The MIKBUG 2.0 is implemented in the MC6846 COMBO ROM as a 2K Byte firmware program for the development and debug of MC6802/6846 systems. The ROM occupies the address space from F800 Hex to FFFF Hex. It is primarily intended to operate with the MC6802 microprocessor but will also operate with the MC6800 MPU. For MIKBUG 2 ROM to execute it assumes ther is a MC6810 RAM in address space A000 Hex to A07F Hex. It also assumes a MC6850 ACIA at address 8008 Hex. This ACIA interface allows MIKBUG 2 to operate with a teletype or a RS232 terminal depending on the type of interface drivers provided.

An audio tape interface called EXORTAPE is also provided in the MIKBUG 2. This interface assumes a PIA at 8004 Hex to implement this interface and uses one MC14583B for receiver buffer from the audio cassette.

The following is a summary of features provided with MIKBUG 2:

L		Loads formated object tapes into memory
M	NNNN	Memory Change at Location NNNN
P		Print/Punch ASCII formated object tapes
R		Display Contents of MPU User's Registers
S	1	Stop Bit Selection for ACIA
S	3	Stop Bit Selection for ACIA
B		Print out all Breakpoints
C		Continue Execution From Current Location
N		Execute Next Instruction
T	NNNN	Trace NNNN Instructions
G	NNNN	Go to User Program at Location NNNN
D		Delete All Breakpoints
U	NNNN	Reset Breakpoint at Location NNNN
V	NNNN	Set A Breakpoint at Location NNNN
E		EXORTAPE Cassette Interface
		C - Check Tape
		L - Load From Tape to Memory
		D - Dump From Memory to Tape
		S - Set Band Rate

Unsigned	Multiply	Subroutine
Signed	Multiply	Subroutine
Positive	Divide	Subroutine

* MIKBUG is a trademark of Motorola, Inc.

Austin, Texas
Microcomputer Capital of the World

Date: 8-3-77

Description of the first run engineering sample on M6846-L1 Combo:

1. It contains a 2K byte ROM with Mik-Bug II and cassette tape driver programmer, a programmable timer, and a programmable I/O peripheral interface port.

2. ROM passes functional test at 1MHz clock rate, and is selected by setting CS0 = 1 and CS1 = 1.

3. Timer and I/O can be selected by setting
   CS0 = 0, CS1 = 1,
   A3 = 1, (it will be changed to A3 = 0 in the future except this sample lot.)
   A4 = 0, A5 = 0, A6 = 0, A8 = 1,
   A7, A9 and A10 are don't care.

4. Programmable timer passes functional test at 1MHz clock rate.

5. CP1, the handshake control signal in the peripheral interface port, does not function properly. It does generate IRQ flag signal corresponding to its active transition as programmed, and it does generate input latch signal for the peripheral input data lines as programmed. But its IRQ flag is not registered in the combination status register Bit 1. Therefore it can not be cleared by "Read CSR then Read or Write Peripheral Data Register" operation. To clear the CP1 IRQ flag can only be done by PIA reset, which means to Write "1" in the peripheral control register bit 7, and then rewrite peripheral control register, peripheral Data Direction Register, and Peripheral Data Register.

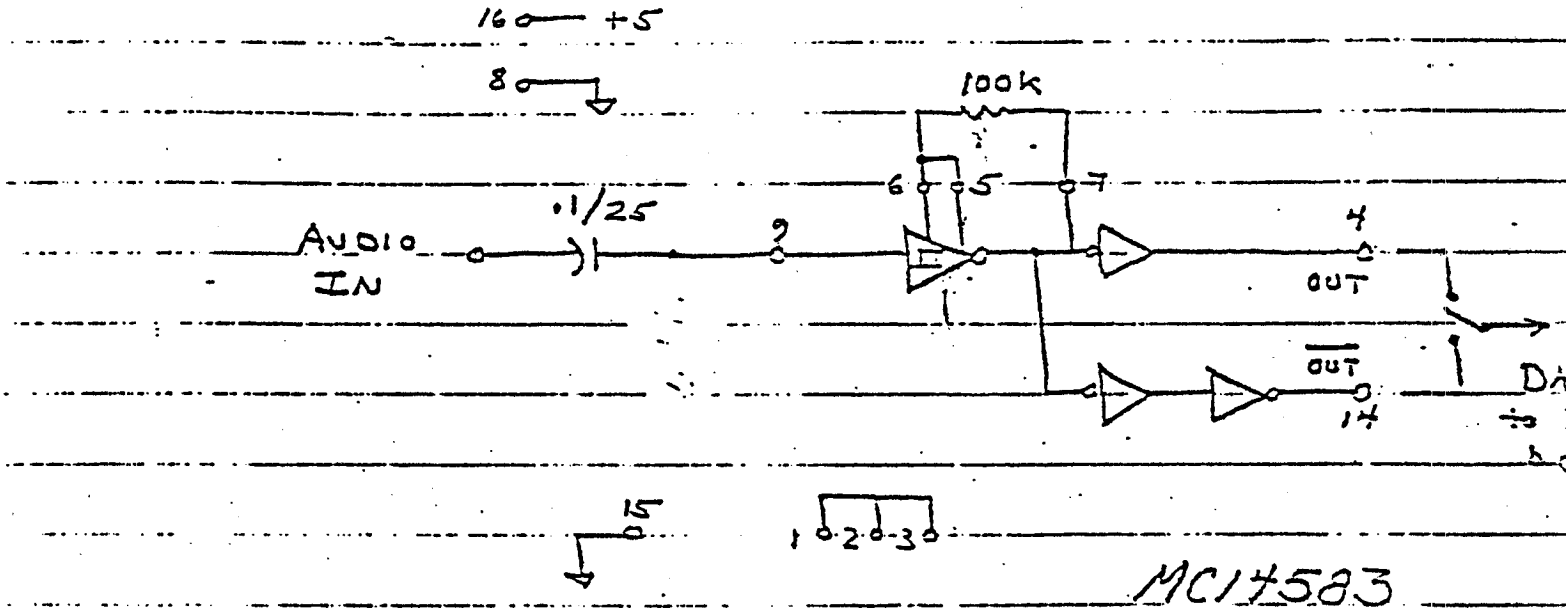
   This error has been corrected. For the moment, with this sample lot, we suggest to connect CP1 to ground and to operate handshake with CP2 only.

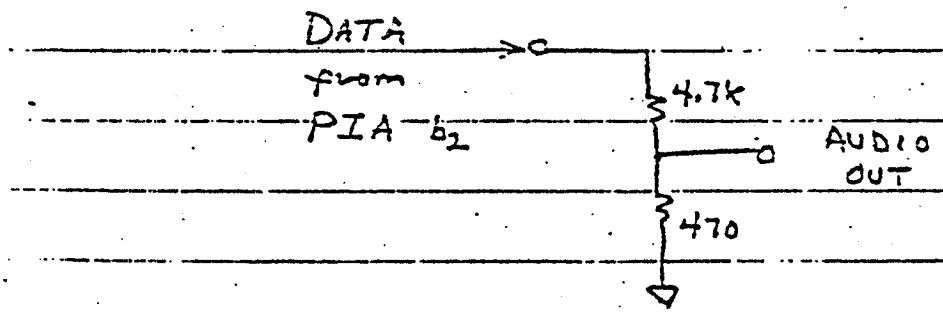
6. The rest of the peripheral interface port passes functional test at 1MHz clock rate.

```
)0001
)0002                                NAM    MIKBUG
)0003                                TTL    2.0 WITH AUDIO CASSETTE
)0004                          *     REV 0
)0005                          *     COPYRIGHT (C) 1977 BY MOTOROLA INC.
)0006                          *
)0007                          *     MIKBUG (TM) MOTOROLA
)0008                          *
)0009                          *     AUSTIN, TEXAS
)0010                          *     MICROCOMPUTER CAPITAL OF THE WORLD!
)0011                          *
)0012                          *     L  LOAD
)0013                          *     M  MEMORY CHANGE
)0014                          *     P  PRINT/PUNCH DUMP
)0015                          *     R  DISPLAY CONTENTS OF TARGET STACK
)0016                          *          CC   B   A   X   P   S
)0017                          *     S  1,SET SPEED FOR 10 CPS
)0018                          *        3,SET SPEED FOR 30 CPS
)0019                          *     B  PRINT OUT ALL BREAKPOINTS
)0020                          *     C  CONTINUE EXECUTION FROM CURRENT LOCATION
)0021                          *     N  NEXT INSTRUCTION
)0022                          *     T  TRACE 'N' INSTRUCTIONS
)0023                          *     G  GO TO LOCATION 'N'
)0024                          *     D  DELETE ALL BREAKPOINTS
)0025                          *     U  RESET BREAKPOINT WITH ADDRESS 'N'
)0026                          *     V  SET A BREAKPOINT WITH ADDRESS 'N'
)0027                          *     E  EXORTAPE CASSETTE INTERFACE
)0028                          *
)0029                                 OPT    S,O,LLEN=80,CREF
)0030          8008  A ACIAS   EQU    $8008
)0031          8009  A ACIAD   EQU    $8009
)0032          003F  A SWI     EQU    $3F       SWI OP CODE
)0033                *
)0034A F800                           ORG    $F800
)0035          F800  A BASORG  EQU    *         BASE ORIGIN
)0036                *
)0037                *     I/O INTERRUPT SEQUENCE
)0038                *
)0039A F800 FE A000  A IO       LDX    IOV
)0040A F803 6E 00    A         JMP    X
)0041                *
)0042                *     NMI SEQUENCE
)0043                *
)0044A F805 FE A006  A POWDWN LDX    NIO       GET NMI VECTOR
)0045A F808 6E 00    A         JMP    X
)0046                *
)0047                *     SWI INTERRUPT SEQUENCE
)0048                *
)0049A F80A FE A00A  A SFEI   LDX    SWI1
)0050A F80D 6E 00    A         JMP    X
```

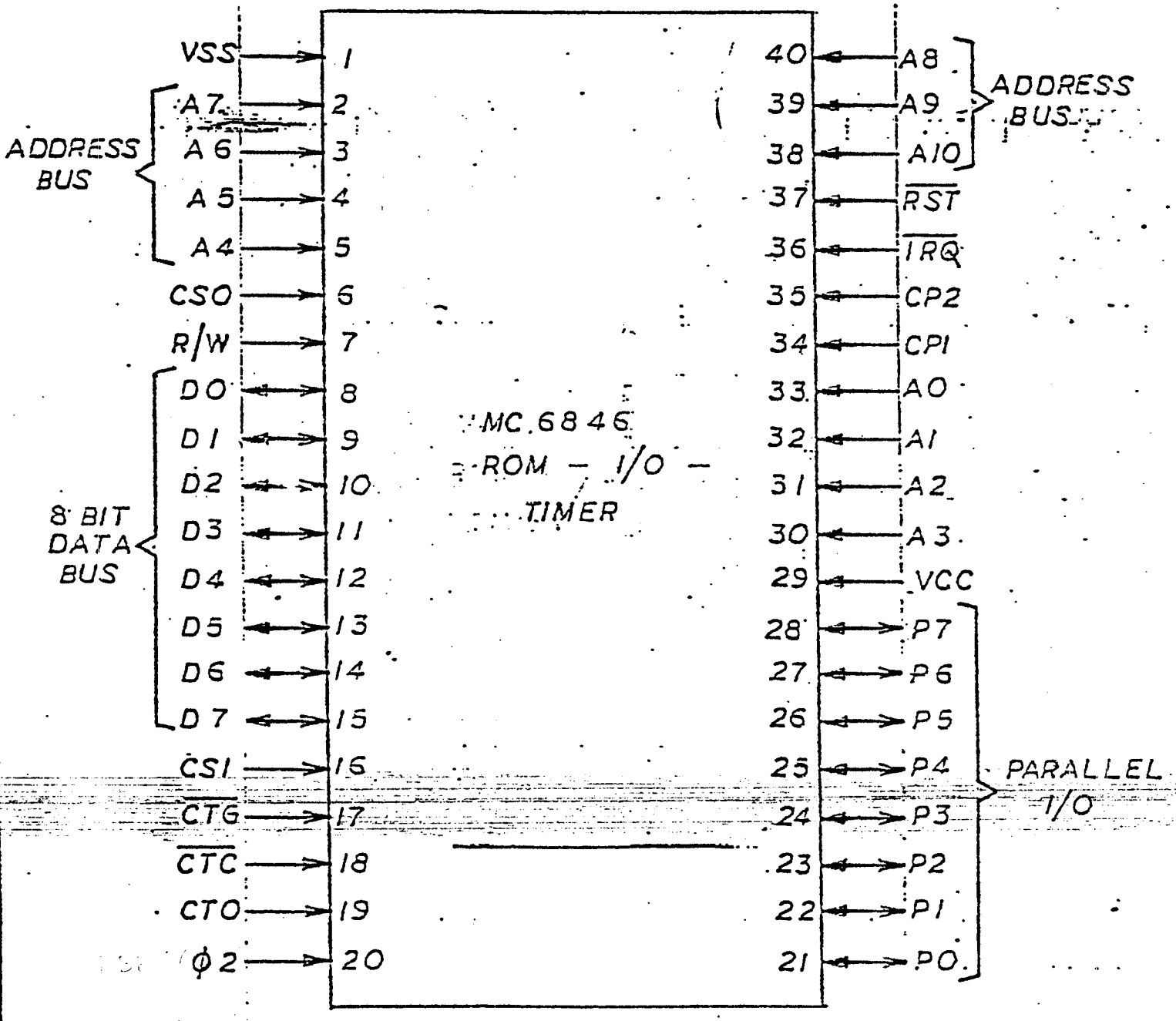
```
00052                    *
00053                    * JUMP TABLE TO ROUTINES PERFORMING MIKBUG FCTN'S
00054                    *
00055          F80F   A FCTABL EQU    *
00056A F80F    42     A        FCC    /B/       "B" - PRINT ALL BREAKS
00057A F810    FA15   A        FDB    PNTBRK
00058A F812    43     A        FCC    /C/       "C" - CONTINUE
00059A F813    FA54   A        FDB    CONT
00060A F815    44     A        FCC    /D/       "D" - DELETE ALL BREAKS
00061A F816    FA0B   A        FDB    DELBRK
00062A F818    47     A        FCC    /G/       "G" - GO TO ENTERED ADDRESS
00063A F819    FA25   A        FDB    GOTO
00064A F81B    4C     A        FCC    /L/       "L" - LOAD
00065A F81C    F8D0   A        FDB    LOAD
00066A F81E    4D     A        FCC    /M/       "M" - MEMORY CHANGE
00067A F81F    F91D   A        FDB    CHANGE
00068A F821    4E     A        FCC    /N/       "N" - NEXT (TRACE 1 INSTR)
00069A F822    FA37   A        FDB    NEXT
00070A F824    50     A        FCC    /P/       "P" - PUNCH
00071A F825    FB02   A        FDB    PUNCH
00072A F827    52     A        FCC    /R/       "R" - PRINT STACK
00073A F828    FA5C   A        FDB    PSTAK1
00074A F82A    53     A        FCC    /S/       "S" - CHANGE SPEED FOR TTY
00075A F82B    F9FE   A        FDB    SPD
00076A F82D    54     A        FCC    /T/       "T" - TRACE N INSTRUCTIONS
00077A F82E    FA50   A        FDB    TRACE
00078A F830    55     A        FCC    /U/       "U" - RESET A BREAKPOINT
00079A F831    FA11   A        FDB    RSTBRK
00080A F833    45     A        FCC    /E/       "E" - EXORTAPE CASSETT INTF "AC
00081A F834    FD05   A        FDB    EXORT
00083A F836    56     A        FCC    /V/       "V" - SET A BREAKPOINT
00083A F837    FA1D   A        FDB    SETBRK
00084          F839   A FCTBEN EQU    *
```

