NewBear Computing Store Ltd

HEAD OFFICE: 40 BARTHOLOMEW STREET NEWBURY BERKSHIRE RG14 5LL NEWBURY (0635) 30505

```
**********  *******    *    *   *******
    *       *      *   *    *  *
    *       *      *   *    *  *
    *       *******    *    *  *     ***
    *       *      *   *    *  *       *
    *       *      *   *    *  *       *
    *       *******     ****    ******
```

TBUG MONITOR SOFTWARE PACKAGE.

THIS MONITOR HAS THE FOLLOWING FEATURES...

- LIVES AT E000 - E3FF
- MON2 IS COMPATIBLE WITH 77-68VDU IF TBUG IS USED
- MIKBUG COMPATIBLE DUMP/LOAD FORMATS
- MIKBUG COMPATIBLE SUBROUTINES AT CORRECT ADDRESSES.
- DRIVES 77-68 MEMORY MAPPED VDU. FULL SCROLLING 40X24 DISPLAY
- INPUT/OUTPUT ON SERIAL PORT (ACIA @8000) ALSO INPUT ON TWO 8BIT PARALELL PORTS (PIA @ 8010)
- CASSETTE FILE HANDLING
- 12 COMMANDS INCLUDING ALL 5 MIKBUG COMMANDS
- GRAPHICS ON 77-68 VDU MADE EASY WITH 'PLOT' SUBROUTINE
- COMMERCIAL SOFTWARE (EG BASIC, GAMES ETC) RUNS WITHOUT MODIFICATION
- SO NOW THE ADVANTAGES OF A MEMORY MAPPED VDU NEED NOT BE FORFEITED TO USE A MIKBUG COMPATIBLE MONITOR.
- SINGLE STEP TRACE WITH UNMODIFIED HARDWARE ON MON-2 BOARD.

INTRODUCTION.

TBUG IS A SOFTWARE PACKAGE AVAILABLE IN EPROM FOR THE MON2 BOARD
TBUG HAS BEEN WRITTEN FOR THE 77-68 SYSTEM WITH MON2 AND THE
77-68 MEMORY MAPPED VDU. SO MON2 IS COMPATIBLE WITH THE 77-68
VDU IF TBUG IS USED AS THE MONITOR. TBUG ACCEPTS INPUT FROM EITHER
OF THREE PORTS, ONE SERIAL (AN ACIA AT 8000(INTENDED FOR
CASSETTE/TELETYPE)) AND TWO PARALELL PORTS (A PIA AT 8010(INTENDED FOR
PAPER TAPE READER AND KEYBOARD)). TBUG OUTPUTS TO THE SERIAL PORT
AND TO THE VDU PROVIDING A SCROLLING DISPLAY OF THE FULL 40 X 24.
IT ENABLES THE USER TO GET AWAY FROM SWITCHES AND LED'S AND TO TALK
TO HIS COMPUTER VIA AN ASCII KEYBOARD. TBUG ENABLES
PROGRAMS TO BE ENTERED, RUN, DEBUGGED AND DUMPED/LOADED TO/FROM
CASSETTE. OTHER FUNCTIONS ARE ALSO PROVIDED.
TBUG CONTAINS ALL OF THE COMMONLY USED MIKBUG SUBROUTINES AT THEIR
CORRECT ADDRESSES SO THAT COMMERCIAL SOFTWARE CAN AND DOES!
RUN FIRST TIME.

ROM MONITOR PARTS LIST FOR TBUG.

X1	74LS08
X2, 5, 13, 14	DM81LS97
X4	6820
X7	TBUG EPROM
X15	6850
X6, 9	74LS04
X10, 12	74LS10
X11, 26	74LS127 OR 74LS27
X16, 22, 23, 27	74LS00
X25, 21, 17	74LS02
X28	74LS163
X30, 29	74LS74
X24	MC14536
X20	MON2 ROM

CAPACITORS, RESISTORS, SWITCHES: AS IN MON2 DESIGN NOTE.

SOCKETS 1 X 40 PIN
2 X 24 PIN
1 X 16 PIN

NOTE: SOME POINTS CONCERNING TBUG AND THE MON2 BOARD
CONTRADICT THOSE IN THIS PUBLICATION IN SUCH CASES
THIS PUBLICATION IS CORRECT.

CONSTRUCTION.

FOLLOW SCHEDULE SHOWN IN MON2 DESIGN NOTE INCLUDING ALL LINKS ETC.
BUT BUT OMMITTING COMPONENTS NOT SHOWN IN THE ABOVE PARTS LIST.
NOTE THAT SWITCH X31 SHOULD BE ORIENTATED SO THAT SW8 IS
IS ADJACENT TO R16.
FOLLOW TEST PROCEDURES SHOWN IN MON2 DESIGN NOTE UP TO AND INCLUDING
THOSE ON PAGE 8.
THEN MAKE THE FOLLOWING MODIFICATIONS TO THE BOARD.....
REMOVE LINK CONNECTED TO X17 PIN 10
REMOVE LINK CONNECTED TO X9 PIN 11
CONNECT X9 PIN 11 TO X17 PIN 10.
THE EPROM IS NOW LOCATED AT E000 INSTEAD OF E400
CONNECT X24 PIN 13 TO X15 PIN 4.
THE NEXT MOD CONCERNS THE VDU BOARD..
DUE TO A 'FUNNY' ON THE 8212 CHIP (X24 VDU) THIS HAS NOT BEEN
USED AS KEYBOARD PORT BY TBUG SO WHEN USING TBUG EITHER REMOVE
THIS CHIP OR BREAK THE CONNECTION TO PIN 23 OF IT.
BACK TO MON2 NOW..
CONNECT X24 PIN 13 TO X15 PIN 4. MAKE THIS CONNECTION ON THE
BACK OF THE BOARD.
IN THE POSITION WHERE X8 IS SHOWN (NOT USED WITH TBUG) LINK WHAT
WOULD HAVE BEEN X8 PIN 10 TO X8 PIN 1 AND X8 PIN 12 TO X8 PIN 20.
CONNECT KEYBOARD TO PARALELL PORT.

X31 (8 WAY DIL SWITCH)
SET SW1 AND SW2 OFF (SO AS TO SET UP CORRECT ROM QUADRANT FOR TBUG).
BAUD RATES ARE NOW SELECTED WITH SWITCHES IN POSITIONS SHOWN BELOW.

BAUD RATE	SW3	SW4	SW5	SW6	SW7	SW8	SET FOR	HZ AT 'TP'.
110	ON		ON	ON			R18	1578
300		ON	ON		ON		R17	4800
1200	ON	ON	ON			ON	R16	12900

AFTER RESET TBUG OUTPUTS THE WORD 'TBUG' (EQUIVALENT IN HUMAN TERMS
TO A CHEERY WAVE !) FOLLOWED BY A LINE FEED CARRIAGE RETURN
AND AN ASTERISK WHICH INVITES THE USER TO ENTER A COMMAND.

IF A SCOPE OR FREQUENCY COUNTER IS NOT AVAILABLE FOR SETTING
R16 - R18. THEN SET R18 SO THAT A 256 BYTE DUMP (P COMMAND) TAKES
1 MIN WITH SW3 - SW8 SET FOR 110 BAUD. SET R17 SO THAT
A 1KBYTE DUMP TAKES 1 MIN AND 44 SECS WITH SW3 - SW8 SET FOR 300
BAUD. SET R16 SO THAT A 1KBYTE TAKES 24 SECS WITH SW3 - SW8 SET
FOR 1200 BAUD.
NOTE THAT I/O ON AC1A IS TTL LOGIC LEVEL NOT
V24 OR 20MA.

TBUG COMMANDS.
FOLLOWING IS A LIST OF COMMANDS AVAILABLE TO THE USER THE COMMANDS ARE ISSUED VIA AN INPUT DEVICE EG KEYBOARD. NOTE THAT COMMANDS CAN BE INPUT THROUGH EITHER OF TBUGS THREE PORTS.

- M - MEMORY INSPECT/MODIFY
- R - DISPLAY CONTENTS OF STACK
- G - EXECUTE RTI INSTRUCTION
- J - JUMP TO USERS PROGRAM
- P - DUMP AN AREA OF MEMORY IN 'S1' FORMAT
- L - LOAD DATA IN 'S1' FORMAT INTO MEMORY
- T - CALCULATE BRANCH OFFSET
- K - OUTPUT THE CONTENTS OF A BLOCK OF MEMORY IN HEX
- U - CHANGE FORMAT OF SERIAL PORT
- Y - COPY A BLOCK OF MEMORY TO ANOTHER BLOCK
- V - OUTPUT A BLOCK OF MEMORY WITH NAME ATTACHED
- I - INPUT A FILE FROM EXTERNAL DEVICE (CASSETTE)

M MEMORY INSPECT/MODIFY
AFTER THE M ENTER THE ADDRESS OF THE FIRST BYTE TO BE OPERATED UPON. TBUG WILL DISPLAY THE CONTENTS OF THIS LOCATION. IF YOU WISH TO CHANGE THIS LOCATIONS CONTENTS THEN ENTER A SPACE FOLLOWED BY THE HEX NUMBER TO BE PLACED IN THIS LOCATION. IF YOU DO NOT WISH TO ALTER THE CONTENTS OF THIS LOCATION THEN HIT THE RETURN KEY. IN BOTH CASES THE ADDRESS AND CONTENTS OF THE NEXT LOCATION WILL BE DISPLAYED. TO EXIT FROM THE 'M' FUNCTION HIT THE SPACE KEY TWICE.

