FADDR   255,$FFFFFF     END KEY

F100     BSR     GETSER1        A0 = PORT1 ADDRESS
         BRA     OUTCH

F110     BSR     GETSER1        A0 = PORT1 ADDRESS
         BRA     INCHNE

F120     BSR     OUTPUT         OUTPUT STRING,CR,LF PORT1 (A5) (A6)
         MOVEQ   #CR,D0
         BSR     F100           OUTPUT CHAR
         MOVEQ   #LF,D0
         BSR     F100           OUTPUT CHAR
         RTS


         DCB.B   $54,0          PAD BYTES


*-------------------------------------------------------------------------
* File YROM      Version/checksum/identification                 07/29/82

VERSION  EQU     4         BINARY FOR VERSION 1.3
*                3         .                  1.2; CKSUM= $44,$DB
*                2         .                  1.1; CKSUM= $66,$C1
*                1         .                  1.0
```

```
        DC.B    VERSION,VERSION
        DC.B    $2E,$BA         CHECKSUM
        DC.B    $11,$10         SOCKET I. D.
LAST    DS.W    0               LAST ADDRESS+1

        END     START
```