Complete Cassette Interface





ROM-I/O-TIMER


MC6846
ROM – I/O –
TIMER
VSS 1
A7 2
A6 3
A5 4
A4 5
CS0 6
R/W 7
D0 8
D1 9
D2 10
D3 11
D4 12
D5 13
D6 14
D7 15
CS1 16
CTG 17
CTC 18
CTO 19
φ2 20
40 A8
39 A9
38 A10
37 RST
36 IRQ
35 CP2
34 CP1
33 A0
32 A1
31 A2
30 A3
29 VCC
28 P7
27 P6
26 P5
25 P4
24 P3
23 P2
22 P1
21 P0
ADDRESS BUS
8 BIT DATA BUS
ADDRESS BUS
PARALLEL I/O

```
0086                    *
0087                    *       INITIALIZATION/RESET CODE
0088                    *
0089          F833  A ADRSTR EQU    *
0  0A F833    A078  A         FDB    STACK     INIT FOR "SP"
0091A F833B   F878  A         FDB    SWI1S     INIT FOR "SWI1"
0092A F83D    F8A0  A         FDB    BRKINH    INIT FOR "SWI2"
0093                    *
00944 F83F 20 03 F844           BRA    BRG       "BRA" INST IS REPLACED BY
0095A F841 7E F8DE  A           JMP    BRNOGO    COND BRA INST IN ROUT.
0096A F844 7E F8E2  A BRG       JMP    BRGO      WHICH DETERMINES IF
0097                    *                        BRA IS GO/NOGO
0098                    *
0099                    * BUILD ADDRESS
0100                    *
0101A F847 8D 0C F855 BADDR   BSR    BYTE      READ 2 FRAMES
0102A F849 B7 A03E  A         STAA   XHI
0103A F84C 8D 07 F855         BSR    BYTE
0104A F84E B7 A03F  A         STAA   XLOW
0105A F851 FE A03E  A         LDX    XHI       (X) ADDRESS WE BUILT
0106A F854 39                 RTS
0108                    * INPUT BYTE (TWO FRAMES)
0109A F855 8D 53 F8AA BYTE    BSR    INHEX     GET HEX CHAR
0110A F857 48         BYTE2   ASLA
0111A F858 48                 ASLA
0112A F859 48                 ASLA
0113A F85A 48                 ASLA
0  4A F85B 16                 TAB
0  5A F85C 8D 4C F8AA         BSR    INHEX
0116A F85E 1B                 ABA
0117A F85F 16                 TAB
0118A F860 FB A03C  A         ADDB   CKSM
0119A F863 F7 A03C  A         STAB   CKSM
0120A F866 39                 RTS
0122A F867 44           OUTHL   LSRA             OUT HEX LEFT BCD DIGIT
0123A F868 44                 LSRA
0124A F869 44                 LSRA
0125A F86A 44                 LSRA
0128A F86B 84 0F    A OUTHR   ANDA   #$F       OUT HEX RIGHT BCD DIGIT
0129A F86D 8B 30    A         ADDA   #$30
0130A F86F 81 39    A         CMPA   #$39
0131A F871 23 02 F875         BLS    OUTCH
0132A F873 8B 07    A         ADDA   #$7
0134                    * OUTPUT ONE CHAR
0135A F875 7E F959  A OUTCH   JMP    OUTCH1
0136A F878 7E F966  A INCH    JMP    INCH1
0137                    * PRINT DATA POINTED AT BY X-REG
0138A F87B 8D F8 F875 PDATA2  BSR    OUTCH
0139A F87D 08                 INX
0140A F87E A6 00    A PDATA1  LDAA   X
0141A F880 81 04    A         CMPA   #4
0  2A F882 26 F7 F87B         BNE    PDATA2
0143A F884 39                 RTS              STOP ON EOT
0144                    *
0145                    *  INPUT ADDRESS
0146                    *
0147A F885 8D 17 F89E GETADD  BSR    PCRLF     PRINT CR LF
0148A F887 CE FCAE  A         LDX    #MCL4
```

```
00149A F88A 8D F2 F87E        BSR     PDATA1   ASK FOR BEGADDR
00150A F88C 8D B9 F847        BSR     BADDR    GET BEG ADDR
00151A F88E FF A002  A        STX     BEGA
00152A F891 8D 0B F89E        BSR     PCRLF    PRINT CR LF
00153A F893 CE FCB9  A        LDX     #MCL5
00154A F896 8D E6 F87E        BSR     PDATA1   ASK FOR END ADDR
00155A F898 8D AD F847        BSR     BADDR    GET END ADDR
00156A F89A FF A004  A        STX     ENDA
00157A F89D 39                RTS              *
00158                 * PRINT CR LF
00159                 *
00160A F89E FF A03E  A PCRLF  STX     XHI      SAVE XR
00161A F8A1 CE FC86  A        LDX     #MCL1
00162A F8A4 8D D8 F87E        BSR     PDATA1   PRINT CRLF
00163A F8A6 FE A03E  A        LDX     XHI
00164A F8A9 39                RTS
00165
00166                 *
00167A F8AA 8D CC F878 INHEX  BSR     INCH
00168A F8AC 80 30     A INHEX2 SUBA   #$30
00169A F8AE 2B 6A F91A        BMI     C1       NOT HEX
00170A F8B0 81 09     A       CMPA    #$09
00171A F8B2 2F 0A F8BE        BLE     IN1HG
00172A F8B4 81 11     A       CMPA    #$11
00173A F8B6 2B 62 F91A        BMI     C1       NOT HEX
00174A F8B8 81 16     A       CMPA    #$16
00175A F8BA 2E 5E F91A        BGT     C1       NOT HEX
00176A F8BC 80 07     A       SUBA    #7
00177A F8BE 39          IN1HG RTS
00178                 *
00179A F8BF A6 00     A OUT2H  LDAA   0,X      OUTPUT 2 HEX CHAR
00180A F8C1 8D A4 F867 OUT2HA BSR     OUTHL    OUT LEFT HEX CHAR
00181A F8C3 A6 00     A       LDAA    0,X      PICK UP BYTE AGAIN
00182A F8C5 08                INX
00183A F8C6 20 A3 F86B        BRA     OUTHR    OUTPUT RIGHT HEX CHAR AND RTS
00185A F8C8 8D F5 F8BF OUT4HS BSR     OUT2H    OUTPUT 4 HEX CHAR + SPACE
00186A F8CA 8D F3 F8BF OUT2HS BSR     OUT2H    OUTPUT 2 HEX CHAR + SPACE
00187A F8CC 86 20     A OUTS   LDAA   #$20     SPACE
00188A F8CE 20 A5 F875        BRA     OUTCH    (BSR & RTS)
00189           F8D0  A LOAD   EQU     *
00190A F8D0 86 11     A       LDAA    #@21
00191A F8D2 8D A1 F875        BSR     OUTCH    OUTPUT CHAR
00192                 *
00193                 * TURN READER RELAY ON
00194                 *
00195A F8D4 B6 A044  A        LDAA    ACIAT    GET ACIA CONTROL REG FORMAT
00196A F8D7 8A 40     A       ORAA    #$40     SET RTS TO TURN RDR RELAY ON
00197A F8D9 B7 8008  A        STAA    ACIAS    TURN IT ON
00198                 *
00199A F8DC 7C A016  A        INC     OUTSW    DO NOT ECHO TAPE
00200                 *
00201A F8DF 8D 97 F878 LOAD3  BSR     INCH
00202A F8E1 20 03 F8E6        BRA     LOAD4
00203A F8E3 7E F9A4  A ENTER  JMP     ENT1     MIKBUG 1 ENTRY POINT
00204           F8E6  A LOAD4  EQU     *
00205A F8E6 81 53     A       CMPA    #'S
00206A F8E8 26 F5 F8DF        BNE     LOAD3    1ST CHAR NOT (S)
00207A F8EA 8D 8C F878        BSR     INCH     READ CHAR
```

```
                        CMPA    #'3
0208A F8EC 81 33     A  BEQ     C1
0208A F8EE 27 2A F91A   CMPA    #'1
0210A F8F0 81 31     A  BNE     LOAD3     2ND CHAR NOT (1)
 .11A F8F2 26 E8 F8DF   CLR     CKSM      ZERO CHECKSUM
0212A F8F4 7F A03C   A  JSR     BYTE      READ BYTE
0213A F8F7 BD F855   A  SUBA    #2
0214A F8FA 80 02     A  STAA    BYTECT    BYTE COUNT
0215A F8FC B7 A03D   A* BUILD ADDRESS
0216                    JSR     BADDR
0217A F8FF BD F847   A* STORE DATA
0218                    JSR     BYTE
0219A F902 BD F855   A LOAD11   DEC     BYTECT
0220A F905 7A A03D   A  BEQ     LOAD15    ZERO BYTE COUNT
0221A F908 27 03 F913   STAA    X         STORE DATA
0222A F90A A7 00     A  CMPA    0,X       CHECK DATA
0223A F90C A1 00     A  BNE     LOAD19    DATA NOT STORED
0224A F90E 26 06 F916   INX
0225A F910 08           BRA     LOAD11
0226A F911 20 EF F902
0227                    *
0228                    * DOES CHECKSUM CHECK?
0229                    *
0230A F913 5C           LOAD15  INCB
0231A F914 27 C9 F8DF   BEQ     LOAD3
0232A F916 86 3F     A LOAD19   LDAA    #'?       PRINT QUESTION MARK
0233A F918 8D 3F F959   BSR     OUTCH1
0234A F91A 7E F8BA   A C1      JMP     CONTRL
0235                    *
0237                    * CHANGE MEMORY (M AAAA DD NN)
0238                    *
0233A F91D BD F847   A CHANGE   JSR     BADDR     BUILD ADDRESS
0240A F920 8D AA F8CC   BSR     OUTS      OUTPUT SPACE
0241A F922 FE A03E   A CHANG   LDX     XHI
0242A F925 8D A3 F8CA   BSR     OUT2HS    PRINT DATA OLD
0243A F927 09           DEX
0244A F928 8D 3C F966 CHA1  BSR     INCH1     INPUT CHAR
0245A F92A 81 0A     A  CMPA    #$0A      CHECK FOR LINE FEED
0246A F92C 27 12 F940   BEQ     LF
0247A F92E 81 5E     A  CMPA    #$5E      CHECK FOR ^
0248A F930 27 11 F943   BEQ     UA
0249A F932 BD F8AC   A  JSR     INHEX2    S BSR BYTE
0250A F935 BD F857   A  JSR     BYTE2     GET NEW BYTE
0251A F938 A7 00     A  STAA    X         CHANGE MEMORY
0252A F93A A1 00     A  CMPA    X
0253A F93C 26 D8 F916   BNE     LOAD19    NO CHNAGE
0254A F93E 20 E8 F928   BRA     CHA1
0255A F940 08       LF  INX               INC ADDR
0256A F941 20 05 F948   BRA     UA1
0257A F943 86 0A     A UA       LDAA    #$0A
0258A F945 8D 12 F959   BSR     OUTCH1    OUTPUT LF
0259A F947 09           DEX               DEC ADDR
0260A F948 FF A03E   A UA1  STX     XHI       SAV DATA ADDR
0261A F94B CE FC7E   A      LDX     #MCL+1
0262A F94E BD F87E   A      JSR     PDATA1    PRINT CR
0263A F951 CE A03E   A      LDX     #XHI
0264A F954 BD F8C8   A      JSR     OUT4HS    OUTPUT DATA ADDR
0265A F957 20 C9 F922   BRA     CHANG
                        *
0266
```

```
00267A F953 37          OUTCH1 PSHB            SAVE BREG
00268A F95A F6 8008  A OUTC1  LDAB   ACIAS
00269A F95D 57                ASRB
00270A F95E 57                ASRB
00271A F95F 24 F9 F95A        BCC    OUTC1     XMIT NOT READY
00272A F961 B7 8009  A        STAA   ACIAD     OUTPUT CHAR
00273A F964 33                PULB
00274A F965 39                RTS
00275                   *
00276                   * INPUT ONE CHAR TO AREG
00277A F966 B6 8008  A INCH1  LDAA   ACIAS
00278A F969 47                ASRA
00279A F96A 24 FA F966        BCC    INCH1     RECEIVER NOT READY
00280A F96C B6 8009  A        LDAA   ACIAD     INPUT CHAR
00281A F96F 84 7F    A        ANDA   #$7F      RESET PARITY BIT
00282A F971 81 7F    A        CMPA   #$7F
00283A F973 27 F1 F966        BEQ    INCH1     RUBOUT;DEL
00284A F975 7D A018  A        TST    OUTSW     SHOULD INPUT BE ECHOED?
00285A F978 27 DF F959        BEQ    OUTCH1    IF SO, OUTPUT THE CHAR
00286A F97A 39                RTS              ELSE, RETURN TO CALLER OF INCH
00287                   *
00288                   * CONSTANT INITIALIZATION
00289                   * S = POINTER TO ROM BYTES TO BE COPIED TO RAM
00290                   * X = POINTER TO RAM BYTES TO BE INITIALIZED
00291                   *
00292           F97B  A START  EQU    *          ACTUAL CODE START
00293A F97B 8E F838  A        LDS    #ADRSTR-1 START OF CONSTANT DATA
00294A F97E CE A008  A        LDX    #SP        START OF RAM AREA
00295                   *
00296A F981 32          INILP1 PULA              GET NEXT CONSTANT BYTE
00297A F982 A7 00    A        STAA   0,X        INIT NEXT RAM BYTE
00298A F984 08                INX              UPDATE POINTER
00299A F985 8C A016  A        CPX    #BRANEN    END OF CONSTANT RAM AREA?
00300A F988 26 F7 F981        BNE    INILP1     NO, CONTINUE INITIALIZATION
00301                   *
00302                   * INITIALIZATION TO 0
00303                   * X HOLDS INDEX OF 1ST BYTE TO BE SET TO 0
00304                   *
00305A F98A 6F 00    A INILP2 CLR    0,X        CLEAR NEXT BYTE OF RAM
00306A F98C 08                INX              UPDATE INDEX
00307A F98D 8C A080  A        CPX    #ENDIN0    ANY MORE BYTES TO INIT?
00308A F990 26 F8 F98A        BNE    INILP2     NO, CONTINUE CLEARING
00309                   *
00310                   * SET CC SO WHEN WE 'GO' TO USER PGM THE
00311                   *   INTERUPT MASK IS SET
00312                   *
00313A F992 86 D0    A        LDAA   #$D0
00314A F994 B7 A079  A        STAA   $A079      PUT IN STACK TO BE PULLED
00315                   *
00316                   * INITIALIZE ACIA
00317                   *
00318A F997 86 03    A        LDAA   #3         MASTER RESET CODE
00319A F999 B7 8008  A        STAA   ACIAS      RESET ACIA
00320                   *
00321A F99C 86 11    A INZ    LDAA   #%00010001 CHAR LEN=8; NO PARITY
00322                   *                              2 STOP BITS
00323                   *
00324A F99E B7 A044  A INZ1   STAA   ACIAT      SAVE FOR CONTROL LOOP ACIA IN
```

```
00325                          *
00325A F9A1 B7 8008   A        STAA   ACIAS    INZ ACIA
00327A F9A4 BE A008   A ENT1   LDS    SP
00[illegible] F9A7 BD F89E   A        JSR    PCRLF
00[illegible] F9AA 20 02 F9AE         BRA    CONTB    SKIP TO INSURE MIKBUG 1 COMPAT.
00330A F9AC 20 B8 F966 INCH2  BRA    INCH1    MIKBUG 1.0 INPUT 1 CHARACTER
00331A F9AE CE FC8D   A CONTB  LDX    #MCLE    PRINT HEADER
00332A F9B1 BD F87E   A        JSR    PDATA1   PRINT DATA STRING
00333A F9B4 CE F9EB   A        LDX    #NMI     INIT PDN
00334A F9B7 FF A006   A        STX    NIO
```

```
0336                    *
0337                    * MAIN COMMAND/CONTROL LOOP
0338                    *
0339          F9BA  A CONTRL EQU    *
0340                    *
0341                    * RESTORE STACK POINTER REGISTER
0342                    *
0343A F9BA BE A008  A         LDS    SP        SP WAS INITIALIZED EARLIER
0344                    *
0345A F9BD B6 A044  A         LDAA   ACIAT     GET PROPER ACIA INIT BITS
                                                FOR USER'S TERMINAL
0346                    *
0347A F9C0 B7 8008  A         STAA   ACIAS     INZ ACIA
0348                    *
0349A F9C3 7F A015  A         CLR    OUTSW     MAKE SURE INPUT IS ECHOED
0350                    *
0351A F9C6 CE FC7C  A         LDX    #MCLOFF   TERMINAL INIT STRING
0352A F9C9 BD F87E  A         JSR    PDATA1    PRINT DATA STRING
0353                    *
0354A F9CC 8D 38 F966         BSR    INCH1     READ COMMAND CHARACTER
0355A F9CE 16                 TAB              SAVE CHARACTER IN B
0356A F9CF 20 02 F9D3         BRA    CNTA      SKIP OVER MIKBUG 1.0 VECTOR
0357A F9D1 20 85 F958 OUT2    BRA    OUTCH1    MIKBUG 1.0 ; OUTPUT 1 CHAR ROUT
0358A F9D3 BD F8CC  A CNTA    JSR    OUTS      PRINT SPACE AFTER COMMAND
0359                    *
0360                    * B REGISTER HOLDS CHARACTER INPUT BY USER.
0361                    * USE JUMP TABLE TO GO TO APPROPRIATE ROUTINE.
0362                    *
0363A F9D5 CE F80F  A         LDX    #FCTABL   X:= ADDRESS OF JUMP TABLE
0364A F9D8 E1 00    A NXTCHR  CMPB   0,X       DOES INPUT CHAR MATCH?
0365A F9DB 27 0A F9E7         BEQ    GOODCH    YES, GOTO APPROPRIATE ROUTINE
0366A F9DD 08                 INX              ELSE, UPDATE INDEX INTO TABLE
0367A F9DE 08                 INX
0368A F9DF 08                 INX
0369A F9E0 8C F839  A         CPX    #FCTBEN   END OF TABLE REACHED?
0370A F9E3 26 F4 F9D9         BNE    NXTCHR    NO, TRY NEXT CHAR
0371A F9E5 20 D3 F9BA         BRA    CONTRL    NO MATCH, REPROMPT USER
0372                    *
0373                    *
0374A F9E7 EE 01    A GOODCH  LDX    1,X       GET ADDRESS FROM J.T.
0375A F9E9 6E 00    A         JMP    0,X       GOTO APPROPRIATE ROUTINE
0376                    *
0377                    *
0378                    *
0379                    * NMI ENTRY
0380                    *
0381A F9EB BF A008  A NMI     STS    SP        SAVE STACK
0382A F9EE BD F89E  A         JSR    PCRLF
0383A F9F1 86 42    A         LDAA   #'B       PRINT B
0384A F9F3 8D DC F9D1         BSR    OUT2
0385A F9F5 BD F8CC  A         JSR    OUTS
0386A F9F8 86 02    A         LDAA   #2        REMOVE BREAKPOINTS
0387A F9FA 8D 6E FA6A         BSR    BRKSUB
0388A F9FC 20 5E FA5C         BRA    PSTAK1
0389                    *
0390                    * SET SPEED FOR USER TTY
0391                    *
0392A F9FE 8D AC F9AC SPD     BSR    INCH2     INPUT CHAR