R REGISTER PRINT
THIS COMMAND DISPLAYS THE CONTENTS OF THE INTERNAL REGISTERS WHICH HAVE BEEN STORED ON THE STACK. CCR, ACC B, ACC A, INDEX(HIGH), INDEX(LOW), PC(HIGH), PC(LOW), SP(HIGH) AND SP(LOW).

G GO TO USERS PROGRAM (EXECUTE RTI)
THIS COMMAND LOADS THE INTERNAL REGISTERS OF THE MICROPROCESSOR WITH THE CONTENTS OF THE STACK. SO BY CHANGING THE DATA ON THE STACK VALUES MAY BE SET IN REGISTERS BEFORE ENTERING ROUTINES TO TEST THE ROUTINES. IF THE STACK POINTERS CONTENTS ARE NOT ALTERED THE AREAS OF MEMORY CONCERNED WITH THIS INSTRUCTION LIE BETWEEN A043 AND A049 AND ARE IN THIS ORDER CCR, ACC B, ACC A, INDEX(HIGH), INDEX(LOW), PC(HIGH) AND PC(LOW).

J JUMP
THIS COMMAND IS ALSO USED FOR RUNNING A USERS PROGRAM. BUT IS EASIER TO USE. ENTER J FOLLOWED BY THE ADDRESS TO WHICH YOU WISH TO PASS CONTROL.

P OUTPUT AN AREA OF MEMORY IN 'S1' FORMAT
THE START ADDRESS OF THE AREA TO BE OUTPUT MUST BE PRESENT IN MEMORY LOCATIONS A002 (MSB) AND A003 (LSB). THE END ADDRESS OF AREA TO BE OUTPUT MUST BE PRESENT IN MEMORY LOCATIONS A004 (MSB) AND A005 (LSB) BEFORE THE P COMMAND IS ISSUED.
THE COMMAND IS INTENDED FOR DUMPING MEMORY TO CASSETTE OR PAPER -TAPE PUNCH. THE 'S1' FORMAT IS USED BY OTHER 6800 MONITORS LIKE SWTBUG AND MIKBUG AND SO DUMPS MADE IN THIS FORMAT ARE PORTABLE. THE OUTPUT IS IN ASCII AND CONSISTS OF: A START OF RECORD HEADER, BYTE COUNT, ADDRESS, BYTES OF DATA AND A CHECKSUM (SAME FUNCTION AS A PARITY BIT). THE DUMP WILL BE TERMINATED WITH AN END OF FILE MARKER 'S9'. IF A CASSETTE IS TO BE USED IT SHOULD BE SET TO RECORD JUST BEFORE ENTERING P. ALSO THE BAUD RATE WILL HAVE TO BE SET FOR 300. WHEN THE DUMP HAS BEEN COMPLETED THEN AN ASTERISK WILL BE OUTPUT.

L LOAD

THIS COMMAND IS USED TO LOAD DATA IN 'S1' FORMAT FROM CASSETTE/ PAPERTAPE INTO MEMORY. IF DURING THE LOAD A CHECK SUM ERROR IS FOUND THEN A QUESTION MARK WILL BE OUTPUT. THE TAPE SHOULD THEN BE REWOUND FOR A FRACTION AND L ENTERED AGAIN. THERE IS NO NEED TO REWIND TO THE BEGINNING OF THE FILE.
WHEN THE LOAD IS COMPLETE AN ASTERISK WILL BE OUTPUT AND THE CASSETTE/PAPERTAPE READER SHOULD BE SWITCHED OFF.

T BRANCH CALCULATOR

WHEN YOU ARE WRITING A PROGRAM TBUG WILL CALCULATE YOUR BRANCH OFFSETS FOR YOU. FOR EXAMPLE SAY WE WISH TO BRANCH FROM E0F9 TO E0FF ENTER T, START ADDRESS (IN THIS CASE E0F9) AND DESTINATION ADDRESS (IN THIS CASE E0FF) TBUG WILL OUTPUT THE OFFSET FOR YOU AS A HEX NUMBER. IF THE BRANCH IS OUT OF RANGE A QUESTION MARK WILL BE OUTPUT.

K BLOCK TABULATE

THE K COMMAND CAUSES THE CONTENTS OF A BLOCK OF MEMORY TO BE DISPLAYED ON THE VDU IN HEX. THE START ADDRESS OF THIS BLOCK IS GIVEN AS A PARAMETER OF THE COMMAND. EG. KE000 WILL DISPLAY THE CONTENTS OF SIXTY FOUR MEMORY LOCATIONS STARTING WITH E000.

U CHANGE SERIAL FORMAT

THE SERIAL PORT IS NORMALLY INITIALISED BY TBUG FOR 1 STOP BIT BUT IF A TELETYPE IS TO BE USED THEN 2 STOP BITS MUST BE TRANSMITTED AND RECOGNISED BY THE PORT. SIMPLY ISSUE THE U COMMAND AFTER RESET AND IT WILL BE DONE !

Y COPY

THE COPY COMMAND IS USED TO COPY AREAS OF MEMORY. THE START ADDRESS OF THE BLOCK TO BE COPIED MUST BE ENTERED THEN THE END ADDRESS OF THE BLOCK MUST BE ENTERED FOLLOWING THAT THE START ADDRESS OF THE AREA OF MEMORY TO WHICH THE FIRST AREA IS TO BE COPIED MUST BE ENTERED. WHEN COMPLETED AN ASTERISK WILL BE OUTPUT.

V SAVE

WITH THE V AND I COMMANDS DUMPS TO TAPE CAN BE REGARDED AS FILES EACH WITH A NAME CHOSEN BY THE USER ASSOCIATED WITH IT.
WHEN LOADING THEREFORE THE USER TYPES THE NAME OF THE FILE HE WISHES TO LOAD, TBUG WILL THEN WAIT UNTIL IT SEES THAT FILE AND WILL LOAD IT INTO MEMORY OTHER FILES WITH DIFFERENT NAMES WILL BE IGNORED.
USE OF THE SAVE COMMAND.
ENTER V FOLLOWED BY THE NAME YOU HAVE CHOSEN FOR THE FILE WHICH MUST BE NO LARGER THAN 10 CHARS LONG , THE COMMAND WILL BE EXECUTED WHEN THE RETURN KEY HAS BEEN DEPRESSED. FOUR CONTROL CHARS ARE THEN OUTPUT TO INDICATE THE START OF A NAME THE NAME IS THEN OUTPUT FOLLOWED BY AN 'S1' DUMP OF MEMORY CONTENTS. AS WITH THE P COMMAND THE ADDRESSES OF THE START OF THE AREA TO BE OUTPUT MUST BE ENTERED INTO MEMORY LOCATIONS A002 - A005. NOTE TBUG WILL REJECT RUDE NAMES !

I INPUT

THIS COMMAND IS USED FOR LOADING A SPECIFIC FILE INTO MEMORY FROM CASSETTE. ENTER I FOLLOWED BY THE NAME OF THE FILE YOU WISH TO LOAD INTO MEMORY, AGAIN NO MORE THAN TEN CHARS AND TERMINATED WITH RETURN. AS TBUG COMES ACROSS OTHER FILES ON THE TAPE IT WILL OUTPUT THEIR NAMES ONLY AND WILL OUTPUT 'END' WHEN IT DISCOVERS THE END OF A FILE. THIS ALLOWS THE USER TO MORE EASILY POSITION THE TAPE WHEN HE WISHES TO RECORD OVER ANOTHER FILE. WHEN THE FILE THAT TBUG IS LOOKING FOR HAS BEEN FOUND IT WILL BE LOADED. IF A CHECKSUM ERROR SHOULD OCCUR FOLLOW THE PROCEDURE DESCRIBED FOR THE L COMMAND. BY OUTPUTTING JUST THE FILE NAMES CAN BE OBTAINED BY ENTERING I THEN CR ONLY.

TBUG UTILITY ROUTINES.

SEVERAL SUBROUTINES ARE AVAILABLE IN THE TBUG MONITOR WHICH THE PROGRAMMER MAY TAKE ADVANTAGE OF WHEN WRITING HIS OWN SOFTWARE. A LIST OF THEM IS GIVEN BELOW ALONG WITH THEIR FUNCTION, ADDRESS, ENTRY REQUIREMENTS AND EXIT PARAMETERS. NOTE THAT ALL ARE SUBROUTINES AND, AS SUCH, MUST BE CALLED FOR BY A 'JSR' INSTRUCTION TO THE PROPER ADDRESS.
IN WHAT FOLLOWS, A, B, AND X STANDS RESPECTIVELY FOR THE ACCUMULATOR A, ACCUMULATOR B, AND THE INDEX REGISTER. M STANDS FOR A MEMORY LOCATION AND IS FOLLOWED BY THE ADDRESS OF THAT LOCATION.
IN ADDITION, THE DATA ARE REPRESENTED IN THE HEXADECIMAL NOTATION.