0393A FA00 81 31    A         CMPA   #'1
```

```
)0394A FA02 27 38 F93C          BEQ     INZ
)0395A FA04 86 15    A          LDAA    #$15
)0396A FA06 20 96 F95E          BRA .   INZ1        SET 2 STOP BITS
)0398                           *
)03                             *
)0400A FA08 7E F847  A BADDRJ   JMP     BADDR       GO BUILD ADDRESS
)0401                           *
)0402                           *
)0403                           * RESET ALL BREAKPOINTS
)0404                           *
)0405A FA0B 86 01    A DELBRK   LDAA    #1          RESET BREAKS FLAG
)0406A FA0D 8D 5B FA5A BSRBRK   BSR     BRKSUB      BREAK HANDLING SUBR.
)0407A FA0F 20 56 FA67          BRA     CNTRL2      RETURN TO COMMAND LEVEL
)0408                           *
)0409                           * RESET 1 BREAKPOINT
)0410                           *
)0411A FA11 8D F5 FA08 RSTBRK   BSR     BADDRJ      PUTS USER ENTERED ADDRESS
)0412                           *                       INTO XHI,XLOW
)0413A FA13 4F                  CLRA                RESET 1 BREAK FLAG
)0414A FA14 20 F7 FA0D          BRA     BSRBRK      GO RESET 1
)0415                           *
)0416                           * PRINT OUT ALL NON-ZERO BREAK ADDRESSES
)0417                           *
)0418A FA16 BD F89E  A PNTBRK   JSR     PCRLF       DO CR/LF
)0419A FA19 86 02    A          LDAA    #2          PRINT BREAK ADDRESSES FLAGS
)0420A FA1B 20 F0 FA0D          BRA     BSRBRK      GO PRINT
)0421                           *
)042?                           * SET ONE BREAK
)04..                           *
)04244 FA1D 8D E9 FA08 SETBRK   BSR     BADDRJ      GET USER ENTERED ADDRESS (XHI,X
)0425A FA1F 86 04    A          LDAA    #4          SET ONE BREAK FLAG
)0426A FA21 8D 47 FA6A          BSR     BRKSUB      GO SET IT
)0427A FA23 20 F1 FA16          BRA     PNTBRK      PRINT ALL BREAKPOINTS
```

```
00429                    *
00430                    * GO TO REQUESTED
00431                    *
00432A FA25 8D E1 FA08 GOTO   BSR    BADDRJ   GO GET ADDRESS FROM USER
00433                    *                     XHI,XLOW HOLD ADDRESS
00434A FA27 86 FF    A         LDAA   #$FF     FLAG FOR PUTTING IN BREAKS
00435A FA29 8D 3F FA6A         BSR    BRKSUB   GO PUT IN BREAKS
00436A FA2B 30                 TSX
00437A FA2C B6 A03E  A         LDAA   XHI      SAVE PCH ON STACK
00438A FA2F A7 05    A         STAA   5,X
00439A FA31 B6 A03F  A         LDAA   XLOW     PSH PCL
00440A FA34 A7 06    A         STAA   6,X
00441A FA36 3B                 RTI             GO TO USER PRG
00442                    *
00443                    * SINGLE INSTRUCTION TRACE REQUESTED
00444                    *
00445A FA37 CE 0001  A NEXT   LDX    #1        # INSTRUCTIONS TO TRACE
00446A FA3A 7F A043  A TRACE2 CLR    BRKTRC    CLEAR FLAG INDICATING TRACE
00447                    *                     IS DUE TO BREAK
00448A FA3D FF A01A  A TRACE3 STX    NTRACE    SAVE # INST'S TO TRACE
00449                    *                           IS DUE TO BREAK
00450A FA40 FE A008  A         LDX    SP        X : = STACK POINTER
00451A FA43 EE 06    A         LDX    6,X       X : = ADDRESS OF INSTR TO BE EX
00452A FA45 FF A017  A         STX    TRCADR    SAVE IN TRACE ADDRESS STORE
00453A FA48 A6 00    A         LDAA   0,X       GET INSTRUCTION TO BE TRACED
00454A FA4A B7 A019  A         STAA   TRCINS    SAVE IN TRACE INSTRUCTION STORE
00455A FA4D 7E FBCB  A         JMP    CONTRC    GO TO CONTINUE TRACE PART OF  P
00456                    *
00457                    * MULTIPLE INSTRUCTION TRACE
00458                    *
00459A FA50 8D B6 FA08 TRACE  BSR    BADDRJ    GET # OF INSTRUCTIONS TO TRACE
00460A FA52 20 E3 FA3A         BRA    TRACE2    GO TRACE'M
00461                    *
00462                    * CONTINUE EXECUTION
00463                    *
00464A FA54 7C A043  A CONT   INC    BRKTRC    TRACE 1 TO RESTORE SWI'S
00465A FA57 CE 0001  A         LDX    #1        ONE TRACE ONLY
00466A FA5A 20 E1 FA3D         BRA    TRACE3
00467                    *
00468                    *
00469                    *  R COMMAND
00470                    *
00471                    * PRINT STACK CONTENTS
00472                    *
00473A FA5C BD F89E  A PSTAK1 JSR    PCRLF     PRINT CR LF
00474A FA5F CE FC98  A         LDX    #MCL3     PRINT HEADER
00475A FA62 BD F87E  A         JSR    PDATA1
00476A FA65 8D 7B FAE2 PSTAK  BSR    PRINT     PRINT STACK
00477A FA67 7E F93A  A CNTRLZ JMP    CONTRL    RETURN TO COMMAND LEVEL
```

```
0479                    **********
0480                    *
0481                    * BRKSUB
0482                    *
0483                    *
0484                    * THIS ROUTINE DOES A NUMBER OF OPERATIONS HAVING
0485                    * TO DO WITH BREAKPOINTS.
0486                    *
0487                    * THE A REGISTER DETERMINES FUNCTION PERFORMED:
0488                    *
0489                    * A = -1 => BREAKS ARE PUT INTO USER'S CODE
0490                    * A =  0 => THE BREAKPOINT WHOSE ADDRESS IS IN
0491                    *           XHI, XLOW IS PURGED;
0492                    *           ALL BREAKPOINTS ARE TEMPORARILY REMOVED
0493                    * A =  1 => ALL BREAKPOINTS ARE PURGED
0494                    * A =  2 => ALL BREAKPOINTS ARE PRINTED OUT
0495                    *           ALL BREAKPOINTS ARE TEMPORARILY REMOVED
0496                    * A =  3 => ALL BREAKPOINTS ARE TEMPORARILY REMOVED
0497                    * A =  4 => THE BREAK ADDRESS IN XHI, XLOW IS
0498                    *           PUT INTO THE FIRST ZERO BREAKPOINT
0499                    *           POSITION; ALL BREAKS ARE TEMPORARILY REMOV
0500                    *
0501                    **********
0502                    *
0503          FA6A  A BRKSUB EQU   *
0504A FA6A BF A040  A        STS   SSAVE    SAVE S SO WE CAN USE
0505A FA6D B7 A03C  A        STAA  ASAVE    A HOLDS THE FUNCTION #
05                      *
0507A FA70 CE A01C  A        LDX   #BRKADR  INIT X FOR LOOP THROUGH BREAKS
0508                    *
0509                    * START OF LOOP THROUGH BREAK ADDRESSES
0510                    *
0511A FA73 B6 A03C  A BRKLP  LDAA  ASAVE    GET FUNCTION #
0512A FA76 AE 00    A        LDS   0,X      S:=NEXT ADDRESS IN BRKPT LIST
0513A FA78 27 2D FAA7        BEQ   LN       IF 0, THEN NOT A VALID BREAK
0514                    *
0515A FA7A 7D A042  A        TST   BRKSIN   ARE BREAKS IN USER'S CODE?
0516A FA7D 27 36 FAB5        BEQ   NOBRIN   BRANCH, IF NOT
0517                    *
0518                    * BREAKS ARE IN USER'S CODE
0519                    *
0520A FA7F 4D                TSTA           SHOULD BREAKS BE IN?
0521A FA80 2B 21 FAA3        BMI   BKDONE   YES, RETURN TO CALLER
0522                    *
0523                    * BREAKS ARE TO BE TAKEN OUT OF USER'S
0524                    * CODE TEMPORARILY
0525                    *
0525A FA82 A6 10    A BRK2   LDAA  2*NBRBPT,X GET INSTR. BELONG-
0527                    *                      ING IN USER CODE
0528A FA84 36                PSHA           PUT IT THERE
0529                    *
05                      * OTHER ACTIONS TO BE PERFORMED EACH TIME THROUGH
0531                    * LOOP WHEN BREAK ADDRESS NOT EQUAL TO 0.
0532                    *
0533A FA85 B6 A03C  A BKCON1 LDAA  ASAVE    WHAT FUNCTION IS TO BE DONE
0534A FA88 27 37 FAC1        BEQ   FNDRPL   SEE IF BREAKPOINT NEEDS TO
0535                    *                      BE REPLACED
0536A FA8A 81 01    A        CMPA  #1       IS BRK ADDRESS TO BE RESET?
```

```
00537A FA8C 27 41 FACF          BEQ     CLRBRK   YES, SET BRKADR TO 0
00538                    *
00539A FA8E 81 02    A          CMPA    #2       IS BRK ADDR TO BE PRINTED?
00540A FA90 27 49 FADB          BEQ     PRNTBK   YES, GO PRINT ADDRESS
00541                    *
00542                    * UPDATE LOOP INDEX AND LOOP IF APPROPRIATE
00543                    *
00544A FA92 08           BKCON2 INX              MAKE X POINT TO
00545A FA93 08                  INX              NEXT BREAK ADDRESS
00546A FA94 8C A02C  A  BKCON3 CPX     #BRKINS  ANY MORE BREAKS?
00547A FA97 26 DA FA73          BNE     BRKLP    YES,LOOP
00548                    *
00549                    * WRAP-UP PROCESSING AND EXIT
00550                    *
00551A FA99 4F                  CLRA             A = BREAKS IN FLAG
00552A FA9A 7D A03C  A          TST     ASAVE    IS FUNCTION = -1?
00553A FA9D 2A 01 FAA0          BPL     BKPUT    NO, SO BRKSIN = 0
00554A FA9F 4C                  INCA             FCTN = -1 => BRKSIN:=1
00555A FAA0 B7 A042  A  BKPUT  STAA    BRKSIN   STORE APPROPRIATE FLAG
00556                    *
00557                    * RESTORE S-REG AND RETURN TO CALLER
00558                    *
00559A FAA3 BE A040  A  BKDONE LDS      SSAVE    RESTORE USER S-REG
00560A FAA6 39                  RTS              RETURN
00561                    *
00562                    *
00563                    * MISCELLANEOUS ROUTINES FOR BRKSUB
00564                    *
00565                    * BREAKPOINT ADDRESS = 0 - IF FUNCTION = 4 THEN
00566                    * PUT BREAKPOINT ADDRESS IN CURRENT POSITION
00567                    * A HOLDS THE FUNCTION #, X HOLDS BREAKPOINT INDEX
00568                    *
00569A FAA7 81 04    A  LN       CMPA    #4       IS FUNCTION = 4
00570A FAA9 26 E7 FA92          BNE     BKCON2   IF NOT, THEN CONTINUE LOOP
00571                    *
00572A FAAB BE A03E  A          LDS     XHI      GET NEW BREAK ADDRESS
00573A FAAE AF 00    A          STS     0,X      PUT IN CURRENT POSITION
00574                    *
00575A FAB0 7A A03C  A          DEC     ASAVE    DO NOT PLACE ADDRESS MORE
00576                    *                            THAN ONCE-CONT TO
00577                    *                            TAKE OUT BREAKPOINTS.
00578A FAB3 20 DD FA92          BRA     BKCON2   CONTINUE LOOP
00579                    *
00580                    * BREAKS ARE NOT IN AND ADDRESS IS NON-ZERO.
00581                    * IF FUNCTION = -1 THEN SWI'S ARE TO BE PUT IN.
00582                    * A HOLDS FUNCTION NUMBER, S HOLDS ADDRESS
00583                    *
00584A FAB5 4D           NOBRIN TSTA             IS FUNCTION = -1
00585A FAB6 2A CD FA85          BPL     BKCON1   NO,CONTINUE
00586                    *
00587A FAB8 34                  DES              MAKE ADDRESS POINT TO 1 LESS
00588A FAB9 32                  PULA             GET USER INSTRUCTION
00589A FABA A7 10    A          STAA    2*NBRBPT,X SAVE
00590A FABC 86 3F    A          LDAA    #SWI     GET SWI OP CODE
00591A FABE 36                  PSHA             REPLACE USER INSTRUCTION
00592A FABF 20 D1 FA92          BRA     BKCON2   CONTINUE LOOP
00593                    *
00594                    * FUNCTION=0, BRK ADDR NOT = 0, USER'S INSTR
```

```
00595                    * IS IN (NOT SWI).
00596                    * IF ADDRESS = XHI,XLO THEN SET ADDRESS = 0
00597                    *
00598A FAC1 A6 00     A FNDRPL LDAA   0,X       GET TOP BYTE OF ADDRESS
00599A FAC3 B1 A03E   A        CMPA   XHI       DO TOP BYTES COMPARE
00600A FAC6 26 CA FA92         BNE    BKCON2    NO,CONTINUE LOOP
00601A FAC8 E6 01     A        LDAB   1,X       GET LOW BYTE OF ADDR
00602A FACA F1 A03F   A        CMPB   XLOW      SAME FOR LOW BYTES
00603A FACD 26 C3 FA92         BNE    BKCON2
00604                    *
00605A FACF 6F 00     A CLRBRK CLR    0,X       CLEAR OUT BREAK
00606A FAD1 6F 01     A        CLR    1,X       ADDRESS FIELD
00607A FAD3 20 BD FA92         BRA    BKCON2    CONTINUE LOOP
00608                    *
00609                    *
00610A FAD5 7E F8CA   A OT2HS  JMP    OUT2HS
00611A FAD8 7E F8C8   A OT4HS  JMP    OUT4HS
00612                    *
00613                    *
00614                    * PRINT OUT BREAK ADDRESS
00615                    * FUNCTION = 2, BREAK ADDRESS NOT = 0, X = ADDRESS IND
00616                    *
00617A FADB BE A040   A PRNTBK LDS    SSAVE
00618A FADE 8D F8 FAD8         BSR    OT4HS     OUTPUT ADDRESS AND SPACE
00619A FAE0 20 B2 FA94         BRA    BKCON3    OUT4HS INCREMENTS X,
00620                    *                       SO BYPAS 2 INX'S
```

```
00622                          *
00623                          * PRINT CONTENTS OF STACK
00624                          *
00625A FAE2 BD F89E   A PRINT  JSR    PCRLF     PRINT CR LF
00626A FAE5 FE A008   A        LDX    SP        PRINT OUT STACK
00627A FAE8 08                 INX
00628A FAE9 8D EA FAD5         BSR    OT2HS     CONDITION CODES
00629A FAEB 8D E8 FAD5         BSR    OT2HS     ACC-B
00630A FAED 8D E6 FAD5         BSR    OT2HS     ACC-A
00631A FAEF 8D E7 FAD8         BSR    OT4HS     X-REG
00632A FAF1 8D E5 FAD8         BSR    OT4HS     P-COUNTER
00633A FAF3 CE A008   A        LDX    #SP
00634A FAF6 8D E0 FAD8         BSR    OT4HS     STACK POINTER
00635A FAF8 39                 RTS
00638                          *      PUNCH DUMP
00639                          *      PUNCH FROM BEGINING ADDRESS (BEGA) THRU ENDING
00640                          *      ADDRESS (ENDA)
00641                          *
00643A FAF9     0D     A MTAPE1 FCB    $D,$A,0,0,0,0,'S,'1,4 PUNCH FORMAT
     A FAFA     0A     A
     A FAFB     00     A
     A FAFC     00     A
     A FAFD     00     A
     A FAFE     00     A
     A FAFF     53     A
     A FB00     31     A
     A FB01     04     A
00645          FB02   A PUNCH  EQU    *
00647A FB02 BD F885   A        JSR    GETADD    GET ADDRESS
00648A FB05 86 12     A        LDAA   #$12      TURN TTY PUNCH ON
00649A FB07 BD F875   A        JSR    OUTCH     OUT CHAR
00650                          *
00651                          *      PUNCH LEADER - 25 NULLS
00652                          *
00653A FB0A C6 19     A        LDAB   #25       B HOLDS # NULLS TO PUNCH
00654A FB0C 4F           PNULL CLRA             A=0 (NULL CHAR)
00655A FB0D BD F875   A        JSR    OUTCH     GO OUTPUT NULL
00656A FB10 5A                 DECB             DECREMENT COUNTER
00657A FB11 26 F9 FB0C         BNE    PNULL     IF NOT DONE, THEN LOOP
00658                          *
00659A FB13 FE A002   A        LDX    BEGA
00660A FB16 FF A040   A        STX    TW        TEMP BEGINING ADDRESS
00661A FB19 B6 A005   A PUN11  LDAA   ENDA+1
00662A FB1C B0 A041   A        SUBA   TW+1
00663A FB1F F6 A004   A        LDAB   ENDA
00664A FB22 F2 A040   A        SBCB   TW
00665A FB25 26 04 FB2B         BNE    PUN22
00666A FB27 81 10     A        CMPA   #16
00667A FB29 25 02 FB2D         BCS    PUN23
00668A FB2B 86 0F     A PUN22  LDAA   #15
00669A FB2D 8B 04     A PUN23  ADDA   #4
00670A FB2F B7 A03D   A        STAA   MCONT     FRAME COUNT THIS RECORD
00671A FB32 80 03     A        SUBA   #3
00672A FB34 B7 A03C   A        STAA   TEMP      BYTE COUNT THIS RECORD
00673                          *       PUNCH C/R,L/F,NULLS,S,1
00674A FB37 CE FAF9   A        LDX    #MTAPE1
00675A FB3A BD F87E   A        JSR    PDATA1
00676A FB3D 5F                 CLRB             ZERO CHECKSUM
```

```
0677                        *       PUNCH FRAME COUNT
0678A FB3E CE A03D  A               LDX     #MCONT
0679A FB41 8D 2E FB71               BSR     PUNT2       PUNCH 2 HEX CHAR
0680                        *       PUNCH ADDRESS
0681  A FB43 CE A040  A               LDX     #TW
0682A FB46 8D 23 FB71               BSR     PUNT2
0683A FB48 8D 27 FB71               BSR     PUNT2
0684                        *       PUNCH DATA
0685A FB4A FE A040  A               LDX     TW
0686A FB4D 8D 22 FB71   PUN32       BSR     PUNT2       PUNCH ONE BYTE (2 FRAMES)
0687A FB4F 7A A03C  A               DEC     TEMP        DEC BYTE COUNT
0688A FB52 26 F9 FB4D               BNE     PUN32
0689A FB54 FF A040  A               STX     TW
0690A FB57 53                       COMB
0691A FB58 37                       PSHB
0692A FB59 30                       TSX
0693A FB5A 8D 15 FB71               BSR     PUNT2       PUNCH CHECKSUM
0694A FB5C 33                       PULB                RESTORE STACK
0695A FB5D FE A040  A               LDX     TW
0696A FB60 09                       DEX
0697A FB61 BC A004  A               CPX     ENDA
0698A FB64 26 B3 FB19               BNE     PUN11
0699A FB66 BD F83E  A               JSR     PCRLF
0700A FB69 CE FC71  A               LDX     #MEOF
0701A FB6C BD F87E  A               JSR     PDATA1      OUTPUT EOF
0702A FB6F 20 32 FBA3               BRA     CTRL        BRANCH TO CONTRL
0704                        *       PUNCH 2 HEX CHAR, UPDATE CHECKSUM
0705A FB71 EB 00    A   PUNT2       ADDB    0,X         UPDATE CHECKSUM
0707  A FB73 7E F8BF  A               JMP     OUT2H       OUTPUT TWO HEX CHAR AND RTS
```

```
00708                          *
00709                          * SWI-1 SOFTWARE INTERRUPT LEVEL 1 PROCESSING
00710                          *
00711              FB76   A SWI1S  EQU     *
00712A FB76 BF A008   A         STS     SP          SAVE USER'S SP
00713                          *
00714A FB79 86 03     A         LDAA    #3
00715A FB7B BD FA6A   A         JSR     BRKSUB      GO TAKE OUT ALL BREAKS
00716                          *
00717                          * DECREMENT P-COUNTER
00718                          *
00719A FB7E 30                 TSX                 X:=STACK POINTER - 1
00720A FB7F 6D 06     A         TST     6,X         IF LOWER BYTE = 0 => BORROW
00721A FB81 26 02 FB85          BNE     SWI1S1      BRANCH IF BORROW NOT REQ'D
00722A FB83 6A 05     A         DEC     5,X         DECREMENT UPPER BYTE
00723A FB85 6A 06     A SWI1S1  DEC     6,X         DECREMENT LOWER BYTE
00724                          *
00725                          * TEST FOR ADDRESS TRACE OR BREAK
00726                          *
00727A FB87 EE 05     A         LDX     5,X         X:=P COUNTER
00728A FB89 BC A017   A         CPX     TRCADR      IS SWI FOR TRACE?
00729A FB8C 27 18 FBA6          BEQ     TRCINH      YES, GO TO TRACE INT HANDLER
00730                          *
00731A FB8E A6 00     A         LDAA    0,X         GET INSTRUCTION CAUSING SWI
00732A FB90 81 3F     A         CMPA    #SWI        WAS IT REPLACED BY CALL TO BREAK
00733A FB92 26 0C FBA0          BNE     BRKINH      YES, SO MUST BE A BREAK
00734                          *
00735                          * USER SWI-TRANSFER THROUGH LEVEL 2 SWI
00736                          *
00737A FB94 30                 TSX                 X:=STACK POINTER
00738A FB95 6C 06     A         INC     6,X         UPDATE LOW BYTE OF P-COUNTER
00739A FB97 26 02 FB9B          BNE     INCNOV      BRANCH IF NO CARRY
00740A FB99 6C 05     A         INC     5,X         UPDATE HIGH BYTE IF NECESSARY
00741                          *
00742A FB9B FE A00C   A INCNOV LDX     SWI2        X:=POINTER TO LEVEL 2 SWI HANDLE
00743A FB9E 6E 00     A         JMP     0,X         GO TO LEVEL 2 HANDLER
00744                          *
00745                          *
00746                          *
00747                          *
00748                          * BREAK INTERRUPT HANDLER
00749                          *
00750              FBA0   A BRKINH EQU     *
00751A FBA0 BD FAE2   A         JSR     PRINT       STOP AND SHOW REGS TO USER
00752A FBA3 7E F83A   A CTRL    JMP     CONTRL      RETURN TO CONTROL LOOP
```

```
00754                    *
00755                    * TRACE INTERRUPT HANDLER
00756                    * P-COUNTER  HAS BEEN DECREMENTED TO POINT AT SWI
00757                    * TRCINS HOLDS OP CODE REPLACED BY SWI
00758                    * X HOLDS ADDRESS OF WHERE TRACE SWI IS
00759                    *
00760A FBA6 B6 A013  A TRCINH LDAA   TRCINS   GET OP CODE OF TRACED INSTR
00761A FBA9 A7 00     A        STAA   0,X      RESTORE TO USER'S CODE
00762                    *
00763A FBAB 7D A043  A        TST    BRKTRC   IS PROCESSING TO BE
00764                    *                      IMMEDIATELY CONTINUED?
00765A FBAE 27 0F FBBF        BEQ    NBKTRC   BRANCH IF NOT
00766                    *
00767                    * PROCESSING IS TO "CONTINUE"
00768                    *
00769A FBB0 7F A043  A        CLR    BRKTRC   RESET CONTINUE FLAG
00770A FBB3 86 FF    A        LDAA   #$FF     FLAG TO SET BREAKS IN CODE
00771A FBB5 BD FA6A  A        JSR    BRKSUB   PUT BREAKS IN
00772A FBB8 7F A017  A        CLR    TRCADR   NO MORE TRACE, SO CLEAR ADDRESS
00773A FBBB 7F A018  A        CLR    TRCADR+1
00774A FBBE 3B                RTI             CONTINUE
00775                    *
00776                    * TRACE IS DUE TO N OR T TRACE COMMANDS
00777                    *
00778A FBBF BD FAE2  A NBKTRC JSR    PRINT    PRINT STACK
00779A FBC2 FE A01A  A        LDX    NTRACE   GET # INSTRUCTIONS TO TRACE
00780A FBC5 09                DEX             DECREMENT COUNT
007  A FBC6 FF A01A  A        STX    NTRACE   AND RESTORE
00782A FBC9 27 D8 FBA3        BEQ    CTRL     BRANCH IF ALL TRACES DONE
00783                    *
00784                    * TRACE NOT DONE - TRACE NEXT INSTRUCTION
00785                    *
00786A FBCB B6 A013  A CONTRC LDAA   TRCINS   GET CURRENT INSTRUCTION
00787A FBCE B7 A00E  A        STAA   BRINS    SAVE IN CASE IT'S A BRANCH
00788A FBD1 8D 70 FC43        BSR    OPCBYT   GO GET # BYTES/TYPE
00789A FBD3 4D                TSTA            CHECK FOR BRANCH
00790A FBD4 2A 35 FC0B        BPL    CKOBRA   CHECK FOR OTHER THAN BRANCH
```

```
0792                    *
0793                    * RELATIVE BRANCH TYPE INSTRUCTION
0794                    * DETERMINE WHERE TO PUT SWI
0795                    * S- HOLDS POINTER TO USER STACK AFTER SWI
0796                    *
0797A FBD5 32                   PULA            GET CONDITION CODE
0798A FBD7 34                   DES             UPDATE STACK PTR AFTER PULL
0799A FBD8 8A 10     A          ORAA    #%00010000 MAKE INT'S INHIBITED
0800A FBDA 06                   TAP             RESTORE USER'S C. CODE REG
0801A FBDB 7E A00E   A          JMP     BRINS   GO SEE HOW RELATIVE BRANCH
0802                    *                               FARES
0803                    *
0804                    * BRANCH WAS NOGO - PUT SWI AT NEXT INSTRUCTION
0805                    *
0806A FBDE 86 02     A BRNOGO LDAA    #2             A = # BYTES AFTER CURRENT INSTR
0807A FBE0 20 23 FC0B          BRA     CKOBRA  GO PUT SWI APPROPRIATELY
0808                    *
0809                    * BRANCH WAS GO, PUT SWI AT ADDRESS BEING
0810                    * JUMPED TO
0811                    *
0812A FBE2 FE A017   A BRGO   LDX     TRCADR  X : = TRACE ADDRESS
0813A FBE5 A6 01     A          LDAA    1,X     GET BRANCH OFFSET
0814A FBE7 08                   INX             OFFSET IS RELATIVE TO
0815A FBE8 08                   INX             INSTR FOLLOWING BRANCH
0816A FBE9 2B 12 FBFD           BMI     BRGODC  BRANCH IF OFFSET NEGATIVE
0817A FBEB 8D 16 FC03 BRG1      BSR     INCX    INCREMENT X BY AMOUNT IN
0818                    *                               A REG
0819A FBED FF A017   A BRG2     STX     TRCADR  SAVE ADDRESS OF NEXT
0820                    *                               INSTR TO STOP ON
0821A FBF0 A6 00     A          LDAA    0,X     GET INSTRUCTION TO BE REPLACED
0822A FBF2 B7 A019   A          STAA    TRCINS  SAVE
0823A FBF5 86 3F     A          LDAA    '#SWI   GET SWI OP CODE
0824A FBF7 A7 00     A          STAA    0,X     REPLACE INSTR WITH SWI
0825A FBF9 BE A008   A          LDS     SP      GET ORIGINAL STACK POINTER
0826A FBFC 3B                   RTI             TRACE ANOTHER INSTR
0827                    *
0828                    * X NEEDS TO BE DECREMENTED (OFFSET NEGATIVE)
0829                    *
0830A FBFD 09           BRGODC DEX              DECREMENT ADDRESS
0831A FBFE 4C                  INCA             INCREMENT COUNTER
0832A FBFF 26 FC FBFD          BNE     BRGODC  IF COUNTER NOT 0, BRANCH
0833A FC01 20 EA FBED          BRA     BRG2    IF DONE, GO RETURN TO USER PROG
0834                    *
0835                    * SUBROUTINE TO INCREMENT X BY CONTENTS OF A
0836                    *
0837A FC03 4D           INCX   TSTA             IS A = 0?
0838A FC04 27 04 FC0A          BEQ     INCXR   IF SO, INC DONE
0839A FC06 08           INXLP  INX              ELSE INCREMENT X
0840A FC07 4A                  DECA             DECREMENT COUNT
0841A FC08 26 FC FC06          BNE     INXLP   IF COUNT NOT YET 0, LOOP
0842A FC0A 39           INCXR  RTS              RETURN FROM THIS SUBROUTINE
```

```
00844                          *
00845                          * INSTRUCTION TO BE TRACED IS NOT A BRANCH.
00846                          *
00847A FC0B FE A017  A CKOBRA LDX    TRCADR   X : = TRACE ADDRESS
00848A FC0E E6 00    A        LDAB   0,X      GET INSTR TO BE TRACED
00849A FC10 C1 6E    A        CMPB   #$6E     IS IT A JUMP, INDEXED?
00850A FC12 27 1A FC2E        BEQ    JMPIDX   YES, GO SIMULATE JUMP IDXED
00851A FC14 C1 7E    A        CMPB   #$7E     JUMP, EXTENDED?
00852A FC16 27 1D FC35        BEQ    JMPEXT
00853A FC18 C1 AD    A        CMPB   #$AD     JSR, INDEXED?
00854A FC1A 27 12 FC2E        BEQ    JMPIDX   (JUMP IDXED IS SAME AS
00855                          *                    TRANSFER OF CONTROL)
00856A FC1C C1 BD    A        CMPB   #$BD     JSR, EXTENDED?
00857A FC1E 27 15 FC35        BEQ    JMPEXT
00858A FC20 C1 3B    A        CMPB   #$3B     RTI?
00859A FC22 27 15 FC39        BEQ    RTISIM
00860A FC24 C1 39    A        CMPB   #$39     RTS?
00861A FC26 27 16 FC3E        BEQ    RTSSIM
00862A FC28 C1 8D    A        CMPB   #$8D     BSR?
00863A FC2A 27 B6 FBE2        BEQ    BRGO     (BRANCH PROCESSING)
00864                          *
00865                          * NOT A BRANCH, JUMP, RTI, RTS
00866                          * A REGISTER HOLDS # BYTES IN INSTRUCTION
00867                          *
00868A FC2C 20 BD FBEB        BRA    BRG1     PUT IN NEW SWI AND
00869                          *                    TRACE NEXT INSTRUCTION
00870                          *
00871                          * JUMP, JSR INDEXED SIMULATION
00872                          *
00873A FC2E A6 01    A JMPIDX LDAA   1,X      A : = ADDRESS OFFSET
00874A FC30 30                TSX
00875A FC31 EE 03    A        LDX    3,X      GET TARGET'S X REG
00876A FC33 20 B6 FBEB        BRA    BRG1     UPDATE X, TRACE NEXT INSTR
00877                          *
00878                          * JUMP, JSR EXTENDED
00879                          *
00880A FC35 EE 01    A JMPEXT LDX    1,X      GET ADDRESS TO BE JUMPED TO
00881A FC37 20 B4 FBED        BRA    BRG2     GO TRACE NEXT INSTR
00882                          *
00883                          * RTI ENCOUNTERED
00884                          *
00885A FC39 30                RTISIM TSX
00886A FC3A EE 0C    A        LDX    12,X     GET P-COUNTER FROM STACK
00887A FC3C 20 AF FBED        BRA    BRG2     GO TRACE NEXT INSTR.
00888                          *
00889                          * RTS ENCOUNTERED
00890                          *
00891A FC3E 30                RTSSIM TSX
00892A FC3F EE 07    A        LDX    7,X      GET RETURN P-REG FROM STACK
00893A FC41 20 AA FBED        BRA    BRG2     GO TRACE NEXT INSTR
```

```
00835                    **************************
00836                    *
00837                    * OPBCYT
00838                    *
00839                    * THIS ROUTINE DETERMINES THE # OF BYTES IN AN INSTRU
00900                    * GIVEN ITS OP CODE.
00901                    *
00902                    * INPUT: A HOLDS THE OP CODE
00903                    *
00904                    * OUTPUT: X HOLDS INDEX OF TABLE ELEMENT
00905                    * B NOT RESTORED
00906                    * A HOLDS # BYTES IN INSTRUCTION
00907                    * EXCEPT FOR BRANCHES IN WHICH CASE A IS NEGATIVE
00908                    *
00909                    **************************
00910                    *
00911          FC43  A OPCBYT EQU    *
00912A FC43 16                   TAB             B:= OP CODE
00913A FC44 44                   LSRA
00914A FC45 44                   LSRA
00915A FC46 44                   LSRA            PUT 4 UPPER BITS OF OP CODE I
00916A FC47 44                   LSRA            LOWER 4 BITS OF A
00917                    *
00918A FC48 CE FC51  A           LDX    #OPBTTB  X:= ADDRESS OF TABLE
00919A FC4B 8D B6 FC03           BSR    INCX     INCREMENT X TO POINT TO CORRE
00920                    *
00921A FC4D A6 00    A           LDAA   0,X      GET TABLE ENTRY
00922A FC4F 26 0F FC60           BNE    OPBTRT   IF NOT 0 THEN NO FURTHER
00923                    *                              PROCESSING NEEDED
00924                    *
00925                    * IF TOP 4 BITS = 8 OR C, THEN THERE ARE TWO CLASSES
00926                    * OF INSTRUCTIONS: 2 BYTE INSTRUCTIONS AND
00927                    * CE, 8C AND 8E WHICH ARE 3 BYTE INSTRUCTIONS
00928                    *
00929A FC51 86 02    A           LDAA   #2       # BYTES IN MOST OF 8# INSTRUC
00930A FC53 C1 8C    A           CMPB   #$8C     3 BYTE INSTRUCTION?
00931A FC55 27 08 FC5F           BEQ    OPBT3    YES, UPDATE A
00932A FC57 C1 CE    A           CMPB   #$CE     3 BYTE INSTR?
00933A FC59 27 04 FC5F           BEQ    OPBT3    YES, UPDATE A
00934A FC5B C1 8E    A           CMPB   #$8E     3 BYTE INSTRUCTION?
00935A FC5D 26 01 FC60           BNE    OPBTRT   NO, RETURN
00936                    *
00937A FC5F 4C           OPBT3   INCA            # BYTES IN INSTRUCTION:=3
00938                    *
00939A FC60 39           OPBTRT  RTS             RETURN TO CALLER
```

```
00941                    *
00942                    * OP CODE TO NUMBER OF BYTES CONVERSION TABLE
00943                    *
00944                    *              # BYTES   TOP 4 BITS OF OPCODE
00945                    *              -------   --------------------
00946                    *
00947            FC81  A OPBTTB EQU     *
00948A FC81      01    A        FCB     1         0