AUTO (E297)
FUNCTION: OUTPUT THIRTY TWO NULLS (OO).
ENTRY CONDITIONS: NONE
EXIT CONDITIONS: A AND B ARE DESTROYED.

BADDR (E047)
FUNCTION: BUILD A 16 BIT NUMBER FROM 4 HEX CHARS. ENTERED FROM EITHER OF THE INPUT PORTS.
ENTRY CONDITIONS: NONE
EXIT CONDITIONS: X CONTAINS THE 16 BIT NUMBER. M AOOC ALSO CONTAINS THE SAME 16 BIT NUMBER.

BDSP (E311)
FUNCTION: SAME AS FOR 'BADDR' BUT A SPACE IS OUTPUT AFTER THE FOUR HEX CHARS. HAVE BEEN INPUT.
ENTRY CONDITIONS: SAME AS FOR 'BADDR'.
EXIT CONDITIONS: SAME AS FOR 'BADDR'.

BYTE (E055)
FUNCTION: INPUT TWO HEX CHARS. FROM THE INPUT DEVICE AND FORM A ONE BYTE NUMBER.
ENTRY CONDITIONS: NONE
EXIT CONDITIONS: A CONTAINS THE 8 BIT NUMBER, B IS DESTROYED.

INPOLL (E1AC)
FUNCTION: INPUT ONE CHAR. TO A FROM EITHER OF THE THREE INPUT PORTS. THE MOST SIGNIFICANT BIT IS CLEARED.
ENTRY CONDITIONS: NONE
EXIT CONDITIONS: A CONTAINS THE CHAR.

INXBITS (E1AE)
FUNCTION: INPUT SELECTED BITS OF A CHAR. FROM EITHER OF THE THREE INPUT PORTS.
ENTRY CONDITIONS: BITS DESIRED ARE SET IN A.
EXIT CONDITIONS: DESIRED BITS OF THE INPUTTED CHAR ARE IN A. EG. IF BEFORE ENTERING THE ROUTINE A CONTAINED FF THEN ALL OF THE BITS OF THE INPUTTTED CHAR. WOULD BE PRESENT IN A ON LEAVING THE ROUTINE, USED FOR BINARY LOADERS.

OUT2H (E0BF)
FUNCTION: OUTPUT TWO HEX DIGITS OF HEX NUMBER POINTED TO BY X
ENTRY CONDITIONS: X CONTAINS THE ADDRESS OF THE HEX NUMBER.
EXIT CONDITIONS: A IS DESTROYED, X IS INCREMENTED.

OUT2HS (E0CA)
FUNCTION: OUTPUT TWO HEX CHARS. FOLLOWED BY A SPACE.
ENTRY CONDITIONS: X CONTAINS THE ADDRESS OF THE HEX NUMBER.
EXIT CONDITIONS: A IS DESTROYED, X IS INCREMENTED.

OUT4HS (E0C8)
FUNCTION: OUTPUT FOUR HEX DIGITS (TWO BYTES) PLUS A SPACE.
ENTRY CONDITIONS: X CONTAINS ADDRESS OF FIRST BYTE.
EXIT CONDITIONS: A IS DESTROYED, X CONTAINS ADDRESS OF SECOND BYTE.

TBUG UTILITY ROUTINES CONTINUED.

OUTE (E1D1)
FUNCTION: OUTPUT ONE ASCII CHAR. TO VDU AND SERIAL PORT.
ENTRY CONDITIONS: A CONTAINS ASCII CHAR. TO OUTPUT.
EXIT CONDITIONS: NO CHANGE.

OUTS (E0CC)
FUNCTION: OUTPUT A SPACE.
ENTRY CONDITIONS: NONE.
EXIT CONDITIONS: A IS DESTROYED.

OUTHL (E067)
FUNCTION: OUTPUT RIGHT DIGIT OF HEX NUMBER.
ENTRY CONDITIONS: A CONTAINS THE HEX NUMBER.
EXIT CONDITIONS: A IS DESTROYED.

OUTHR(E06B)
FUNCTION: OUTPUT LEFT DIGIT OF HEX NUMBER.
ENTRY CONDITIONS: A CONTAINS THE HEX NUMBER.
EXIT CONDITIONS: A IS DESTROYED.

PDATA (E07E)
FUNCTION: PRINT AN ASCII DATA STRING POINTED TO BY X. THE STRING MUST BE TERMINATED WITH THE ASCII EOT CHAR. (04).
ENTRY CONDITIONS: X CONTAINS THE ADDRESS OF THE FIRST BYTE IN THE STRING. THE STRING IS TERMINATED WITH EOT.
EXIT CONDITIONS: A IS DESTROYED. X CONTAINS THE ADDRESS OF THE EOT CHAR.

PLOT (E220)
FUNCTION: DISPLAY A CHAR. ON THE VDU IN POSITION DESCRIBED BY TWO BYTES (COLUMN AND ROW).
ENTRY CONDITIONS: X MUST CONTAIN A080, A CONTAINS THE CHAR. TO BE DISPLAYED, M A082 CONTAINS A BYTE DESCRIBING THE COLUMN, M A083 CONTAINS A BYTE DESCRIBING THE ROW. (00 00) = TOP LEFT CORNER.
EXIT CONDITIONS: X CONTAINS A080, B CONTAINS THE CHAR. THAT WAS AT THE LOCATION SPECIFIED BY THE COL. AND ROW COORDORDINATES. A IS UNALTERED.
NOTE THAT THE DISPLAY WILL SCROLL IF THE COLUMN BYTE IS GREATER THAN 27.

SCROLL(E264)
FUNCTION: SCROLL THE VDU DISPLAY UP ONE LINE.
ENTRY CONDITIONS: NONE.
EXIT CONDITIONS: X IS DESTROYED.

INTERUPTS (NMI, IRQ)
THE INTERUPT VECTORS ARE AVAILABLE TO THE USER IN RAM.
THEIR LOCATIONS ARE: IRQ @ A000 AND NMI @A006.

OTHER FUNCTIONS OF TBUG.

BREAKPOINT (SWI) FUNCTION.

THE SOFTWARE INTERUPT FUNCTION MAY USED TO INSERT BREAKPOINTS IN THE USERS PROGRAM. TO ENTER A BREAKPOINT INTO A USERS PROGRAM, REPLACE A SELECTED INSTRUCTION WITH A SWI INSTRUCTION. ON DETECTING THIS SWI INSTRUCTION IN THE USERS PROGRAM, THE SYSTEM PEFORMS THE TBUG SOFTWARE INTERUPT ROUTINE, PRINTS THE CONTENTS OF THE MPU REGISTERS (AS IN R COMMAND) AND PASSES CONTROL TO TBUG

ENTERING A SWI BREAPOINT.

REPLACE THE FIRST BYTE OF THE SELECTED INSTRUCTION WITH A SWI (3F) INSTRUCTION USING THE M COMMAND AND MAKING A NOTE OF THE ORIGINAL INSTRUCTION. RUN THE PROGRAM USING THE G OR J COMMANDS AND ON EXECUTING THE SWI INSTRUCTION THE MPU REGISTER CONTENTS WILL BE PRINTED OUT AND CONTROL WILL RETURN TO TBUG.

REMOVING A SWI BREAKPOINT.

USING THE M COMMAND REPLACE THE SWI INSTRUCTION WITH THE ORIGINAL INSTRUCTION, PROGRAM EXECUTION MAY THEN CONTINUE UNALTERED IF THE G COMMAND IS USED

WAIT AFTER LOAD OR INPUT.

WHEN A LOAD OR INPUT HAS BEEN COMPLETED OTHER DATA ON THE TAPE IS LOOKED UPON BY TBUG AS COMMANDS. FORTUNATELY THERE ARE NO TBUG COMMAND CHARS WITHIN THE RANGE A - F. BUT NAMES OF FILES COULD BE INTERPRETED AS COMMANDS BY TBUG. IF MEMORY LOCATION A091 IS SET TO ZERO BEFORE A LOAD OR INPUT THEN WHEN THE LOAD OR INPUT HAS BEEN COMPLETED, THE SYSTEM WILL ENTER A CLOSED SOFTWARE LOOP WITHIN TBUG AND SO WILL NOT ACKNOWLEDGE ANY INPUT TO EITHER PORT. THIS ALLOWS THE USER TO HAVE A CUP OF TEA WITH A CLEAR CONSCIENCE WHILST TBUG IS SEARCHING FOR AND LOADING DATA FROM TAPE. TO REGAIN CONTROL OF THE SYSTEM (AFTER THE TEA), PRESS RESET.

TRACE.

SINGLE STEP HARDWARE ON MON2 WORKS WITH TBUG WITHOUT MODIFICATIONS. SEE MON2 DESIGN NOTE FOR DETAILS.

COMPATIBILITY BETWEEN TBUG AND MIKBUG.

TBUG IS COMPATIBLE WITH MIKBUG IN ALL BUT ONE POINT:- MIKBUG USES A PIA AS ITS PORT IT WORKS THIS PIA AS A SERIAL PORT ! (?). WHEN COMMERCIAL SOFTWARE WRITERS PRODUCE LARGE ITEMS (4K, 8K BASIC) IT IS NOT RECORDED IN 'S1' FORMAT BUT IN 'X1' BINARY FORMAT WHICH SAVES TIME. AT THE BEGINNING OF THIS TYPE OF TAPE IS A SMALL PROGRAM THAT READS 'X1' BINARY DATA AND LOADS IT INTO MEMORY. UNFORTUNATELY THIS LITTLE PROGRAM READS FROM THE MIKBUG PIA DIRECTLY, IT DOES THIS SO THAT THE FULL 8 BITS OF DATA CAN BE LOADED (MIKBUGS INPUT SUBROUTINE CLEARS THE MOST SIGNIFICANT BIT AND SO CANNOT BE USED FOR THE BINARY LOADER TYPE OF PROGRAM). SO WE HAVE TO USE OUR OWN PROGRAM TO LOAD 'X1' DATA AND WE CAN USE TBUG'S SUBROUTINE 'INXBITS' TO READ THE FULL 8 BITS FOR US. DONT WORRY THE SOFTWARE TO DO THIS HAS BEEN INCLUDED ELSEWHERE IN THIS DESIGN NOTE SEE 'BINLOAD'. THIS PROGRAM SHOULD BE ENTERED AND RUN FROM A092 A COMPLETED LOAD IS INDICATED BY 8 LINE FEEDS, AN ASTERISK AND CONTROL IS PASSED BACK TO TBUG.