00949A FC82      01    A        FCB     1         1
00950A FC83      82    A        FCB     2+%10000000 2 (MINUS=> BRANCHES)
00951A FC84      01    A        FCB     1         3
00952A FC85      01    A        FCB     1         4
00953A FC86      01    A        FCB     1         5
00954A FC87      02    A        FCB     2         6
00955A FC88      03    A        FCB     3         7
00956A FC89      00    A        FCB     0         8 * BYTES=2 EXCEPT 8C,8E
00957A FC8A      02    A        FCB     2         9
00958A FC8B      02    A        FCB     2         A
00959A FC8C      03    A        FCB     3         B
00960A FC8D      00    A        FCB     0         C * BYTES=2 EXCEPT CE
00961A FC8E      02    A        FCB     2         D
00962A FC8F      02    A        FCB     2         E
00963A FC70      03    A        FCB     3         F
```

```
00965                    *
00966                    * CONSTANT DATA
00967                    *
00968A FC71    53     A MEOF   FCC    /S9030000FC/
     A FC72    39     A
     A FC73    30     A
     A FC74    33     A
     A FC75    30     A
     A FC76    30     A
     A FC77    30     A
     A FC78    30     A
     A FC79    46     A
     A FC7A    43     A
00969A FC7B    04     A        FCB    4
00970A FC7C    13     A MCLOFF FCB    $13       READER OFF
00971A FC7D    0A     A MCL    FCB    $A,$D,$14,0,0,0,0,'*,4 LF,CR,PUNCH
     A FC7E    0D     A
     A FC7F    14     A
     A FC80    00     A
     A FC81    00     A
     A FC82    00     A
     A FC83    00     A
     A FC84    2A     A
     A FC85    04     A
00972A FC86    0D     A MCL1   FCB    $D,$A,0,0,0,0,4 CR LF
     A FC87    0A     A
     A FC88    00     A
     A FC89    00     A
     A FC8A    00     A
     A FC8B    00     A
     A FC8C    04     A
00973A FC8D    4D     A MCL2   FCC    /MIKBUG 2.0/
     A FC8E    49     A
     A FC8F    43     A
     A FC90    42     A
     A FC91    55     A
     A FC92    47     A
     A FC93    20     A
     A FC94    32     A
     A FC95    2E     A
     A FC96    30     A
00974A FC97    04     A        FCB    4
00975A FC98    43     A MCL3   FCC    /CC B A X P S/
     A FC99    43     A
     A FC9A    20     A
     A FC9B    42     A
     A FC9C    20     A
     A FC9D    20     A
     A FC9E    41     A
     A FC9F    20     A
     A FCA0    20     A
     A FCA1    20     A
     A FCA2    58     A
     A FCA3    20     A
     A FCA4    20     A
     A FCA5    20     A
     A FCA6    20     A
     A FCA7    50     A
```

```
      A FCA8     20     A
      A FCA9     20     A
      A FCAA     20     A
      A FCAB     20     A
      A FCAC     53     A
009784 FCAD     04     A          FCB     4
009785 FCAE     42     A MCL4     FCC     /BEG ADDR ?/
      A FCAF     45     A
      A FCB0     47     A
      A FCB1     20     A
      A FCB2     41     A
      A FCB3     44     A
      A FCB4     44     A
      A FCB5     52     A
      A FCB6     20     A
      A FCB7     3F     A
009794 FCB8     04     A          FCB     4
009795 FCB9     45     A MCL5     FCC     /END ADDR ?/
      A FCBA     4E     A
      A FCBB     44     A
      A FCBC     20     A
      A FCBD     41     A
      A FCBE     44     A
      A FCBF     44     A
      A FCC0     52     A
      A FCC1     20     A
      A FCC2     3F     A
0097A4 FCC3     04     A          FCB     4
( 131                        *
```

```
00983                   *
00984                   * MAXIMAL SOFTWARE IMPLEMENTATION OF THE
00985                   *    RITTER-ZETTNER STANDARDS
00986                   *
00987                   * COPYRIGHT (C) 1977 MOTOROLA INC. AND T.F. RITTER
00988                   *
00989                   *       COMMANDS FOR EXORTAPE
00990                   * C L D S :
00991                   * C - CHECK TAPE
00992                   * L - LOAD FROM TAPE TO MEMORY
00993                   * D - DUMP FROM MEMORY TO TAPE
00994                   * S - SET BAUD RATE
00995                   *
00996                   * SPEED :
00997                   * ENTER 04 08 12 16 20
00998                   *
00999                   * FILE ID :
01000                   * ENTER FOUR HEX CHARACTERS
01001                   *
01002                   * STARTSTOP PAGES :
01003                   * ENTER STARTING PAGE, TWO HEX CHARACTERS
01004                   * ENTER STOPPAGE, TWO HEX CHARACTERS
01005                   *
01006                   *
01007                   *
01008                   *          FILE SPECIFICATIONS
01009                   * ALC := UNDEFINED BIT; AUTOMATIC LEVEL CONTROL
01010                   * POST := UNDEFINED BIT; MISSING PULSE PROTECT
01011                   * START SEQUENCE := 55AFH
01012                   * CRC := CYCLIC REDUNDENCY CHECK BIT; X16+X15+X2+X0
01013                   * HEADERECORD := (32 ALC) (START SEQUENCE) (08H)
01014                   *    (16 FILE.ID) (8 # OF GOOD PAGES) (8 POST)
01015                   * TRAILERECORD := (16 ALC) (START SEQUENCE)
01016                   *    (20H) (8 # OF BAD PAGES) (8 POST)
01017                   * DUMPAGERECORD := (16 ALC) (START SEQUENCE) (10H)
01018                   *    (8 PAGE #) (2048 DATA) (16 CRC) (8 POST)
01019                   * CHARECORD := (32 ALC) (START SEQUENCE) (40H)
01020                   *    (8 LENGTH) (1-256 CHARS) (16 CRC) (8 POST)
01021                   * OTHERECORD := NEITHER HEADER NOR TRAILER RECORD
01022                   * SUBFILE := (HEADERECORD) (0-N OTHERECORDS)
01023                   * FILE := (1-N SUBFILES) (TRAILERECORD)
01024                   *
01025                   *
01026                   *          DATA SPECIFICATIONS
01027                   * SYNCHRONOUS DATA; NO START OR STOP BITS
01028                   * MSB SENT FIRST (CORRECT ORDER ON SCOPE)
01029                   * VOICE MESSAGES MAY BE PRESENT BETWEEN FILES
01030                   *
01031                   *
01032                   *          AUDIO MODULATION SPECIFICATIONS
01033                   * DOUBLE-FREQUENCY RETURN-TO-BIAS MODULATION
01034                   * LOGIC ONE = FIVE ELEMENT PATTERN: 0 1 0 1 0
01035                   * LOGIC ZERO = FIVE ELEMENT PATTERN: 0 1 0 0 0
01036                   * LOCAL EL CHEAPO (REALISTIC CTR-34) DOES 1200 BAUD
01037                   *    (DATA RATE EQUALS 1650 BAUD 2-STOP ASYNC)
01038                   * 0 ERRORS IN 2.6 MILLION BITS RECOVERED
01039                   *    FROM MEMOREX MRX2
01040                   *
```

```
01041                   *
01042                   *          AUDIO HARDWARE SPECIFICATIONS
01043                   * OUTPUT FROM PIA 32 THROUGH 11:1 VOLTAGE DIVIDER
01044                   *    (4.7K, 470) INTO MIKE JACK
01045                   * INPUT FROM SPKR JACK, THROUGH DC-RESTORING
01046                   *    CIRCUIT AND SCHMIDTT TRIGGER
01047                   *    (MC14583 PREFERRED) INTO PIA 30
01048                   *  RECOVERED PULSE PHASE MATTERS, SO USE SWITCH TO
01049                   *     SELECT PHASE -- BOTH AVAILIABLE FROM MC14583
```

```
)1052                   *
)1053                   *BAUD RATES:          400     800    1200    1600    2000
)1054                   *EQUIV ASYNC:         550    1100    1650    2200    2750
)1055                   *ELEMENT (USEC):      500     250     167     125     100
)1056                   *TOTCNT:               35      1B      12      0D      0B
)1057                   *PATDEL:               50      2S      1A      11      0D
)1058                   *
```

```
                    *SYSTEM STORAGE
1051
1052    8008  A TTYCON EQU     $8008     ACIA CONTROL
1053    8009  A TTY    EQU     $8009     TTY ACIA
1054    8004  A TP     EQU     $8004     TAPE PORT
105     8005  A TACON  EQU     $8005     TAPE CONTROL
1056    001B  A ESC    EQU     $1B       ESCAPE CHAR
1057
```

```
01070                          *
01071                          *
01072                          *
01073                          *
01074                          *
01075              0004   A EOT     EQU     4
01076                          *
01077                          *
01080                          *    MESSAGES
01081                          *
01082                          *
01083A FCC4     45     A MSG1    FCC     /EXORTAPE 4.3/
     A FCC5     58     A
     A FCC6     4F     A
     A FCC7     52     A
     A FCC8     54     A
     A FCC9     41     A
     A FCCA     50     A
     A FCCB     45     A
     A FCCC     20     A
     A FCCD     34     A
     A FCCE     2E     A
     A FCCF     33     A
01084A FCD0     04     A         FCB     EOT
01085A FCD1     43     A MSG2    FCC     /C L D S: /
     A FCD2     20     A
     A FCD3     4C     A
     A FCD4     20     A
     A FCD5     44     A
     A FCD6     20     A
     A FCD7     53     A
     A FCD8     3A     A
     A FCD9     20     A
01086A FCDA     04     A         FCB     EOT
01087A FCDB     46     A MSG3    FCC     /FILE ID: /
     A FCDC     49     A
     A FCDD     4C     A
     A FCDE     45     A
     A FCDF     20     A
     A FCE0     49     A
     A FCE1     44     A
     A FCE2     3A     A
     A FCE3     20     A
01088A FCE4     04     A         FCB     EOT
01089A FCE5     53     A MSG4    FCC     /STARTSTOP PAGES: /
     A FCE6     54     A
     A FCE7     41     A
     A FCE8     52     A
     A FCE9     54     A
     A FCEA     53     A
     A FCEB     54     A
     A FCEC     4F     A
     A FCED     50     A
     A FCEE     20     A
     A FCEF     50     A
     A FCF0     41     A
     A FCF1     47     A
     A FCF2     45     A
```

```
    A FCF3    53    A
    A FCF4    3A    A
    A FCF5    20    A
    A FCF6    04    A          FCB     EOT
    A FCF7    53    A MSG5     FCC     /SPEED: /
    A FCF8    50    A
    A FCF9    45    A
    A FCFA    45    A
    A FCFB    44    A
    A FCFC    3A    A
    A FCFD    20    A
    A FCFE    04    A          FCB     EOT
```

```
1094                    *
1095                    * PRINT LINE WITH A PRECEEDIOG!CR/LF
1096                    *   X POINTS TO STRING. STRING MUST
1097                    *   TERMINATE WITH A $4 CHARACTER.
1098                    *
1099A FCFF BD F89E   A PDATA  JSR    PCRLF
1100A FD02 7E F87E   A PDAT1P JMP    PDATA1
1101                    *
1102                    * TINS PROVIDES TAPE "IN'S" FOR MIKBUG KEYBOARD
1103                    *     CONTROL OF TAPE SYSTEM
1104                    *
1105                    *
1106                    * ENTER HERE
1107                    *
1108                    *
1109A FD05 8D 08 FD0F EXORT  BSR    EXOR     CALL TAPE ROUTINE VIA A BSR
1110A FD07 7E F9A4   A      JMP    ENT1     RETURN TO EXEC
1111                    *
1112                    *
1113                    *
1114                    *
1115                    *
1116                    * TINS' ERROR ROUTINE
1117                    *
1118A FD0A 86 3F     A ERR    LDAA   #'?      PRINT '?'
1119A FD0C BD F875   A      JSR    OUTCH
1120                    * FALL INTO TINS
1121                    *
1122                    *
1123                    *
1124           FD0F   A EXOR   EQU    *
1125A FD0F CE 1B2S   A TINS   LDX    #$1B2S   STANDARD SPEED
1126A FD12 FF A049   A      STX    TOTCNT
1127           FD1S   A SOFT   EQU    *
1128A FD15 CE FCC4   A TINSS  LDX    #MSG1    SEND FGM TITLE
1129A FD18 8D E5 FCFF TI1    BSR    PDATA
1130A FD1A CE FCF7   A TI2A   LDX    #MSGS    SEND SPEED QUESTION
1131A FD1D 8D E0 FCFF TI3    BSR    PDATA
1132A FD1F BD F8S5   A      JSR    BYTE     INPUT 2 HEX CHARACTERS
1133A FD22 81 20     A      CMPA   #$20     2000 BAUD?
1134A FD24 26 05 FD2B       BNE    TIN1
1135A FD26 CE 0B0D   A      LDX    #$0B0D
1136A FD29 20 22 FD4D       BRA    TINS
1137A FD2B 81 16     A TIN1   CMPA   #$16     1600 BAUD?
1138A FD2D 26 05 FD34       BNE    TIN2
1139A FD2F CE 0D11   A      LDX    #$0D11
1140A FD32 20 18 FD4D       BRA    TINS
1141A FD34 81 12     A TIN2   CMPA   #$12     1200 BAUD?
1142A FD36 26 05 FD3D       BNE    TIN3
1143A FD38 CE 1218   A      LDX    #$1218
1144A FD3B 20 10 FD4D       BRA    TINS
1145A FD3D 81 04     A TIN3   CMPA   #$04     0400 BAUD?
1146A FD3F 26 05 FD46       BNE    TIN4
1147A FD41 CE 3550   A      LDX    #$3550
1148A FD44 20 07 FD4D       BRA    TINS
1149A FD46 81 08     A TIN4   CMPA   #$08
1150A FD48 26 C0 FD0A       BNE    ERR      NOT A VALID SPEED
1151A FD4A CE 1B2S   A      LDX    #$1B26   800 BAUD IS NORMAL
```

```
01152A FD4D FF A049  A TIN5   STX    TOTCNT
01153A FD50 CE FCD1  A TIN8   LDX    #MSG2     SEND MODE QUESTION
01154A FD53 8D AA FCFF TI2    BSR    PDATA     C=CHECK; L=LOAD
0115 A FD55 BD F878  A        JSR    INCH      D=DUMP; S=SPEED
01156A FD58 81 53    A        CMPA   #'S
01157A FD5A 27 BE FD1A        BEQ    TI2A
01159                 *
01160                 *
01161A FD5C B7 A04B  A TIN6   STAA   T1        STORE MODE CHAR
01162A FD5F CE FCDB  A        LDX    #MSG3     SEND FILE ID PROMPT
01163A FD62 8D 98 FCFF TI4    BSR    PDATA
01164A FD64 BD F847  A        JSR    BADDR     GET FILE ID
01165A FD67 FF A045  A        STX    FIDH
01166A FD6A BD F89E  A        JSR    PCRLF
01167A FD6D B6 A04B  A        LDAA   T1
01168A FD70 81 44    A        CMPA   #'D       DUMP MODE?
01169A FD72 26 11 FD85        BNE    TIN7
0116 A FD74 CE FCE5  A        LDX    #MSG4     SEND START/STOP PROMPT
0117 A FD77 8D 89 FD02 TI5    BSR    PDAT1P
01172A FD79 BD F847  A        JSR    BADDR
01173A FD7C FF A047  A        STX    STARTP
01174A FD7F BD F89E  A        JSR    PCRLF
01175A FD82 7E FF37  A        JMP    MASTER
01176A FD85 81 43    A TIN7   CMPA   #'C       CHECK MODE?
01177A FD87 27 03 FD8E        BEQ    CHECK
01178A FD89 81 4C    A        CMPA   #'L       CHECK FOR LOAD
0117 A FD8B 27 03 FD90        BEQ    LOAD2
0117 A FD8D 7E FD0A  A        JMP    ERR
01181                 *
01182                 * FALL INTO LOAD
```

```
1185A FD90 20 04 FD96 LOADV  BRA    LOAD2
1186A FD92 86 01     A CHECK LDAA   #$01      INSERT CHECK COMMAND
1187A FD94 20 02 FD98        BRA    C12
1188A FD96 86 E7     A LOAD2 LDAA   #$E7      INSERT LOAD COMMAND
1189A FD98 B7 A04B   A C12   STAA   CL        (STORE B INDEXED)
1190A FD9B 7F A04C   A       CLR    CLL       NOP/ZERO DISPLACEMENT
1191A FD9E 86 39     A       LDAA   #$39      INSERT RTS
1192A FDA0 B7 A04D   A       STAA   CLLL
1193                      * FALL INTO GETLOAD
1195                      *
1196                      * GETLOAD SEARCHES FOR THE DESIRED FILE,
1197                      *      THEN LOADS IT INTO RAM
1198                      *
1199                      *      RAM:  Q, R, S, H, L (SCRATCH)
1200                      *            TOTCNT, FIDH, FIDL (PERM)
1201                      *
1202A FDA3 BD FF7B   A GETLOA JSR    LSETUP    SET UP PIA
1203A FDA6 8D 5E FE06        BSR    GETFIL    SEARCH FOR CORRECT FILE
1204A FDA8 27 22 FDCC        BEQ    GET4      OUT IFF ESCAPED