Tc

```
00001                          NAM    TBUG
00002 E000                     ORG    $E000
00003       8000    ACIACN     EQU    $8000
00004       8001    ACIADT     EQU    $8001
00005       8010    PIAAPR     EQU    $8010
00006       8011    PIAACN     EQU    $8011
00007       8012    PIABPR     EQU    $8012
00008       8013    PIABCN     EQU    $8013
00009       0000    YCIACN     EQU    $0000
00010       0001    YCIADT     EQU    $0001
00011       0010    XIAAPR     EQU    $0010
00012       0011    XIAACN     EQU    $0011
00013       0012    XIABPR     EQU    $0012
00014       0013    XIABCN     EQU    $0013
00015 E000 FE A000  IRQ        LDX    IRV       IRQ VECTOR FROM SCRATCHPAD
00016 E003 6E 00               JMP    0,X       AND JUMP TO WHERE IT POINTS
00017 E005 FE A006  NMI        LDX    NMV       GET NMI VECTOR
00018 E008 6E 00               JMP    0,X       AND JUMP
00019 E00A 20 07    LOAD       BRA    LOAD3
00020 E00C 7D A091  QWAIT      TST    WAIT      IF WAIT=0 THEN LOOP FOREVER
00021 E00F 27 FE    WTLOOP     BEQ    WTLOOP    EXIT IS ACHIEVED BY RESET
00022 E011 20 31               BRA    C1
00023 E013 8D 63    LOAD3      BSR    INCH      GET CHAR LOOK FOR 'S1' (START
00024 E015 81 53               CMP A  #'S       OF BLOCK) OR 'S9' END OF FILE
00025 E017 26 FA               BNE    LOAD3     NOT 'S SO LOOK AGAIN
00026 E019 8D 5D               BSR    INCH
00027 E01B 81 39               CMP A  #'9
00028 E01D 27 ED               BEQ    QWAIT     BRANCH IF END OF FILE (S9)
00029 E01F 81 31               CMP A  #'1
00030 E021 26 F0               BNE    LOAD3     IF 'S1' OR 'S9' NOT FOUND
00031 E023 7F A00A             CLR    CKSM      'S1' FOUND BEGIN LOAD
00032 E026 8D 2D               BSR    BYTE      GET BYTE COUNT
00033 E028 80 02               SUB A  #02
00034 E02A B7 A00B             STA A  BYTECT
00035 E02D 8D 18               BSR    BADDR     GET ADDRESS OF BLOCK
00036 E02F 8D 24    LOAD11     BSR    BYTE      GET DATA
00037 E031 7A A00B             DEC    BYTECT
00038 E034 27 05               BEQ    LOAD15
00039 E036 A7 00               STA A  0,X
00040 E038 08                  INX
00041 E039 20 F4               BRA    LOAD11
00042 E03B 7C A00A  LOAD15     INC    CKSM
00043 E03E 27 D3               BEQ    LOAD3     CHECKSUM O.K.
00044 E040 86 3F    LOAD19     LDA A  #'?       ERROR SO PRINT QUESTION MARK
00045 E042 8D 31               BSR    OUTCH
00046 E044 7E E0E3  C1         JMP    CONTRL
00047 E047 8D 0C    BADDR      BSR    BYTE      GET 1ST. BYTE OF ADDRESS
00048 E049 B7 A00C             STA A  XHI
```

```
00049 E04C 8D 07          BSR    BYTE     GET 2ND. BYTE
00050 E04E B7 A00D        STA A  XLOW
00051 E051 FE A00C        LDX    XHI      X=ADDRESS
00052 E054 39             RTS
00053 E055 8D 53   BYTE   BSR    INHEX    GET 1ST. HEX CHAR (MOST SIG.)
00054 E057 48             ASL A           SHIFT LEFT 4 TIMES
00055 E058 48             ASL A
00056 E059 48             ASL A
00057 E05A 48             ASL A
00058 E05B 16             TAB             SAVE IN B
00059 E05C 8D 4C          BSR    INHEX    GET 2ND. HEX CHAR.
00060 E05E 1B             ABA             ADD ACCUMULATORS TO FORM BYTE
00061 E05F 16             TAB
00062 E060 FB A00A        ADD B  CKSM     ADD THIS BYTE TO CHECKSUM
00063 E063 F7 A00A        STA B  CKSM
00064 E066 39             RTS
00065 E067 44      OUTHL  LSR A
00066 E068 44             LSR A
00067 E069 44             LSR A
00068 E06A 44             LSR A
00069 E06B 84 0F   OUTHR  AND A  #$0F
00070 E06D 8B 30          ADD A  #$30     CONVERT HEX TO ASCII
00071 E06F 81 39          CMP A  #$39
00072 E071 23 02          BLS    OUTCH
00073 E073 8B 07          ADD A  #07
00074 E075 7E E1D1 OUTCH  JMP    OUTE
00075 E078 7E E1AC INCH   JMP    INPOLL
00076 E07B 8D F8   PDT2   BSR    OUTCH
00077 E07D 08             INX
00078 E07E A6 00   PDATA  LDA A  0,X      OUTPUT AREA OF MEMORY AS TEXT
00079 E080 81 04          CMP A  #04      UNTIL EOT(04) IS FOUND
00080 E082 26 F7          BNE    PDT2
00081 E084 39             RTS
00082 E085 8D C0   CHANGE BSR    BADDR    MEMORY INSPECT/MODIFY COMMAND
00083 E087 CE E19B CHA51  LDX    #MCL
00084 E08A 8D F2          BSR    PDATA    GET ADDRESS
00085 E08C CE A00C        LDX    #XHI
00086 E08F 8D 37          BSR    OUT4HS   PRINT ADDRESS
00087 E091 FE A00C        LDX    XHI
00088 E094 8D 34          BSR    OUT2HS   PRINT CONTENTS OF THIS
00089 E096 FF A00C        STX    XHI      LOCATION
00090 E099 8D DD          BSR    INCH
00091 E09B 81 20          CMP A  #$20
00092 E09D 26 E8          BNE    CHA51    IF NOT SP THEN TRY NEXT LOC
00093 E09F 8D B4          BSR    BYTE     GET NEW DATA FOR THIS  LOC
00094 E0A1 09             DEX
00095 E0A2 A7 00          STA A  0,X      AND STORE
00096 E0A4 A1 00          CMP A  0,X      DID IT STORE?
00097 E0A6 27 DF          BEQ    CHA51    IF IT DID THEN LOOK AT NEXT
00098 E0A8 20 96          BRA    LOAD19   MEMORY FAULT PRINT '?'
00099 E0AA 8D CC   INHEX  BSR    INCH
00100 E0AC 80 30          SUB A  #$30     CONVERT ASCII TO HEX
00101 E0AE 2B 94          BMI    C1       - TO 'C1' IF NOT HEX
```

```
00102 E0B0 81 09          CMP A  #$09
00103 E0B2 2F 0A          BLE    HGRET
00104 E0B4 81 11          CMP A  #$11
00105 E0B6 2B 8C          BMI    C1          "
00106 E0B8 81 16          CMP A  #$16
00107 E0BA 2E 88          BGT    C1          "
00108 E0BC 80 07          SUB A  #$07
00109 E0BE 39       HGRET RTS
00110 E0BF A6 00    OUT2H LDA A  0,X         OUTPUT 2 HEX CHARS
00111 E0C1 8D A4    OUT2HA BSR   OUTHL       OUTPUT LEFT HEX CHAR.
00112 E0C3 A6 00          LDA A  0,X
00113 E0C5 08             INX
00114 E0C6 20 A3          BRA    OUTHR       OUTPUT RIGHT HEX CHAR RETURN.
00115 E0C8 8D F5    OUT4HS BSR   OUT2H       OUTPUT 4 HEX + SPACE
00116 E0CA 8D F3    OUT2HS BSR   OUT2H       OUTPUT 2 HEX + SPACE
00117 E0CC 86 20    OUTS   LDA A  #$20       OUTPUT A SPACE
00118 E0CE 20 A5          BRA    OUTCH
00119 E0D0 8E A042 START  LDS    #$A042      INIT STACK POINTER
00120 E0D3 BF A008        STS    SP
00121 E0D6 86 03          LDA A  #03         RESET ACIA
00122 E0D8 B7 8000        STA A  ACIACN
00123 E0DB 4C             INC A
00124 E0DC B7 8011        STA A  PIAACN      INIT PIA(A)
00125 E0DF BD E2A3        JSR    INIT        'INIT' CONTINUES INITIALISING
00126 E0E2 01             NOP                THE SYSTEM
00127 E0E3 8E A042 CONTRL LDS    #$A042
00128 E0E6 CE E19B        LDX    #MCL        PRINT ASTERISK
00129 E0E9 8D 93          BSR    PDATA
00130 E0EB 8D 8B          BSR    INCH        GET COMMAND
00131 E0ED 16             TAB
00132 E0EE 8D DC          BSR    OUTS        OUTPUT A SPACE
00133 E0F0 CE E3D3        LDX    #TABLE
00134 E0F3 6D 00    SCAN  TST    0,X         END OF TABLE?
00135 E0F5 27 14          BEQ    NOCOMM
00136 E0F7 E1 00          CMP B  0,X         DOES COMMAND MATCH?
00137 E0F9 26 04          BNE    INCT        IF NOT THEN BRANCH
00138 E0FB EE 01          LDX    1,X         GET COMMAND ADDRESS
00139 E0FD 6E 00          JMP    0,X         AND JUMP TO IT
00140 E0FF 08       INCT  INX                X+1 TO POINT AT THE NEXT
00141 E100 08             INX                COMMAND CHAR.
00142 E101 08             INX
00143 E102 20 EF          BRA    SCAN
00144 E104 86 11    SERFOR LDA A  #$11       CHANGE ACIA SERIAL FORMAT
00145 E106 B7 8000        STA A  ACIACN      TO 2 STOP  (FOR 110 BAUD)
00146 E109 20 D8          BRA    CONTRL      THEN RETURN
00147 E10B 7E E040 NOCOMM JMP    LOAD19
00148 E10E 01             NOP
00149 E10F BE A008 GO     LDS    SP          RESTORE STACK POINTER
00150 E112 3B             RTI                AND LOAD INTERNAL REGS AND GO
00151 E113 BF A008 SFE    STS    SP          SAVE STACK POINTER
00152 E116 30             TSX
00153 E117 6D 06          TST    6,X         DECREMENT PROGRAM COUNTER NOW
00154 E119 26 02          BNE    NDEC        ON STACK
```

```
00155 E11B 6A 05          DEC    5,X
00156 E11D 6A 06   NDEC    DEC    6,X
00157 E11F FE A008 PRINT   LDX    SP        PRINT CONTENTS OF STACK
00158 E122 08              INX
00159 E123 8D A5           BSR    OUT2HS
00160 E125 8D A3           BSR    OUT2HS
00161 E127 8D A1           BSR    OUT2HS
00162 E129 8D 9D           BSR    OUT4HS
00163 E12B 8D 9B           BSR    OUT4HS
00164 E12D CE A008         LDX    #SP
00165 E130 8D 96           BSR    OUT4HS
00166 E132 20 AF   C2      BRA    CONTRL
00167 E134 0D      MTAPE   FCB    $D,$A,0,0,0,0,'S,'1,$4   'S1'
      E135 0A
      E136 00
      E137 00
      E138 00
      E139 00
      E13A 53
      E13B 31
      E13C 04
00168 E13D BD E299 PUNCH   JSR    AUTO      - SETTLE AUTO RECORDING LEVEL
00169 E140 FE A002 NAUTP   LDX    BEGA
00170 E143 FF A00F         STX    TW        - PUT START ADDRESS IN TEMP
00171 E146 B6 A005 PUN11   LDA A  ENDA+1
00172 E149 B0 A010         SUB A  TW+1
00173 E14C F6 A004         LDA B  ENDA
00174 E14F F2 A00F         SBC B  TW
00175 E152 26 04           BNE    PUN22
00176 E154 81 10           CMP A  #16
00177 E156 25 02           BCS    PUN23
00178 E158 86 0F   PUN22   LDA A  #15
00179 E15A 8B 04   PUN23   ADD A  #4
00180 E15C B7 A011         STA A  MCONT     FRAME COUNT OF THIS RECORD
00181 E15F 80 03           SUB A  #3