1205A FDAA 8D 54 FE00 GET2   BSR    STARTV    GET THE NEXT RECORD-TYPE
1206A FDAC 27 1E FDCC        BEQ    GET4      OUT IF ESCAPE
1207A FDAE C1 20     A       CMPB   #$20      TRAILERECORD?
1208A FDB0 27 0A FDBC        BEQ    GET1
1209A FDB2 C1 10     A       CMPB   #$10      DUMPAGERECORD?
1210A FDB4 26 F4 FDAA        BNE    GET2
1211A FDB6 8D 15 FDCD        BSR    DUMPR     BRING IT IN!
1212A FDB8 27 12 FDCC        BEQ    GET4      OUT IFF ESCAPE
1213A FDBA 20 EE FDAA        BRA    GET2
1214A FDBC 7D A053   A GET1  TST    V         CHECK PAGE COUNT
1215A FDBF 27 0B FDCC        BEQ    GET4
1216A FDC1 2B 04 FDC7        BMI    GET3
1217A FDC3 86 2D     A       LDAA   #'-      '-' FOR MISSING PAGE(S)
1218A FDC5 20 02 FDC9        BRA    GET5
1219A FDC7 86 2B     A GET3  LDAA   #'+      '+' FOR EXTRA PAGE(S)
1220A FDC9 BD F875   A GET5  JSR    OUTCH     PRINT IT!
1221A FDCC 39            GET4  RTS            RETURN
```

```
1224                    *
1225                    * DUMPR BRINGS IN A DUMPAGE RECORD
1226                    *
127                     *       RAM:  H, L (SCRATCH)
1228                    *       REGS:  ACCA, ACCB (SCRATCH); IX (PERM)
1229                    *       EXIT:  FALLS INTO CRCK
1230                    *
1231A FDCD 8D 2E FDFD DUMPR  BSR    LDV       GET PAGE NUMBER
1232A FDCF F7 A04E  A       STAB   H         )
1233A FDD2 7F A04F  A       CLR    L         ) LOAD IX!
1234A FDD5 FE A04E  A       LDX    H         )
1235A FDD8 8D 23 FDFD DUM1   BSR    LDV
1236A FDDA 27 49 FE25       BEQ    GE        OUT IFF ESCAPE
1237A FDDC BD A04B  A       JSR    CL        CHECK, OR LOAD
1238A FDDF 08               INX               STORAGE PTR
1239A FDE0 7C A04F  A       INC    L         BYTE COUNT
1240A FDE3 26 F3 FDD8       BNE    DUM1
1241A FDE5 7A A053  A       DEC    V         PAGE COUNT
1242                    * FALL INTO CRCK
1244                    *
1245                    * CRCK SHECKS THE CRC AND PRINTS A CHAR
1246                    *
1247                    *       RAM:  Q, R, S, TOTCNT (FROM LOADBYTE)
1248                    *       REGS:  ACCA (SCRATCH, ACCB (FROM LOADBYTE)
1249                    *
1250A FDE8 8D 13 FDFD CRCK   BSR    LDV       ) INPUT CRC CHARS
1251A FDEA 8D 11 FDFD       BSR    LDV       )
12  A FDEC B6 A051  A       LDAA   R         CHECK CRC REGISTERS
1253A FDEF BA A052  A       ORAA   S
1254A FDF2 26 04 FDF8       BNE    CRCK1
1255A FDF4 86 47    A       LDAA   #'G       PRINT G FOR GOOD CRC
1256A FDF6 20 02 FDFA       BRA    CRCK2
1257A FDF8 86 42    A CRCK1  LDAA   #'B       PRINT B FOR BAD CRC
1258A FDFA 7E FF73  A CRCK2  JMP    TTY01     PRINT IT!
1260A FDFD 7E FF6F  A LDV    JMP    LOADBV    GET A TAPE BYTE IN ACCB
1261A FE00 20 24 FE25 STARTV BRA    STARTF
```

```
)1263                          *
)1264                          * GETFILE LOOKS FOR:
)1265                          *      1)  A START SEQUENCE
)1266                          *      2)  A HEADERECORD TYPE
)1267                          *      3)  A FILE ID MATCH WITH FIDH, FIDL
)1268                          * AND RETURNS WHEN FOUND, OR ESCAPED
)1269                          *
)1270                          *      REGS:  ACCA, ACCB (SCRATCH ONLY)
)1271                          *
)1272A FE02 86 58     A G1      LDAA   #'X        INDICATE WRONG FILE FOUND
)1273A FE04 8D F4 FDFA         BSR    CRCK2      SEND CHAR TO TTY
)1274A FE06 8D 1E FE26 GETFIL  BSR    STARTF
)1275A FE08 27 1B FE25         BEQ    G2         OUT IFF ESCAPE
)1276A FE0A C1 08     A        CMPB   #$08       HEADERECORD-TYPE?
)1277A FE0C 26 F8 FE06         BNE    GETFIL     BRANCH IF NOT
)1278A FE0E 8D ED FDFD         BSR    LDV        GET BYTE IN ACCB
)1279A FE10 F1 A045   A        CMPB   FIDH       GOOD FILE ID(H) ?
)1280A FE13 26 ED FE02         BNE    G1
)1281A FE15 8D E6 FDFD         BSR    LDV
)1282A FE17 F1 A046   A        CMPB   FIDL       GOOD FILE ID(L) ?
)1283A FE1A 26 E6 FE02         BNE    G1
)1284A FE1C 86 48     A        LDAA   #'H        INDICATE HEADER FOUND
)1285A FE1E 8D DA FDFA         BSR    CRCK2
)1286A FE20 8D DB FDFD         BSR    LDV
)1287A FE22 F7 A053   A        STAB   V          STORE PAGE COUNT
)1288A FE25 39           G2    RTS
)1290                          *
)1291                          * STARTFIND LOOKS FOR THE 16-BIT START SEQUENCE $89A    A
)1292                          *      RETURNS WITH THE NEXT BYTE, THE RECORD TYPE,
)1293                          *      IN Q AND ACCB.
)1294                          *
)1295                          *      RAM:  Q, R, S, TOTCNT (PERM); H (SCRATCH)
)1296                          *      REGS:  ACCA, ACCB (SCRATCH ONLY)
)1297                          *      EXIT:  FALLS INTO LOADBYTE, WHICH
)1298                          *             FALLS INTO LOADBIT, WHICH
)1299                          *             FALLS INTO CRC
)1300                          *      ESCAPE:  ESC IN COMMAND PORT GETS OUT
)1301                          *                  (TEST DONE IN CRC)
)1302                          *
)1303A FE26 7F A04E   A STARTF  CLR    H          ACCUMULATES 16 BITS
)1304A FE29 7F A050   A         CLR    Q
)1305A FE2C 78 A050   A STA1    ASL    Q          SHIFT THE 16-BIT REGISTER
)1306A FE2F 79 A04E   A         ROL    H
)1307A FE32 8D 24 FE58         BSR    LOADBI
)1308A FE34 27 EF FE25         BEQ    G2         OUT IFF ESCAPED
)1309A FE36 B6 A04E   A         LDAA   H
)1310A FE39 F6 A050   A         LDAB   Q
)1311A FE3C 81 89     A         CMPA   #$89       START SEQUENCE
)1312A FE3E 26 EC FE2C         BNE    STA1
)1313A FE40 C1 AF     A         CMPB   #$AF
)1314A FE42 26 E8 FE2C         BNE    STA1
)1315A FE44 7F A051   A STA2    CLR    R          CLEAR THE CRC REGISTERS
)1316A FE47 7F A052   A         CLR    S
)1317                          * FALL INTO LOADBYTE TO RETURN A BYTE
```

```
1320                    *
1321                    * LOADBYTE RECOVERS OF DATA IN Q AND ACCB, AND DOES CRC
1322                    *
1323                    *       RAM:  Q, R, S, TOTCNT (PERM)
132                     *       REGS:  ACCA, ACCB (SCRATCH ONLY)
132                     *       EXIT:  FALLS INTO LOADBIT, WHICH
1326                    *              FALLS INTO CRC
1327                    *
1328A FE4A C6 02     A LOADBY LDAB   #$02      SET STOP
1329A FE4C F7 A050  A        STAB   Q
1330A FE4F 8D 07 FE58 LOAD1  BSR    LOADBI
1331A FE51 27 D2 FE25        BEQ    G2         OUT IFF ESCAPE
1332A FE53 78 A050  A        ASL    Q          STOP INTO CARRY?
1333A FE56 24 F7 FE4F        BCC    LOAD1      BRANCH IF NO
1334                    * FALL INTO LOADBIT FOR LAST BIT
1336                    *
1337                    * LOADBIT RECOVERS ONE BIT OF DATA IN Q AND ACCB,
1338                    *       AND DOES CRC IN R AND S
1339                    *
1340                    *       RAM:  Q, R, S, TOTCNT (PERM)
1341                    *       REGS:  ACCA, ACCB (SCRATCH ONLY)
1342                    *       EXIT:  BRANCHES OR FALLS INTO CRC ROUTINE
1343                    *              DATA BIT GOES INTO Q LSB
1344                    *
1345A FE58 5F         LOADSI CLRB
1346A FE59 5A         LOADBS DECB
1347A FE5A 27 1F FE7B        BEQ    LOADB5
1348A FE5C 8D 43 FEA1        BSR    TAIN1V     GET TAPE DATA IN CARRY
134  A FE5E 25 F9 FE59        BCS    LOADBS     WAIT FOR LOW
1350A FE60 5A         LOADB2 DECB
1351A FE61 27 18 FE7B        BEQ    LOADB5
1352A FE63 8D 3C FEA1        BSR    TAIN1V     WAIT FOR HIGH
1353A FE65 24 F9 FE60        BCC    LOADB2     ) (FRONT OF SYNC EL)
1354A FE67 F6 A043  A        LDAB   TOTCNT     3.5 ELS DELAY
1355A FE6A 5A         LOADB3 DECB
1356A FE6B 27 0E FE7B        BEQ    LOADB5
1357A FE6D 8D 32 FEA1        BSR    TAIN1V     )  WAIT FOR LOW
1358A FE6F 25 F9 FE6A        BCS    LOADB3     )  (END OF SYNC EL)
1359A FE71 5A         LOADB4 DECB
1360A FE72 27 07 FE7B        BEQ    LOADB5     DONE YET?
1361A FE74 8D 2B FEA1        BSR    TAIN1V
1362A FE76 24 F9 FE71        BCC    LOADB4
1363A FE78 7C A050  A        INC    Q          STORE A '1' BIT
1364A FE7B B6 A050  A LOADB5 LDAA   Q          GET DATA FOR CRC
1365                    * FALL INTO CRC1
```

```
1368                    *
1369                    *  CRC ENTERS A BIT INTO CRC REGISTERS R AND S
1370                    *     USES CRC POLYNOMIAL:  X15 + X15 + X2 + X0
1371                    *
1372                    *     ENTRY:  ACCA HOLDS NEW BIT
1373                    *     RAM:  R, S (PERM)
1374                    *     REGS:  ACCA, ACCB (SCRATCH)
1375                    *     EXIT:  ACCB = 0, Z=1 IFF ESC
1376                    *
1377A FE7E 46          CRC1   RORA             LSB INTO CARRY
1378A FE7F 46                 RORA             CARRY INTO MSB
1379A FE80 84 80     A CRC2   ANDA   #$80      MASK MSB (DATA BIT)
1380A FE82 B8 A051   A        EORA   R         ENTER DATA BIT
1381A FE85 F6 A052   A        LDAB   S
1382A FE88 58                 ASLB             SHIFT 16 BITS LEFT
1383A FE89 49                 ROLA
1384A FE8A 24 04 FE90         BCC    CRC3      IF B15 HIGH. . .
1385A FE8C 88 30     A        EORA   #$30      ENTER CRC POLYNOMIAL
1386A FE8E C8 05     A        EORB   #$05
1387A FE90 B7 A051   A CRC3   STAA   R
1388A FE93 F7 A052   A        STAB   S
1389A FE96 F6 A050   A        LDAB   Q
1390A FE99 BD FF7F   A        JSR    TTYIN1    CHECK FOR ESCAPE
1391A FE9C 84 7F     A        ANDA   #$7F      MASK PARITY
1392A FE9E 81 1B     A        CMPA   #ESC
1393A FEA0 39                 RTS
1394A FEA1 7E FF93   A TAIN1V JMP    TAIN1
```

```
1397                    *
1398                    * BYTEOUT SENDS BYTE IN ACCA TO TAPE
1399                    *
1400                    *       RAM:   R, S (CHANGED IN CRC)
140                     *       REGS:  ACCA, ACCB (DESTROYED)
140                     *       EXIT:  ACCA = 0, ACCB UNDEFINED
1403                    *
1404                    *       ACCA = DATA BYTE (SHIFTED)
1405                    *       ACCB = RECORDING PATTERN (ONE BIT)
1406                    *
1407A FEA4 36             BYTEOU PSHA             SAVE THE DATA BYTE
1408A FEA5 8D D9 FE80            BSR     CRC2     ) DO THE CRC FIRST
1409A FEA7 32                  PULA             RECOVER THE DATA BYTE
1410A FEA8 0D                  SEC              SET STOP
1411A FEA9 49                  ROLA             DATA BIT INTO CARRY
1412A FEAA 20 06 FEB2          BRA     BY2
1414A FEAC 32           BY1    PULA             RECOVER FROM DONE TEST
1415A FEAD 36                  PSHA             SAVE SHIFTING BYTE
1416A FEAE 8D D0 FE80          BSR     CRC2
1417A FEB0 32                  PULA             RECOVER SHIFTING BYTE
1418A FEB1 48                  ASLA             DATA BIT INTO CARRY
1419A FEB2 C6 0A     A BY2     LDAB    #$0A     RECORDING PATTERN
1420A FEB4 BD FF8C   A BY3     JSR     TAOU1    SEND ACCB TO TAPE
1421A FEB7 37                  PSHB             SAVE PATTERN
1422A FEB8 F6 A04A   A         LDAB    PATDEL   ELEMENT DELAY
1423A FEBB 5A           BY4    DECB
1424A FEBC 26 FD FEBB          BNE     BY4      BRANCH IF ELEMENT NOT DONE
1425A FEBE 33                  PULB             RECOVER PATTERN
1427A FEBF 59                  ROLB             NEXT ELEMENT (DATA FROM CARRY)
1427A FEC0 24 F2 FEB4          BCC     BY3      BRANCH IF PATTERN NOT DONE
1428A FEC2 36                  PSHA             SAVE DATA BYTE BEFORE TEST
1429A FEC3 48                  ASLA             TEST ACCA
1430A FEC4 26 E6 FEAC          BNE     BY1      BRANCH IF BYTE NOT DONE
1431A FEC6 31           BY5    INS              RESTORE STACK
1432A FEC7 39                  RTS
```

```
1435                        *
1436                        * PREAMBLE SENDS OUT ALC BITS, AND THE START SEQUENCE,
1437                        *     THEN INITS THE CRC REGISTERS
1438                        *
1439                        *     RAM:  R, S (CHANGED)
1440                        *     REGS:  ACCA, ACCB (DESTROYED)
1441                        *     EXIT:  ACCA = 0, ACCB UNDEFINED
1442                        *
1443A FEC8 8D 2E FEF8 PREAMB BSR    BYTOV1   ALC BITS
1444A FECA 8D 2C FEF8        BSR    BYTOV1
1445A FECC 86 83    A       LDAA   #$83     START SEQUENCE (H)
1446A FECE 8D 28 FEF8        BSR    BYTOV1
1447A FED0 86 AF    A       LDAA   #$AF     START SEQUENCE (L)
1448A FED2 8D 24 FEF8        BSR    BYTOV1
1449A FED4 7F A051  A       CLR    R        ) CLEAR THE CRC REGISTERS
1450A FED7 7F A052  A       CLR    S        )
1451A FEDA 39               RTS
1453                        *
1454                        * HEADERECORD SENDS ALC BITS, THE START SEQUENCE,
1455                        *     HEADERECORD-TYPE, FILE ID, AND THE NUMBER OF
1456                        *     PAGES TO BE DUMPED
1457                        *
1458                        *     RAM:  FIDH, FIDL (UNMODIFIED)
1459                        *           R, S (CHANGED)
1460                        *     REGS:  ACCA, ACCB (DESTROYED)
1461                        *     EXIT:  ACCA = 0, ACCB UNDEFINED
1462                        *
1463A FEDB 8D 1B FEF8 HEADER BSR    BYTOV1   ALC BITS
1464A FEDD 8D 19 FEF8        BSR    BYTOV1
1465A FEDF 8D E7 FEC8        BSR    PREAMB   START A RECORD
1466A FEE1 86 08    A       LDAA   #$08     HEADERTYPE
1467A FEE3 8D 13 FEF8        BSR    BYTOV1
1468A FEE5 B6 A045  A       LDAA   FIDH     FILE ID(H)
1469A FEE8 8D 0E FEF8        BSR    BYTOV1
1470A FEEA B6 A046  A       LDAA   FIDL     FILE ID(L)
1471A FEED 8D 09 FEF8        BSR    BYTOV1
1472A FEEF B6 A048  A       LDAA   STOPPG   STOPAGE
1473A FEF2 B0 A047  A       SUBA   STARTP   STARTPAGE
1474A FEF5 4C               INCA
1475A FEF6 8D 00 FEF8        BSR    BYTOV1   SEND # PAGES
1476A FEF8 7E FF72  A BYTOV1 JMP    BYTEOV   EXTRA BITS
```

```
1479                        *
1480                        * TRAILERECORD SENDS A TRAILERECORD TO TAPE
1481                        *
1482                        *       RAM:  R, S (PERM)
1483                        *       REGS:  ACCA, ACCB (DESTROYED)
1484                        *
1485A FEF3 8D C3 FEC3 TRAILR BSR    PREAMB   START A RECORD
1486A FEF5 86 20     A        LDAA   #$20     TRAILERECORD TYPE
1487A FEF7 8D F7 FEF8        BSR    BYTOV1
1488A FF01 20 F5 FEF8        BRA    BYTOV1   EXTRA BITS
1490                        *
1491                        * DUMPAGERECORD SENDS THE START SEQUENCE, DUMPAGE-TYPE,
1492                        *       PAGE NUMBER, 2048 BITS DATA, AND THE CRC CHARACTE
1493                        *
1494                        *       RAM:  H, R, S (PERM); L (SCRATCH)
1495                        *       REGS:  ACCA, ACCB (DESTROYED), NEW IX
1496                        *       EXIT:  ACCA = 0
1497                        *
1498A FF03 8D C3 FEC8 DUMPAG BSR    PREAMB   START A RECORD
1499A FF05 86 10     A        LDAA   #$10     DUMPAGE TYPE
1500A FF07 8D EF FEF8        BSR    BYTOV1
1501A FF09 B6 A04E  A        LDAA   H        SEND PAGE NUMBER
1502A FF0C 8D EA FEF8        BSR    BYTOV1
1503A FF0E 7F A04F  A        CLR    L        DO ENTIRE PAGE
1504A FF11 FE A04E  A        LDX    H        POINT AT THE DATA
1505A FF14 A6 00    A DUMP1  LDAA   0,X
1506A FF16 8D E0 FEF8        BSR    BYTOV1   SEND DATA
1507A FF18 08                INX
1508A FF19 7C A04F  A        INC    L        BYTE CTR
1509A FF1C 26 F6 FF14        BNE    DUMP1
1510A FF1E 8D 07 FF27        BSR    SENCRC
1511A FF20 86 44    A        LDAA   #'D      PRINT D FOR EACH PAGE DUMPED
1512A FF22 8D 54 FF78        BSR    TTY01
1513A FF24 4F                CLRA
1514A FF25 20 D1 FEF8        BRA    BYTOV1   EXTRA BITS
```

```
1517                        *
1518                        * SENCRC SENDS THE CRC REGISTERS TO TAPE
1519                        *
1520                        *       RAM:  R, S (FREED AFTER THIS); L (SCRATCH)
1521                        *       REGS:  ACCA, ACCB (DESTROYED)
1522                        *
1523A FF27 B6 A052  A SENCRC LDAA   S
1524A FF2A B7 A04F  A        STAA   L          TEMPORARY CRC(L) STORAGE
1525A FF2D B6 A051  A        LDAA   R
1526A FF30 8D C5 FEF8        BSR    BYTOV1     SEND CRC(H)
1527A FF32 B6 A04F  A        LDAA   L
1528A FF35 20 C1 FEF8        BRA    BYTOV1     SEND CRC(L)
1530                        *
1531                        * MASTERDUMP SENDS A COMPLETE FILE TO TAPE:
1532                        *       HEADERECORD, DUMPAGERECORDS (1-256), TRAILERECORD
1533                        *
1534                        *       RAM:  STARTP, STOPPG (UNMODIFIED)
1535                        *             H, L (TEMP)
1536                        *       REGS:  NEW EVERYTHING
1537                        *
1538A FF37 8D 45 FF7E MASTER BSR    DSETUP    SETUP DUMP PORT
1539A FF39 8D A0 FEDB        BSR    HEADER    SEND HEADERECORD
1540A FF3B B6 A047  A        LDAA   STARTP    GET STARTPAGE
1541A FF3E B7 A04E  A        STAA   H         PRESENT PAGE
1542A FF41 7A A04E  A        DEC    H
1543A FF44 7C A04E  A MAST1  INC    H
1544A FF47 8D BA FF03        BSR    DUMPAG
1545A FF49 B6 8003  A        LDAA   TTY
1546A FF4C 84 7F    A        ANDA   #$7F
1547A FF4E 81 1B    A        CMPA   #ESC
1548A FF50 27 08 FF5A        BEQ    MAST3
1549A FF52 F6 A048  A        LDAB   STOPPG    GET STOPAGE
1550A FF55 F1 A04E  A        CMPB   H         PRESENT PAGE
1551A FF58 26 EA FF44        BNE    MAST1     BRANCH IF NOT DONE
1552A FF5A 8D 9F FEFB MAST3  BSR    TRAILR    SEND TRAILERECORD
1553A FF5C 39         MAST2  RTS               RETURN
1555A FF5D 7F 8005  A SETUP  CLR    TACON     INTO DATA DIRECTION REG.
1556A FF60 86 04    A        LDAA   #$04      B2 AN OUTPUT, REST INPUTS
1557A FF62 B7 8004  A        STAA   TP        (ONLY B0 USED FOR INPUT)
1558A FF65 B7 8005  A        STAA   TACON     BACK TO DATA REGISTER
1559A FF68 39                RTS
```

```
562                     *
563                     *  VECTORS- IN OTHER VERSIONS OF THIS TAPE
564                     *   INTERFACE THESE VECTORS WILL BE IMPLEMENTED
565                     *   IN RAM TO ALLOW THE USER ACCESS TO
566                     *   THE TAPE SYSTEM.
567                     *
568A FF69 7E FF81   A TAIN1  JMP     TAIN2     TAPE IN PORT VECTOR
569A FF6C 7E FF3E  A TAOU1  JMP     TAOU2     TAPE OUT PORT VECTOR
570A FF6F 7E FE4A  A LOADBV JMP     LOADBY    TAPE BYTE IN VECTOR
571A FF72 7E FEA4  A BYTEOV JMP     BYTEOU    TAPE.BYTE OUT VECTOR
572A FF75 7E FF36  A TTYIN1 JMP     TTYIN2    CONTROL IN PORT VECTOR
573A FF78 7E FF8A  A TTYO1  JMP     TTYO2     CONTROL OUT PORT VECTOR
574A FF7B 7E FF5D  A LSETUP JMP     SETUP     TAPE OUT PIA INIT
575A FF7E 7E FF3D  A DSETUP JMP     SETUP     TAPE IN PIA INIT
576                     *
577                     *
578A FF81 B6 8004  A TAIN2  LDAA    TP        ACCA FROM TAPE
579A FF84 46               RORA               DATA BIT IN CARRY
580A FF85 39               RTS
582A FF86 B6 8009  A TTYIN2 LDAA    TTY       ACCA FROM ACIA
583A FF89 39               RTS
585A FF8A B7 8009  A TTYO2  STAA    TTY       SEND ACCA TO ACIA
586A FF8D 39               RTS
588A FF8E F7 8004  A TAOU2  STAB    TP        ACCB TO TAPE
589A FF91 39               RTS
```

```
1592                    *
1593                    ******** COPYRIGHT (C) 1977 MOTOROLA INC - AUSTIN TEXAS
1594                    *
1595                    *
1596                    **********************
1597                    *
1598                    * UNSIGNED MULTIPLY -----
1599                    *
1600                    *   THIS ROUTINE MULTIPLIES THE UNSIGNED NUMBER IN THE
1601                    *   A REGISTER WITH THE UNSIGNED NUMBER IN THE B
1602                    *   REGISTER AND PUTS THE ANSWER IN THE CONCATENATED
1603                    *   A:B WHERE A IS THE MSB. THE ROUTINE IS
1604                    *   RE-ENTRANT AND POSITION INDEPENDENT AS WELL
1605                    *   AS BEING ROMABLE. THE X REGISTER IS DESTROYED
1606                    *   BY THE ROUTINE.
1607                    *
1608                    *   DURING EXECUTION THE STACK CONTAINS:
1609                    *   0,X  =  LOOP COUNTER
1610                    *   1,X  =  MULTIPLIER (B REG ON ENTRY)
1611                    *
1612                    *   THE ALGORITHM USED MAY NOT APPEAR THE FASTEST
1613                    *   ON PAPER BECAUSE IT ALWAYS REQUIRES 8 PASSES
1614                    *   THRU THE LOOP BUT BECAUSE OF THE FACT ALL
1615                    *   CALCULATIONS CAN BE DONE IN THE REGISTERS IT
1616                    *   IS FASTER EXCEPT FOR WHEN THE MULTIPLICAND
1617                    *   IS VERY SMALL (<10). THE METHOD IS A SHIFT AND
1618                    *   ADD TECHNIQUE THAT BEGINS WITH THE MS BIT
1619                    *   AND WORKS DOWN TO THE LS BIT.
1620                    *
1621                    *   EXECUTION TIME: (29 + ZEROES*13 + ONES*25) CYCLES
1622                    *     WHERE ZEROES AND ONES ARE THE 0'S AND 1'S IN
1623                    *     THE A-REG ON CALL.
1624                    *
1625                    *     AVERAGE EXECUTION TIME: 209 CYCLES
1626                    *
1627                    *
1628                    **********************
1629                    *
1630A FF92 37          MUL     PSHB            PUT MULTIPLIER ON THE STACK
1631A FF93 C6 08    A         LDAB    #8      PUT COUNTER ON STACK
1632A FF95 37                PSHB
1633A FF96 30                TSX             SET X TO POINT TO STACK
1634A FF97 5F                CLRB            CLEAR PLACE TO START ANSWER
1635A FF98 58          ML1   ASLB            SHIFT ANS 1 LEFT AND INTO A
1636A FF99 49                ROLA            SHIFT WHATS LEFT OF THE MULTIPLI
1637A FF9A 24 04 FFA0        BCC     ML2     BRANCH IF NO ADD NEEDED
1638A FF9C EB 91    A        ADDB    1,X     ADD MULTIPLIER AT THIS POSITION
1639A FF9E 89 00    A        ADCA    #0      ADD CARRY TO A IF ANY
1640A FFA0 6A 00    A ML2    DEC     0,X     DONE?
1641A FFA2 26 F4 FF98        BNE     ML1     NOPE
1642A FFA4 31                INS             YES,CLEAN UP THE STACK
1643A FFA5 31                INS
1644A FFA6 39                RTS
```

```
1646                    *
1647                    **********************
1648                    *
1649                    * SIGNED MULTIPLY -----
1650                    *
165                     *     THIS ROUTINE MULTIPLIES THE TWO SIGNED NUMBERS IN
1652                    *     THE A AND B REGISTERS AND PUTS THE SIGNED
1653                    *     16 BIT ANSWER IN THE CONCATENATED A:B WHERE
1654                    *     THE A REGISTER IS THE MSB. THIS ROUTINE DESTROYS
1655                    *     THE CALLER'S X-REGISTER.
1656                    *
1657                    *     DURING EXECUTION THE STACK CONTAINS:
1658                    *     0,X  =  FLAG
1659                    *
1660                    *     THE ROUTINE IS PURE,RE-ENTRANT AND POSITION INDEPE
1661                    *
1662                    *     THE METHOD USE EVALUATES EACH ARGUMENT AND IF IT
1663                    *     IS NEGATIVE IT IS 2'S COMPLEMENTED AND THE FLAG IS
1664                    *     INCREMENTED. IF AFTER EVALUATING BOTH ARGUMENTS TH
1665                    *     FLAG IS EVEN (0 OR 2) THEN THE ANSWER WILL BE POSI
1666                    *     ELSE,THE ANSWER WILL NEED TO BE 2'S COMPLEMENTED.
1667                    *     UNSIGNED MULTIPLY ROUTINE IS USED TO MULTIPLY THE
1668                    *     CORRECTED REGISTERS
1669                    *
1670                    *    AVERAGE EXECUTION TIMES:
1671                    *     A-REG  B-REG
1672                    *      +      +          268 AVERAGE
1673                    *      +      -          283 AVERAGE
167                     *      -      +          283 AVERAGE
167                     *      -      -          286 AVERAGE
1676                    *
1677                    *
1678                    **********************
1679                    *
1680A FFA7 34             SMUL   DES              CARVE OUT A PLACE ON THE STACK F
1681A FFA8 30                    TSX              GET POINTER TO THAT PLACE
1682A FFA9 6F 00     A           CLR    0,X       CLEAR FLAG
1683A FFAB 4D                    TSTA             CHECK MULTIPIER
1684A FFAC 2A 03 FFB1            BPL    SML1      POSITIVE, NO COMP. NEEDED
1685A FFAE 40                    NEGA             2'S COMP. ARGUMENT
1686A FFAF 6C 00     A           INC    0,X       INCR FLAG
1687A FFB1 5D           SML1     TSTB             TEST OTHER ARG
1688A FFB2 2A 03 FFB7            BPL    SML2      NO COMP NEEDED
1689A FFB4 50                    NEGB             2'S COMP ARGUMENT
1690A FFB5 6C 00     A           INC    0,X       INCR FLAG
1691A FFB7 8D D9 FF92 SML2       BSR    MUL       GO DO UNSIGNED MULTIPLY
1692A FFBB 30                    TSX              GET BACK PTR TO FLAG
1693A FFBA 66 00     A           ROR    0,X       SEE IF FLAG IS EVEN OR ODD
1694A FFBC 24 04 FFC2            BCC    SML3      EVEN ANSWER IS OKAY 'CAUSE ITS P
1695A FFBE 40                    NEGA             DBL PRECISION 2'S COMP
1696A FFBF 50                    NEGB
1697A FFC0 82 00     A           SBCA   #0
169    FFC2 31           SML3     INS              CLEANUP STACK
1699A FFC3 39                    RTS              RETURN TO CALLER
```

```
1701                    *
1702                    ************************
1703                    *
1704                    * POSITIVE DIVIDE -----
1705                    *
1706                    *   THIS ROUTINE DIVIDES THE 16 BIT POSITIVE  DIVIDE
1707                    *   IN THE A:B REGISTERS (A IS THE MSB) BY AN 8
1708                    *   BIT POSITIVE  DIVISOR POINTED TO BY THE X-REGISTER.
1709                    *   THE QUOTIENT IS IN B ON EXIT AND THE REMAINDER
1710                    *   IS IN THE A-REGISTER. THE X-REGISTER IS
1711                    *   DESTROYED.
1712                    *   IF DIVISION BY ZERO WAS ATTEMPTED OR THE
1713                    *   QUOTIENT WILL NOT FIT IN 8 BITS THE OVERFLOW
1714                    *   BIT IS SET ON EXIT. ALSO V IS SET IF EITHER
1715                    *   THE DIVIDEND OR THE DIVISOR IS NEGATIVE ON CALL.
1716                    *   OTHERWISE V IS CLEARED.
1717                    *
1718                    *
1719                    *   THE ROUTINE IS PURE,RE-ENTRANT AND POSITION INDEPEN
1720                    *
1721                    *   DURING EXECUTION THE STACK CONTAINS
1722                    *   0,X  =  LOOP COUNTER
1723                    *   1,X  =  DIVISOR
1724                    *   2,X  =  DIVIDEND MSB (ONLY USED FOR TEMP STORE)
1725                    *
1726                    *   THE METHOD USED IS NON RESTORING DIVIDE WHERE THE
1727                    *   DIVIDEND AND DIVISOR ARE KNOWN TO BE POSITIVE. THIS
1728                    *   METHOD PROVED FASTEST SINCE IT CAN BE CARRIED OUT
1729                    *   ESSENTIALLY IN THE ACCUMULATORS. THE ALGORITHM TRIE
1730                    *   TO GET THE REMAINDER AS NEAR ZERO AS POSSIBLE
1731                    *   AND SOMETIMES ADDS THE DIVISOR AND SOMETIMES
1732                    *   SUBTRACTS IT DEPENDING ON WHICH SIDE OF ZERO THE
1733                    *   PARTIAL REMAINDER RESIDES AT ANY TIME. FOR MORE
1734                    *   INFO READ: 'AN ALGORITHM FOR NON RESTORING DIVISION
1735                    *   S. SANYAL ; 'COMPUTER DESIGN' /MAY 1977.
1736                    *
1737                    *   EXECUTION TIME AVERAGE (NO OVERFLOWS): 289 CYCLES
1738                    *     THIS TIME IS RELATIVELY INDEPENDENT OF THE
1739                    *     CALLING VALUES.
1740                    *
1741                    *
1742                    *
1743                    ************************
1744                    *
01745A FFC4 36          DIV     PSHA                    SAVE DIVIDEND MSB A SECOND
01746A FFC5 A6 00     A         LDAA    0,X             FETCH THE DIVISOR
01747A FFC7 2B 29 FFF2          BMI     DIVOV2          NEGATIVE DIVISOR IS A NO-NO
01748A FFC9 36                  PSHA                    PUSH IT
01749A FFCA 86 08     A         LDAA    #8              PUSH LOOP CTR
01750A FFCC 36                  PSHA
01751A FFCD 30                  TSX                     POINT X TO STACK
01752A FFCE A6 02     A         LDAA    2,X             RESTORE ORIGINAL DIVIDEND MSB
01753A FFD0 A1 01     A         CMPA    1,X             WILL QUOTIENT OVERFLOW? (ALS  DI
01754A FFD2 24 1C FFF0          BCC     DIVOVF          YES,GO SET V AND EXIT
01755A FFD4 58          DLOOP   ASLB                    SHIFT DIVIDEND-ANSWER LEFT
01756A FFD5 49                  ROLA
01757A FFD6 24 04 FFDC          BCC     DLZ             IS DIVIDEND MS BIT SET?
01758A FFD8 AB 01     A         ADDA    1,X             YES,ADD DIVISOR TO MSB
```

```
1759A FFDA 20 03 FFDF          BRA     DL3
1760A FFDC A0 01     A DL2     SUBA    1,X       NO,SUB DIVISOR FROM MSB
1761A FFDE 5C                  INCB              SET BIT IN RESULT
1762A FFDF 6A 00     A DL3     DEC     0,X       DONE?
1763A FFE1 26 F1 FFD4          BNE     DLOOP     NO
1764A FFE3 0D                  SEC               SHIFT A 1 IN LS BIT OF QUOTIENT
1765A FFE4 59                  ROLB              CASE THATS OKAY. CORRECT LATTER
1766A FFE5 4D                  TSTA              IS REMAINDER NEGATIVE?
1767A FFE6 2A 04 FFEC          BPL     DL4       NO,EVERYTHING'S OKAY
1768A FFE8 5A                  DECB              RESET QUOTIENT LS BIT
1769A FFE9 AB 01     A         ADDA    1,X       ADD DIVISOR TO GET + REMAINDER
1770A FFEB 0A                  CLV               INSURE V IS CLEARED
1771A FFEC 31          DL4     INS               CLEAN UP THE STACK
1772A FFED 31                  INS
1773A FFEE 31                  INS
1774A FFEF 39                  RTS               RETURN
1775                   *
1776A FFF0 31          DIVOVF  INS               CLEAN UP STACK
1777A FFF1 31                  INS
1778A FFF2 31          DIVOV2  INS