00182 E161 B7 A00E         STA A  TEMP      BYTE COUNT OF THIS RECORD
00183 E164 CE E134         LDX    #MTAPE
00184 E167 BD E07E         JSR    PDATA
00185 E16A 5F              CLR B             CLEAR CHECKSUM
00186 E16B CE A011         LDX    #MCONT
00187 E16E 8D 26           BSR    PUNT2     OUTPUT FRAME COUNT
00188 E170 CE A00F         LDX    #TW
00189 E173 8D 21           BSR    PUNT2     OUTPUT START ADDRESS OF THIS
00190 E175 8D 1F           BSR    PUNT2
00191 E177 FE A00F         LDX    TW
00192 E17A 8D 1A   PUN32   BSR    PUNT2
00193 E17C 7A A00E         DEC    TEMP
00194 E17F 26 F9           BNE    PUN32     OUTPUT A BYTE
00195 E181 FF A00F         STX    TW
00196 E184 53              COM B
00197 E185 37              PSH B
00198 E186 30              TSX
00199 E187 8D 0D           BSR    PUNT2     OUTPUT CHECKSUM
```

```
00200 E189 33              PUL B
00201 E18A FE A00F         LDX    TW
00202 E18D 09              DEX
00203 E18E BC A004         CPX    ENDA
00204 E191 26 B3           BNE    PUN11
00205 E193 7E E2C5         JMP    S9OUT     OUTPUT 'S9' AND RETURN
00206 E196 EB 00   PUNT2   ADD B  0,X       OUTPUT 2 HEX CHARS AND UPDATE
00207 E198 7E E0BF         JMP    OUT2H
00208 E19B 0D      MCL     FCB    $D,$A,0,0,'*,$4
      E19C 0A
      E19D 00
      E19E 00
      E19F 2A
      E1A0 04
00209 E1A1 8D 7A   JUMP    BSR    BADLK     GET ADDRESS FROM USER
00210 E1A3 6E 00           JMP    0,X       AND JUMP TO IT
00211 E1A5 FF A012 SAVX    STX    XTEMP     STORE X
00212 E1A8 CE 8000         LDX    #$8000    POINT X AT IO
00213 E1AB 39              RTS
00214 E1AC 86 7F   INPOLL  LDA A  #$7F      INPUT ONE CHAR FROM PORT INTO
00215 E1AE 8D F5   INXBIT  BSR    SAVX      PUT '7F' IGNORE M.S. BIT
00216 E1B0 37              PSH B             SAVE B
00217 E1B1 E6 13   POLL1   LDA B  XIABCN,X    DOES PIA(B) HAVE SOMETHING
00218 E1B3 2A 04           BPL    POLL2
00219 E1B5 A4 12           AND A  XIABPR,X    GET DATA FROM PIA(B)
00220 E1B7 20 0F           BRA    RET
00221 E1B9 E6 11   POLL2   LDA B  XIAACN,X    DOES PIA(A) HAVE SOMETHING
00222 E1BB 2A 04           BPL    POLL3
00223 E1BD A4 10           AND A  XIAAPR,X    GET DATA FROM PIA(A)
00224 E1BF 20 07           BRA    RET
00225 E1C1 E6 00   POLL3   LDA B  YCIACN,X    DOES ACIA HAVE SOMETHING?
00226 E1C3 54              LSR B
00227 E1C4 24 EB           BCC    POLL1
00228 E1C6 A4 01           AND A  YCIADT,X    GET DATA FROM ACIA
00229 E1C8 7D A090 RET     TST    RUBIG     IGNORE RUBOUT CHARS?
00230 E1CB 26 08           BNE    DIG
00231 E1CD 81 7F           CMP A  #$7F
00232 E1CF 20 02           BRA    CRQ       MESS OF BRANCHES IS SO THAT
00233 E1D1 20 06   OUTE    BRA    OUTP      OUTPUT ROUTINE APPEARS AT
00234 E1D3 27 DC   CRQ     BEQ    POLL1     'E1D1'
00235 E1D5 33      DIG     PUL B
00236 E1D6 FE A012         LDX    XTEMP
00237 E1D9 8D CA   OUTP    BSR    SAVX
00238 E1DB CE A080         LDX    #DISAD    POINT X AT TBUG SCRATCHPAD
00239 E1DE 37              PSH B
00240 E1DF 36              PSH A             SAVE A
00241 E1E0 81 0D           CMP A  #0D       CARRIAGE RETURN?
00242 E1E2 26 0C           BNE    CHK1
00243 E1E4 6D 02           TST    KOLPTR,X    WHERE IS THE CURSOR?
00244 E1E6 27 25           BEQ    CONT      CARRIAGE ALREADY RETURNED
00245 E1E8 86 20           LDA A  #$20      PUT SPACE IN A
00246 E1EA 8D 34           BSR    PLOT      CLEAR OLD CURSOR AND POINT
00247 E1EC 6F 02           CLR    KOLPTR,X    TO BEGINNING OF LINE
```

```
00248 E1EE 20 1D          BRA    CONT
00249 E1F0 81 0A    CHK1  CMP A  #$0A      LINE FEED?
00250 E1F2 26 0D          BNE    CHK2
00251 E1F4 6D 02          TST    KOLPTR,X   WHERE IS THE CURSOR?
00252 E1F6 27 04          BEQ    ATEND      CURSOR AT BEGINNING OF LINE
00253 E1F8 86 20          LDA A  #$20       PUT SPACE IN A
00254 E1FA 8D 24          BSR    PLOT       CLEAR CURSOR
00255 E1FC BD E264  ATEND JSR    SCROLL     SCROLL DISPLAY
00256 E1FF 20 0C          BRA    CONT
00257 E201 81 1F    CHK2  CMP A  #$1F       IS IT A CONTROL CHAR.?
00258 E203 23 08          BLS    CONT       IF SO THEN OUTPUT TO ACIA BUT
00259 E205 8D 19          BSR    PLOT       DONT DISPLAY ON VDU
00260 E207 6C 02          INC    KOLPTR,X   POINT TO NEXT COLUMN
00261 E209 86 2E          LDA A  #'.
00262 E20B 8D 13          BSR    PLOT       AND PUT THE CURSOR THERE.
00263 E20D 32       CONT  PUL A            RESTORE A
00264 E20E F6 8000 NOTRDY LDA B  ACIACN     ACIA READY TO TRANSMIT?
00265 E211 54             LSR B
00266 E212 54             LSR B
00267 E213 24 F9          BCC    NOTRDY     NO
00268 E215 B7 8001        STA A  ACIADT     PASS CHAR TO ACIA FOR TRANSMI
00269 E218 33             PUL B            RESTORE B
00270 E219 FE A012        LDX    XTEMP      RESTORE X
00271 E21C 39             RTS
00272 E21D 7E E047 BADLK  JMP    BADDR
00273 E220 E6 02    PLOT  LDA B  KOLPTR,X   PUT COLUMN POINTER IN B
00274 E222 A7 04          STA A  KHARSTR,X   STORE CHAR TO BE DISPLAYED
00275 E224 A6 03          LDA A  XOWPTR,X    PUT ROW POINTER IN A
00276 E226 C1 27          CMP B  #39         COLUMN>39?
00277 E228 23 05          BLS    INLINE      BRANCH IF NOT
00278 E22A 5F             CLR B            PAST END OF LINE SO POINT AT
00279 E22B 6F 02          CLR    KOLPTR,X    START OF LINE
00280 E22D 8D 35          BSR    SCROLL     AND SCROLL DISPLAY
00281 E22F 81 17  INLINE  CMP A  #23        ROW>23?
00282 E231 23 02          BLS    INROW      BRANCH IF NOT
00283 E233 86 17          LDA A  #23        PUT 23 IN ROW
00284 E235 C5 20  INROW   BIT B  #$20       BIT 5 OF COLUMN POINTER SET?
00285 E237 27 0B          BEQ    NOMOD      COLUMN AND ROW NEED ROTATING
00286 E239 C4 07  C5SET   AND B  #$07       SO CLEAR BITS 3 AND 4 OF COL
00287 E23B E7 00          STA B  XISAD,X    PUT B AWAY TEMPORARILY
00288 E23D 16             TAB              COPY A TO B  A =ROW POINTER
00289 E23E C4 18          AND B  #$18      CLEAR BITS 1,2 0 OF ROW POINT
00290 E240 EA 00          ORA B  XISAD,X    'OR' IT WITH COLUMN POINTER
00291 E242 8A 18          ORA A  #$18      SET BITS 3 AND 4 OF ROW POINT
00292 E244 E7 01  NOMOD   STA B  XISAD+1,X
00293 E246 C6 05          LDA B  #05         - ROTATE COUNT
00294 E248 48     ROTATE  ASL A            NOW ROTATE TO FORM ADDRESS
00295 E249 69 00          ROL    XISAD,X
00296 E24B 5A             DEC B
00297 E24C 26 FA          BNE    ROTATE
00298 E24E AA 01          ORA A  XISAD+1,X
00299 E250 A7 01          STA A  XISAD+1,X
00300 E252 E6 00          LDA B  XISAD,X
```

```
00301 E254 CA F8          ORA B  #$F8      ADD F8 TO MS. BYTE OF ADDRESS
00302 E256 E7 00          STA B  XISAD,X
00303 E258 E6 04          LDA B  KHARST,X   GET CHARACTER
00304 E25A EE 00          LDX    XISAD,X    DISAD=DISPLAY ADDRESS
00305 E25C A6 00          LDA A  0,X
00306 E25E E7 00          STA B  0,X        AND DISPLAY
00307 E260 CE A080        LDX    #DISAD     RETURN $A080 IN X TO CALLER
00308 E263 39             RTS
00309 E264 37       SCROLL PSH B            SAVE B
00310 E265 CE F800        LDX    #$F800     POINT X AT TOP LEFT OF SCREEN
00311 E268 E6 20    SCRL1 LDA B  $20,X      LOAD B WITH CHAR BELOW
00312 E26A E7 00          STA B  0,X        AND STORE IT ABOVE
00313 E26C 08             INX               INCREMENT POINTER
00314 E26D 8C FBD8        CPX    #$FBD8     FINISHED?
00315 E270 26 F6          BNE    SCRL1
00316 E272 CE FAE8        LDX    #$FAE8     NOW TAKE CARE OF NASTY BITS.
00317 E275 E6 00    SCRL2 LDA B  0,X
00318 E277 E7 F8          STA B  $F8,X      BOTTOM ROW USED AS TEMP FOR
00319 E279 08             INX               BYTES THAT WERE AT FB08->FB17
00320 E27A 8C FAF8        CPX    #$FAF8
00321 E27D 26 F6          BNE    SCRL2
00322 E27F CE FAE0        LDX    #$FAE0     AND CLEAR THE BOTTOM LINE
00323 E282 C6 20          LDA B  #$20
00324 E284 E7 00    CLRLIM STA B  0,X
00325 E286 08             INX
00326 E287 8C FB00        CPX    #$FB00     BOTTOM LINE CLEARED?
00327 E28A 26 F8          BNE    CLRLIM
00328 E28C E7 F0    CLRLIL STA B  #$F0,X    NOW CLEAR BYTES (FBF0-FBF7)
00329 E28E 08             INX
00330 E28F 8C FB08        CPX    #$FB08
00331 E292 26 F8          BNE    CLRLIL     DONE?
00332 E294 33             PUL B             SAVE B
00333 E295 CE A080        LDX    #$A080     RETURN $A080 TO CALLER
00334 E298 39             RTS
00335 E299 C6 20    AUTO  LDA B  #$20       OUTPUT 32 NULLS
00336 E29B 4F             CLR A             TO ALLOW AUTO RECORDING
00337 E29C BD E1D1 NULOUT JSR    OUTE       LEVEL TO SETTLE
00338 E29F 5A             DEC B
00339 E2A0 26 FA          BNE    NULOUT
00340 E2A2 39             RTS
00341 E2A3 B7 8013 INIT   STA A  PIABCN     CONTIUATION OF INITIALIZATION
00342 E2A6 86 17          LDA A  #$17       INIT ROW POINTER SO AS TO
00343 E2A8 B7 A083        STA A  ROWPTR     WRITE ON THE BOTTOM LINE
00344 E2AB 86 15          LDA A  #$15       INIT ACIA-ONE STOP BIT AND
00345 E2AD B7 8000        STA A  ACIACN     8 BITS OF DATA NO PARITY
00346 E2B0 7F A090        CLR    RUBIG      RUBOUT CHARS IGNORED.
00347 E2B3 CE E2BA        LDX    #TBSTR     PRINT 'TBUG'
00348 E2B6 BD E07E        JSR    PDATA
00349 E2B9 39             RTS
00350 E2BA 0D       TBSTR FCB    $D,$A,$A,0,0,'T   'TBUG'
      E2BB 0A
      E2BC 0A
      E2BD 00
```

0D0A,54 4A 55 47 0D0A 04

```
      E2BE 00
      E2BF 54
00351 E2C0 42              FCB    'B,'U,'G
      E2C1 55
      E2C2 47
00352 E2C3 04              FCB    $4,0
      E2C4 00
00353 E2C5 CE E3A1 S9OUT   LDX    #S9STR   -OUTPUT 'S9'
00354 E2C8 BD E07E         JSR    PDATA
00355 E2CB 20 42           BRA    C4
00356 E2CD 8D 46   COPY    BSR    BDSP     COPY A BLOCK OF MEMORY-GET
00357 E2CF FF A085         STX    NAME     START OF BLOCK ADDRESS
00358 E2D2 8D 41           BSR    BDSP     GET END ADDRESS OF BLOCK
00359 E2D4 08              INX             INCREMENT IT
00360 E2D5 FF A00F         STX    TW       AND STORE
00361 E2D8 8D 3B           BSR    BDSP     GET START ADDRESS OF START
00362 E2DA FE A085         LDX    NAME     OD DESTINATION BLOCK
00363 E2DD A6 00   MOVEIT  LDA A  0,X      MOVE MOVE MOVE!
00364 E2DF FE A00C         LDX    XHI
00365 E2E2 A7 00           STA A  0,X
00366 E2E4 08              INX
00367 E2E5 FF A00C         STX    XHI
00368 E2E8 FE A085         LDX    NAME
00369 E2EB 08              INX
00370 E2EC FF A085         STX    NAME
00371 E2EF BC A00F         CPX    TW       END?
00372 E2F2 26 E9           BNE    MOVEIT
00373 E2F4 20 19           BRA    C4
00374 E2F6 8D 1D   BLKTAB  BSR    BDSP     TABULATE BLOCK OF 64 BYTES
00375 E2F8 C6 40           LDA B  #$40     GET ADDRESS OF START OF BLOCK
00376 E2FA CE E394         LDX    #CRS
00377 E2FD BD E07E         JSR    PDATA    OUTPUT CARRIAGE RETURN ETC.
00378 E300 FE A00C         LDX    XHI
00379 E303 BD E0BF TAB2    JSR    OUT2H    OUTPUT 2 HEX CHARS (ONE BYTE)
00380 E306 8D 0A           BSR    OUTSLK   AND 3 SPACES
00381 E308 8D 08           BSR    OUTSLK
00382 E30A 8D 06           BSR    OUTSLK
00383 E30C 5A              DEC B           FINISHED?
00384 E30D 26 F4           BNE    TAB2
00385 E30F 7E E0E3 C4      JMP    CONTRL
00386 E312 7E E0CC OUTSLK  JMP    OUTS
00387 E315 BD E047 BDSP    JSR    BADDR    BUILD ADDRESS AND OUTPUT
00388 E318 8D F8           BSR    OUTSLK   A SPACE
00389 E31A 39              RTS
00390 E31B 7E E1D1 OUTLNK  JMP    OUTE
00391 E31E C6 0C   TEXSTR  LDA B  #0C      INPUT UP TO TEN CHARS
00392 E320 CE A085         LDX    #NAME    AND TERMINATE WITH EOT($C4)
00393 E323 5A      NFIN    DEC B           STRING STARTS AT #A085
00394 E324 26 03           BNE    NERR
00395 E326 7E E040         JMP    LOAD19   TOO MANY CHARS! SO EXIT
00396 E329 BD E1AC NERR    JSR    INPOLL
00397 E32C 81 0D           CMP A  #0D      CARRIAGE RETURN(END OF STRING
00398 E32E 27 05           BEQ    ENDSTR
```

```
00399 E330 A7 00          STA A  0,X      STORE CHAR
00400 E332 08             INX
00401 E333 20 EE          BRA    NFIN
00402 E335 86 0A   ENDSTR LDA A  #0A      OUTPUT LINE FEED
00403 E337 8D E2          BSR    OUTLNK
00404 E339 86 04          LDA A  #04      TERMINATE STRING
00405 E33B A7 00          STA A  0,X
00406 E33D 39             RTS
00407 E33E 8D DE   SAVE   BSR    TEXSTR
00408 E340 BD E299        JSR    AUTO
00409 E343 4C             INC A           SOH (START OF HEADER)
00410 E344 8D D5          BSR    OUTLNK   OUTPUT IT 4 TIMES TO MARK THE
00411 E346 8D D3          BSR    OUTLNK   BEGINNING OF A HEADER
00412 E348 8D D1          BSR    OUTLNK
00413 E34A 8D CF          BSR    OUTLNK
00414 E34C CE A085        LDX    #NAME    OUTPUT NAME OF FILE
00415 E34F BD E07E        JSR    PDATA
00416 E352 7E E140        JMP    NAUTP    OUTPUT IT IN 'S1' FORMAT
00417 E355 8D C7   INFILE BSR    TEXSTR   GET NAME OF FILE WHICH WE ARE
00418 E357 C6 04   NOMAT  LDA B  #04      LOOKING FOR
00419 E359 8D 3C   N4IN   BSR    INCASSQ   ACCEPT INPUT FROM SERIAL
00420 E35B 81 01          CMP A  #01      PORT ONLE (CASSETTE)
00421 E35D 27 12          BEQ    HDFN     'SOH' CHAR FOUND? BRANCH IF
00422 E35F 81 53          CMP A  #'S      SO END OF FILE (S9)?
00423 E361 26 F4          BNE    NOMAT    IF NEITHER OF THESE THEN LOOK
00424 E363 8D 32          BSR    INCASSQ   AGAIN
00425 E365 81 39          CMP A  #'9      'S9' ?
00426 E367 26 EE          BNE    NOMAT    BRANCH IF NOT
00427 E369 CE E391        LDX    #ENDST   SAY THAT WE'VE FOUND THE END
00428 E36C BD E07E        JSR    PDATA    ONE OF HIS FILES
00429 E36F 20 E6          BRA    NOMAT    AND LOOK AGAIN
00430 E371 5A      HDFN   DEC B
00431 E372 26 E5          BNE    N4IN     4 'SOH' CHARS FOUND NO BRANCH
00432 E374 CE A085        LDX    #NAME    POINT X TO NAME SEARCHED FOR
00433 E377 BD E1AC CHREQ  JSR    INPOLL   GET CHAR OF NAME COMMING IN
00434 E37A A1 00          CMP A  0,X      DO THEY MATCH?
00435 E37C 27 09          BEQ    NAMSM    IF SO THEN BRANCH
00436 E37E BD E1AC INNAM  JSR    INPOLL   DONT MATCH BUT PRINT NAME
00437 E381 81 0A          CMP A  #0A      OF FILE WE'VE FOUND ANYWAY
00438 E383 26 F9          BNE    INNAM
00439 E385 20 D0          BRA    NOMAT    AND LOOK AGAIN
00440 E387 08      NAMSM  INX
00441 E388 86 04          LDA A  #04
00442 E38A A1 00          CMP A  0,X      END OF NAME?
00443 E38C 26 E9          BNE    CHREQ    BRANCH IF NOT
00444 E38E 7E E00A        JMP    LOAD     LOAD FILE
00445 E391 45      ENDST  FCC    3,END
      E392 4E
      E393 44
00446 E394 0D      CRS    FCB    $D,$A,$4
      E395 0A
      E396 04
00447 E397 B6 8000 INCASS LDA A  ACIACN   INPUT FROM SERIAL PORT ONLY
```

```
00448 E39A 44              LSR A
00449 E39B 24 FA           BCC    INCASSQ
00450 E39D B6 8001         LDA A  ACIADT
00451 E3A0 39              RTS
00452 E3A1 20       S9STR  FCB    '         ,'S
      E3A2 53
00453 E3A3 39              FCB    '9,$4
      E3A4 04
00454 E3A5 BD E315 BCALC   JSR    BDSP      CALCULATE BRANCH OFFSET-
00455 E3A8 08              INX               INCREMENT IT TWICE
00456 E3A9 08              INX
00457 E3AA FF A00F         STX    TW        AND STORE
00458 E3AD BD E315         JSR    BDSP      GET DESTINATION ADDRESS
00459 E3B0 CE A00C         LDX    #XHI
00460 E3B3 E6 01           LDA B  XLOWX,X    CHECK FOR OUT OF RANGE
00461 E3B5 E0 04           SUB B  TWX+1,X
00462 E3B7 A6 00           LDA A  XHIX,X
00463 E3B9 A2 03           SBC A  TWX,X
00464 E3BB 26 07           BNE    POS
00465 E3BD C1 7F           CMP B  #$7F
00466 E3BF 23 0A           BLS    OK
00467 E3C1 7E E040 ERRR    JMP    LOAD19    OUT OF RANGE PRINT ?
00468 E3C4 4C       POS    INC A
00469 E3C5 26 FA           BNE    ERRR
00470 E3C7 C1 7F           CMP B  #$7F
00471 E3C9 23 F6           BLS    ERRR
00472 E3CB E7 00    OK     STA B  XHIX,X
00473 E3CD BD E0BF         JSR    OUT2H     OUTPUT OFFSET TO USER
00474 E3D0 7E E0E3         JMP    CONTRL    AND RETURN
00475 E3D3 47       TABLE  FCB    'G        -JUMP TABLE FOR COMMAND
00476 E3D4 E10F            FDB    GO
00477 E3D6 4D              FCB    'M        -MEMORY INSPECT/MODIFY
00478 E3D7 E085            FDB    CHANGE
00479 E3D9 52              FCB    'R        -PRINT STACK
00480 E3DA E11F            FDB    PRINT
00481 E3DC 50              FCB    'P        -PUNCH IN 'S1' FORMAT
00482 E3DD E13D            FDB    PUNCH
00483 E3DF 4C              FCB    'L        -LOAD 'S1' FORMAT
00484 E3E0 E00A            FDB    LOAD
00485 E3E2 56              FCB    'V        -FILE WITH NAME
00486 E3E3 E33E            FDB    SAVE
00487 E3E5 49              FCB    'I        -INPUT NAMED FILE
00488 E3E6 E355            FDB    INFILE
00489 E3E8 54              FCB    'T        -CALCULATE BRANCH OFFSET
00490 E3E9 E3A5            FDB    BCALC
00491 E3EB 4B              FCB    'K        -OUTPUT A BLOCK OF MEMORY
00492 E3EC E2F6            FDB    BLKTAB
00493 E3EE 59              FCB    'Y        -COPY BLOCK OF MEMORY
00494 E3EF E2CD            FDB    COPY
00495 E3F1 4A              FCB    'J        -JUMP TO ADDRESS
00496 E3F2 E1A1            FDB    JUMP
00497 E3F4 55              FCB    'U        -CHANGE SERIAL FORMAT TO 2 ST
00498 E3F5 E104            FDB    SERFORM
```

```
00499 E3F7 00              FCB   $00
00500 E3F8 E000            FDB   IRQ       IRQ VECTOR-FOR REFLECTION
00501 E3FA E113            FDB   SFE       SW1 VECTOR-OF MONITORS UP AT
00502 E3FC E005            FDB   NMI       NMI VECTOR-FC00->FFFF
00503 E3FE E0D0            FDB   START     RESTART
00504 A000                 ORG   $A000
00505 A000 0002  IRV       RMB   2         IRQ VECTOR
00506 A002 0002  BEGA      RMB   2         DUMP START ADDRESS
00507 A004 0002  ENDA      RMB   2         DUMP, END ADDRESS
00508 A006 0002  NMV       RMB   2         NMI VECTOR
00509 A008 0002  SP        RMB   2         STACK POINTER
00510 A00A 0001  CKSM      RMB   1         CHECKSUM TEMP STORE
00511 A00B 0001  BYTECT    RMB   1         BYTE COUNT TEMP STORE
00512 A00C 0001  XHI       RMB   1         USED IN 'BADDR' TO FORM
00513 A00D 0001  XLOW      RMB   1         THE ADDRESS (16 BIT NUMBER)
00514 A00E 0001  TEMP      RMB   1         USED AS TEMP STORE IN JUMP
00515 A00F 0002  TW        RMB   2         INDEX REG TEMP STORE
00516 A011 0001  MCONT     RMB   1         FRAME COUNT
00517 A012 0002  XTEMP     RMB   2         'SAVX' SAVES INDEX REG HERE
00518 A014 002E            RMB   46        STACK
00519 A042 0001  STACK     RMB   1
00520 A043 0007  STACIN    RMB   7         FREE MEMORY FROM A04A->A07F
00521      0001  XLOWX     EQU   XLOW-XHI
00522      0003  TWX       EQU   TW-XHI
00523      0000  XHIX      EQU   XHI-XHI
00524 A080                 ORG   $A080
00525 A080 0002  DISAD     RMB   2         DISPLAY ADDRESS TEMP STORE
00526 A082 0001  COLPTR    RMB   1         COLUMN POINTER FOR 'PLOT'
00527 A083 0001  ROWPTR    RMB   1         ROW POINTER FOR 'PLOT'
00528 A084 0001  CHARST    RMB   1         TEMP CHARACTER STORE
00529 A085 000B  NAME      RMB   11        TEMP STORE FOR 'NAME' STRING
00530 A090 0001  RUBIG     RMB   1         0 IF RUBOUTS TO BE IGNORED
00531 A091 0001  WAIT      RMB   1         ZERO FOR WAIT AFTER A LOAD
00532      0000  XISAD     EQU   DISAD-DISAD
00533      0002  KOLPTR    EQU   COLPTR-DISAD
00534      0003  XOWPTR    EQU   ROWPTR-DISAD
00535      0004  KHARST    EQU   CHARST-DISAD
00536                       END
```