1779A FFF3 0B                  SEV               SET THE OVERFLOW FLAG
1780A FFF4 39                  RTS
1781                   *
1782                   * INTERRUPT VECTORS
1783                   *
1784A FFF8                     ORG     BASORG+$7F8
1785A FFF8     F800  A         FDB     IO
1786A FFFA     F80A  A         FDB     SFEI
1787A FFFC     F805  A         FDB     POWDWN
1788A FFFE     F97B  A         FDB     START
```

```
01790                    *
01791                    *
01792                    * RAM - LOCATIONS DEVOTED TO VARIABLE INFORMATION
01793                    *
01794                    *
01795A A000                        ORG     $A000     START OF RAM .
01796                    *
01797          0008   A NBRBPT EQU     8         # OF BREAKPOINTS SUPPORTED
01798                    *
01799                    * THE FOLLOWING ARE INITIALIZED AT START
01800                    *
01801A A000    0002   A IOV     RMB     2         I/O INTERRUPT POINTER
****ERROR   236
01802A A002    0002   A BEGA    RMB     2         BEGIN ADDRESS PRINT/PUNCH
01803A A004    0002   A ENDA    RMB     2         END ADDRESS PRINT/PUNCH
01804A A006    0002   A NIO     RMB     2         NMI INTERRUPT POINTER
01805A A008    0002   A SP      RMB     2         USER STACK POINTER
01806A A00A    0002   A SWI1    RMB     2         LEVEL 1 SWI VECTOR
01807A A00C    0002   A SWI2    RMB     2         LEVEL 2 SWI VECTOR
01808A A00E    0008   A BRINS   RMB     8         STORAGE FOR CONDITIONAL BRANCH
                                                       ROUTINE
01809                    *
01810          A016   A BRANEN EQU     *         END OF BRANCH ROUTINE + 1
01811                    *
01812                    * THE FOLLOWING ARE INITIALIZED TO ZERO AT START
01813                    *
01814                    *
01815A A016    0001   A OUTSW   RMB     1         OUTPUT SWITCH
                                                    (ZERO => ECHO INPUT)
01816                    *
01817A A017    0002   A TRCADR RMB     2         TRACE ADDRESS
01818A A019    0001   A TRCINS RMB     1         OP CODE REPLACED BY SWI
01819A A01A    0002   A NTRACE RMB     2         NO. OF INSTRUCTIONS TO TRACE
01820                    *
01821A A01C    0010   A BRKADR RMB     NBRBPT*2  BREAKPOINT ADDRESS TABLE
01822A A02C    0010   A BRKINS RMB     NBRBPT*2  OP CODES FOR BREAK REPLACEMENT
01823                    *                             (UPPER BYTE OF EACH
01824                    *                              PAIR USED ONLY)
01825                    *
01826          A03C   A CKSM    EQU     *         CHECKSUM
01827          A03C   A ASAVE   EQU     *         A REG SAVE
01828          A03C   A TEMP    EQU     *         CHAR COUNT(INADD)
01829A A03C    0001   A         RMB     1
01830                    *
01831                    *
01832          A03D   A BYTECT  EQU     *         BYTE COUNT
01833          A03D   A MCONT   EQU     *         TEMP
01834A A03D    0001   A         RMB     1
01835                    *
01836                    *
01837A A03E    0001   A XHI     RMB     1         X REG HIGH (TEMP)
01838A A03F    0001   A XLOW    RMB     1         X REG LOW
01839                    *
01840                    *
01841          A040   A SSAVE   EQU     *         S REG SAVE
01842          A040   A TW      EQU     *         TEMP DOUBLE BYTE
01843A A040    0002   A         RMB     2
01844                    *
01845                    *
01846A A042    0001   A BRKSIN RMB     1         1=>BREAKS ARE IN USER PROGRAM
```

```
01847                       *
01848A A043     0001   A BRKTRC RMB     1          1=>P-COUNTER IS AT BREAKPOINT
01849                       *                      AND USER WANTS TO CONTINUE-
01850                       *                      ONE TRACE WILL BE DONE AND
01851                       *                      BREAKPOINT RESTORED
01852           A080   A ENDING EQU     $A080      END OF VAR'S INTZ'D TO 0
01853                       *
01854A A044     0001   A ACIAT  RMB     1          SAVE ACIA CONTROL REG FOR
01855                       *                      CONTROL LOOP INITIALIZE.
01856                       *  EXORTAPE RAM
01857                       *AUXILIARY REGISTER STORAGE
01858A A045     0001   A FIDH   RMB     1
01859A A046     0001   A FIDL   RMB     1
01860A A047     0001   A STARTP RMB     1
01861A A048     0001   A STOPPG RMB     1
01862A A049     0001   A TOTCNT RMB     1          WINDOW WIDTH
01863A A04A     0001   A PATDEL RMB     1          PATTERN-ELEMENT LENGTH
01864A A04B     0001   A CL     RMB     1          CHECK/LOAD SUB
01865A A04C     0001   A CLL    RMB     1
01866A A04D     0001   A CLLL   RMB     1          RTS
01867                       *TAPE TEMPORARIES
01868A A04E     0001   A H      RMB     1          LOAD IX, ETC.
01870A A04F     0001   A L      RMB     1
01871A A050     0001   A Q      RMB     1          Q HOLDS LAST BYTE RECOVERED
01872A A051     0001   A R      RMB     1          R,S IS CRC REGISTER
01873A A052     0001   A S      RMB     1
01874A A053     0001   A V      RMB     1          PAGE COUNT
01875           A04B   A T1     EQU     CL
01876                       *
01877                       * REST OF 128 RAM IS USED FOR STACK
01878                       *
01879A A054     0024   A        RMB     36
01880A A078     0001   A STACK  RMB     1          START OF STACK AREA
01881                       END
TOTAL ERRORS 00001
```

```
8009 ACIAD   00031*00272 00280
8008 ACIAS   00030*00197 00298 00277 00319 00326 00347
A044 ACIAT   00195 00324 00345 01854*
F833 ADRSTR  00089*00293
A03C ASAVE   00505 00511 00533 00552 00575 01827*
F847 BADDR   00101*00150 00155 00217 00239 00400 01164 01172
FA08 BADDRJ  00400*00411 00424 00432 00453
F800 BASORG  00035*01784
A002 BEGA    00151 00659 01802*
FA85 BKCON1  00533*00585
FA82 BKCON2  00544*00570 00578 00582 00600 00603 00607
FA54 BKCON3  00546*00619
FAA3 BKDONE  00521 00553*
FAA0 BKPUT   00553 00555*
A016 BRANEN  00299 01810*
F844 BRG     00094 00095*
FBEB BRG1    00817*00858 00876
FBED BRG2    00813*00833 00881 00887 00893
FBE2 BRG0    00096 00812*00863
FBFD BRGODC  00816 00830*00832
```



```
A00E BRINS   00787 00801 01803*
FA82 BRK2    00526*
A01C BRKADR  00507 01821*
FBA9 BRKINH  00952 00733 00750*
A02C BRKINS  00546 01822*
FA73 BRKLP   00511*00547
A042 BRKSIN  00515 00555 01846*
FA6A BRKSUB  00387 00406 00426 00435 00503*00715 00771
A043 BRKTRC  00446 00464 00763 00769 01848*
FBDE BRNOGO  00055 00805*
FA0D BSRBRK  00405*00414 00420
FEAC BY1     01414*01430
FEB2 BY2     01412 01413*
FEB4 BY3     01420*01427
FEBB BY4     01423*01424
FEC6 BY5     01431*
F855 BYTE    00101 00103 00109*00213 00219 01132
F857 BYTE2   00110*00250
A03D BYTECT  00215 00220 01832*
FEA4 BYTEOU  01407*01571
FF72 BYTEOV  01475 01571*
FEF8 BYTOV1  01443 01444 01445 01448 01463 01464 01467 01469 01471 01475
             01476*01487 01488 01500 01502 01506 01514 01526 01528
F91A C1      00169 00173 00175 00209 00234*
FD98 C12     01187 01189*
F928 CHA1    00244*00254
F922 CHANG   00241*00265
F91D CHANGE  00067 00239*
FD92 CHECK   01177 01186*
FC0B CKOBRA  00790 00807 00847*
A03C CKSM    00118 00119 00212 01826*
A04B CL      01189 01237 01864*01875
A04C CLL     01190 01865*
A04D CLLL    01192 01866*
FACF CLRBRK  00537 00605*
F9D3 CNTA    00356 00358*
FA57 CNTRL2  00407 00477*
FA54 CONT    00059 00464*
```

```
   F9AE CONT2   00329 00331*
   FBC2 CONTRC  00455 00786*
   F9BA CONTRL  00234 00333*00371 00477 00752
   FE7E CRC1    01377*
   FE80 CRC2    01379*01408 01415
   FE90 CRC3    01384 01387*
     E8 CRCK    01250*
   FDF8 CRCK1   01254 01257*
   FDFA CRCK2   01256 01258*01273 01285
   F9A3 CTRL    00702 00752*00782
   FA0B DELBRK  00061 00405*
   FFC4 DIV     01745*
   FFF2 DIVOV2  01747 01778*
   FFF0 DIVOVF  01754 01775*
   FFDC DL2     01757 01760*
   FFDF DL3     01753 01762*
   FFEC DL4     01767 01771*
   FFD4 DLOOP   01755*01763
   FF7E DSETUP  01532 01575*
   FDD8 DUM1    01235*01240
PAGE  043   MIKBUG 2.0 WITH AUDIO CASSETTE

   FF14 DUMP1   01505*01509
   FF03 DUMPAG  01488*01544
   FDCD DUMPR   01211 01231*
   A004 ENDA    00156 00561 00663 00697 01803*
   A080 ENDIN0  00307 01852*
   F9A4 ENT1    00203 00327*01110
   F8E3 ENTER   00203*
   0004 EOT     01076*01084 01086 01088 01090 01092
   000A ERR     01118*01150 01180
   001B ESC     01067*01332 01547
   FD0F EXOR    01109 01124*
   FD05 EXORT   00081 01109*
   F80F FCTABL  00055*00363
   F839 FCTBEN  00084*00363
   A045 FIDH    01165 01275 01468 01858*
   A046 FIDL    01282 01470 01859*
   FAC1 FNDRPL  00534 00538*
   FE02 G1      01272*01280 01283
   FE25 G2      01236 01275 01288*01308 01331
   FDBC GET1    01208 01214*
   FDAA GET2    01205*01210 01213
   FDC7 GET3    01215 01219*
   FDCC GET4    01204 01206 01212 01215 01221*
   FDC9 GET5    01218 01220*
   F885 GETADD  00147*00647
   FE06 GETFIL  01203 01274*01277
   FDA3 GETLOA  01202*
   F9E7 GOODCH  00365 00374*
   FA25 GOTO    00063 00432*
   A04E H       01232 01234 01303 01306 01308 01501 01504 01541 01542 01543
                01550 01863*
   FDB2 HEADER  01463*01533
   F8BE IN1HG   00171 00177*
   F878 INCH    00136*00157 00201 00207 01155
   F966 INCH1   00136 00244 00277*00279 00283 00330 00354
   F9AC INCH2   00330*00392
   F99B INCNOV  00739 00742*
   FC03 INCX    00817 00837*00313
```

```
FC0A INCXR   00838 00842*
F8AA INHEX   00109 00115 00167*
F8AC INHEX2  00168*00249
F831 INILP1  00296*00300
F83A INILP2  00305*00308
FC06 INXLP   00839*00841
F83C INZ     00321*00334
F83E INZ1    00324*00335
F800 IO      00033*01785
A000 IOV     00033 01801*
FC35 JMPEXT  00852 00857 00880*
FC2E JMPIDX  00850 00854 00873*
A04F L       01233 01239 01503 01508 01524 01527 01870*
FDFD LDV     01231 01235 01250 01251 01250*01278 01281 01286
F840 LF      00246 00255*
FAA7 LN      00513 00565*
F8D0 LOAD    00065 00189*
FE4F LOAD1   01330*01333
F902 LOAD11  00219*00226
F913 LOAD15  00221 00230*
AGE  050   MIKBUG 2.0 WITH AUDIO CASSETTE

F916 LOAD19  00224 00232*00253
FD36 LOAD2   01173 01185 01188*
F8DF LOAD3   00201*00206 00211 00231
F8E6 LOAD4   00202 00204*
FE60 LOADB2  01350*01353
FE5A LOADB3  01355*01358
FE71 LOADB4  01359*01362
FE7B LOADB5  01347 01351 01355 01360 01364*
FE53 LOADB6  01346*01349
FE58 LOADBI  01307 01330 01345*
FF6F LOADBV  01260 01570*
FE4A LOADBY  01328*01570
FD90 LOADV   01185*
FF7B LSETUP  01202 01574*
FF44 MAST1   01543*01551
FF5C MAST2   01553*
FF5A MAST3   01548 01552*
FF37 MASTER  01175 01538*
FC7D MCL     00261 00371*
FC86 MCL1    00161 00372*
FC8D MCL2    00331 00373*
FC98 MCL3    00474 00375*
FCAE MCL4    00148 00377*
FC89 MCL5    00153 00379*
FC7C MCLOFF  00351 00370*
A03D MCONT   09670 00678 01833*
FC71 MEOF    00700 00968*
FF38 ML1     01635*01641
FFA0 ML2     01637 01640*
FCC4 MSG1    01083*01123
FCD1 MSG2    01085*01153
FCDB MSG3    01087*01162
FCE5 MSG4    01089*01170
FCF7 MSG5    01091*01139
FAF9 MTAPE1  00643*00674
FF32 MUL     01630*01631
FBBF NBKTRC  00765 00778*
0003 NBRBFT  00526 00589 01797*01821 01822
```

```
A006 NIO     00044 00334 01804*
F9E3 NMI     00333 00381*
FAB5 NOBRIN  00516 00584*
A01A NTRACE  00442 00779 00781 01819*
F3D9 NXTCHR  00364*00370
F  'F OPBT3  00931 00933 00937*
FC=0 OPBTRT  00922 00935 00939*
FC61 OPBTTB  00918 00947*
FC43 OPCBYT  00782 00811*
FAD5 OT2HS   00510*00528 00629 00630
FAD8 OT4HS   00511*00618 00631 00632 00634
F9D1 OUT2    00357*00384
F83F OUT2H   00175*00185 00186 00706
F8C1 OUT2HA  00180*
F8CA OUT2HS  00185*00242 00510
F8C8 OUT4HS  00185*00264 00611
F95A OUTC1   00268*00271
F875 OUTCH   00131 00135*00138 00188 00191 00543 00555 01119 01220
F959 OUTCH1  00135 00233 00258 00257*00285 00357
AGE  051    MIKBUG 2.0 WITH AUDIO CASSETTE

F857 OUTHL   00122*00180
F85B OUTHR   00128*00183
F8CC OUTS    00187*00240 00358 00385
A016 OUTSW   00199 00284 00349 01815*
A04A PATDEL  01422 01853*
F89E PCRLF   00147 00152 00160*00328 00382 00418 00473 00525 00599 01099
             01156 01174
FD02 PDATIP  01100*01171
i  'F PDATA  01099*01123 01131 01154 01163
F87E PDATA1  00140*00143 00154 00162 00252 00332 00352 00475 00575 00701
             01100
F872 PDATA2  00138*00142
FA16 PNTBRK  00057 00418*00427
FB0C PNULL   00554*00557
F805 POWDWN  00044*01787
FEC8 PREAMB  01443*01465 01485 01498
FAE2 PRINT   00476 00625*00751 00778
FADB PRNTBK  00540 00617*
FA53 PSTAK   00476*
FA5C PSTAK1  00073 00388 00473*
FB19 PUN11   00661*00688
FB23 PUN22   00655 00658*
FB2D PUN23   00657 00663*
FB4D PUN32   00636*00688
FB02 PUNCH   00071 00545*
FB71 PUNT2   00579 00582 00683 00686 00693 00705*
A050 Q       01304 01305 01310 01323 01332 01363 01364 01383 01871*
A051 R       01252 01315 01380 01387 01443 01525 01872*
FA11 RSTBRK  00073 00411*
FC33 RTISIM  00859 00885*
FC3E RTSSIM  00881 00891*
A052 S       01253 01316 01381 01388 01450 01523 01873*
 27 SENCRC   01510 01523*
FA1D SETBRK  00083 00424*
FF5D SETUP   01553*01574 01575
F80A SFEI    00043*01786
FFB1 SML1    01684 01687*
FFB7 SML2    01688 01691*
```

```
FFC2 SML3   01694 01698*
FFA7 SMUL   01680*
FD15 SOFT   01127*
A008 SP     00294 00327 00343 00381 00450 00625 00633 00712 00825 01205*
F9FE SPD    00075 00392*
A040 SSAVE  00504 00553 00617 01841*
FE2C STA1   01305*01312 01314
FE44 STA2   01315*
A078 STACK  00090 01880*
F97B START  00292*01788
FE25 STARTF 01261 01274 01303*
A047 STARTP 01173 01473 01540 01860*
FE00 STARTV 01205 01261*
A048 STOPPG 01472 01549 01861*
003F SWI    00032*00530 00732 00823
A00A SWI1   00049 01806*
FB76 SWI1S  00091 00711*
FB85 SWI1S1 00721 00723*
A00C SWI2   00742 01807*
A04B T1     01161 01167 01875*
AGE  052   MIKBUG 2.0 WITH AUDIO CASSETTE

8005 TACON  01065*01555 01558
FF59 TAIN1  01394 01568*
FEA1 TAIN1V 01348 01352 01357 01361 01394*
FF81 TAIN2  01568 01578*
FF6C TAOU1  01420 01569*
FF8E TAOU2  01569 01588*
A03C TEMP   00672 00687 01828*
FD18 TI1    01129*
FD53 TI2    01154*
FD1A TI2A   01130*01157
FD1D TI3    01131*
FD62 TI4    01163*
FD77 TI5    01171*
FD2B TIN1   01134 01137*
FD34 TIN2   01138 01141*
FD3D TIN3   01142 01145*
FD46 TIN4   01146 01149*
FD4D TIN5   01135 01140 01144 01148 01152*
FD5C TIN6   01161*
FD85 TIN7   01163 01176*
FD50 TIN8   01153*
FD0F TINS   01125*
FD15 TINSS  01128*
A049 TOTCNT 01126 01152 01354 01862*
8004 TP     01054*01557 01578 01588
FA50 TRACE  00077 00459*
FA3A TRACE2 00446*00460
FA3D TRACE3 00448*00466
FEFB TRAILR 01485*01552
A017 TRCADR 00452 00728 00772 00773 00812 00813 00847 01817*
FBA6 TRCINH 00729 00760*
A019 TRCINS 00454 00760 00786 00822 01818*
8003 TTY    01063*01545 01582 01585
8008 TTYCON 01062*
FF75 TTYIN1 01390 01572*
FF86 TTYIN2 01572 01582*
FF78 TTYO1  01258 01512 01573*
FF8A TTYO2  01573 01585*
```

```
A020 TW      00520 00582 00584 00681 00585 00589 00595 01842*
F343 UA      00248 00257*
F348 UA1     00255 00260*
A053 V       01214 01241 01287 01874*
A03E XHI     00102 00105 00160 00193 00241 00260 00263 00437 00572 00599
             01837*
F XLOW       00104 00433 00502 01838*
```