SYMBOL TABLE

```
ACIACN 8000  ACIADT 8001  PIAAPR 8010  PIAACN 8011  PIABPR 8012
PIABCN 8013  YCIACN 0000  YCIADT 0001  XIAAPR 0010  XIAACN 0011
XIABPR 0012  XIABCN 0013  IRQ    E000  NMI    E005  LOAD   E00A
QWAIT  E00C  WTLOOP E00F  LOAD3  E013  LOAD11 E02F  LOAD15 E03B
LOAD19 E040  C1     E044  BADDR  E047  BYTE   E055  OUTHL  E067
OUTHR  E06B  OUTCH  E075  INCH   E078  PDT2   E07B  PDATA  E07E
CHANGE E085  CHA51  E087  INHEX  E0AA  HGRET  E0BE  OUT2H  E0BF
OUT2HA E0C1  OUT4HS E0C8  OUT2HS E0CA  OUTS   E0CC  START  E0D0
CONTRL E0E3  SCAN   E0F3  INCT   E0FF  SERFOR E104  NOCOMM E10B
GO     E10F  SFE    E113  NDEC   E11D  PRINT  E11F  C2     E132
MTAPE  E134  PUNCH  E13D  NAUTP  E140  PUN11  E146  PUN22  E158
```

```
PUN23   E15A  PUN32   E17A  PUNT2   E196  MCL     E19B  JUMP    E1A1
SAVX    E1A5  INPOLL  E1AC  INXBIT  E1AE  POLL1   E1B1  POLL2   E1B9
POLL3   E1C1  RET     E1C8  OUTE    E1D1  CRQ     E1D3  DIG     E1D5
OUTP    E1D9  CHK1    E1F0  ATEND   E1FC  CHK2    E201  CONT    E20D
NOTRDY  E20E  BADLK   E21D  PLOT    E220  INLINE  E22F  INROW   E235
C5SET   E239  NOMOD   E244  ROTATE  E248  SCROLL  E264  SCRL1   E268
SCRL2   E275  CLRLIM  E284  CLRLIL  E28C  AUTO    E299  NULOUT  E29C
INIT    E2A3  TBSTR   E2BA  S9OUT   E2C5  COPY    E2CD  MOVEIT  E2DD
BLKTAB  E2F6  TAB2    E303  C4      E30F  OUTSLK  E312  BDSP    E315
OUTLNK  E31B  TEXSTR  E31E  NFIN    E323  NERR    E329  ENDSTR  E335
SAVE    E33E  INFILE  E355  NOMAT   E357  N4IN    E359  HDFN    E371
CHREQ   E377  INNAM   E37E  NAMSM   E387  ENDST   E391  CRS     E394
INCASS  E397  S9STR   E3A1  BCALC   E3A5  ERRR    E3C1  POS     E3C4
OK      E3CB  TABLE   E3D3  IRV     A000  BEGA    A002  ENDA    A004
NMV     A006  SP      A008  CKSM    A00A  BYTECT  A00B  XHI     A00C
XLOW    A00D  TEMP    A00E  TW      A00F  MCONT   A011  XTEMP   A012
STACK   A042  STACIN  A043  XLOWX   0001  TWX     0003  XHIX    0000
DISAD   A080  COLPTR  A082  ROWPTR  A083  CHARST  A084  NAME    A085
RUBIG   A090  WAIT    A091  XISAD   0000  KOLPTR  0002  XOWPTR  0003
KHARST  0004
```

```
S00600004844521B
S113E000FEA0006E00FEA0066E0020077DA09127F2
S113E010FE20318D63815326FA8D5D813927ED8190
S113E0203126F07FA00A8D2D8002B7A00B8D188DAC
S113E030247AA00B2705A7000820F47CA00A27D384
S113E040863F8D317EE0E38D0CB7A00C8D07B7A021
S113E0500DFEA00C398D5348484848168D4C1B16AC
S113E060FBA00AF7A00A3944444444840F8B30814E
S113E0703923028B077EE1D17EE1AC8DF808A6003E
S113E080810426F7398DC0CEE19B8DF2CEA00C8D94
S113E09037FEA00C8D34FFA00C8DDD812026E88D89
S113E0A0B409A700A10027DF20968DCC80302B94E3
S113E0B081092F0A81112B8C81162E88800739A69D
S113E0C0008DA4A6000820A38DF58DF3862020A53D
S113E0D08EA042BFA0088603B780004CB78011BD54
S113E0E0E2A3018EA042CEE19B8D938D8B168DDC35
S113E0F0CEE3D36D002714E1002604EE016E000880
S113E100080820EF8611B7800020D87EE04001BEC9
S113E110A0083BBFA008306D0626026A056A06FE09
S113E120A008088DA58DA38DA18D9D8D9BCEA008E3
S113E1308D9620AF0D0A0000000053304BDE29912
S113E140FEA002FFA00FB6A005B0A010F6A004F236
S113E150A00F26048110250286 0F8B04B7A011801E
S113E16003B7A00ECEE134BDE07E5FCEA0118D26B4
S113E170CEA00F8D218D1FFEA00F8D1A7AA00E2622
S113E180F9FFA00F5337308D0D33FEA00F09BCA04B
S113E1900426B37EE2C5EB007EE0BF0D0A00002A30
S113E1A0048D7A6E00FFA012CE800039867F8DF533
S113E1B037E6132A04A412200FE6112A04A410201F
S113E1C007E6005424EBA4017DA0902608817F205B
S113E1D002200627DC33FEA0128DCACEA08037367B
S113E1E0810D260C6D02272586208D346F02201D9B
S113E1F0810A260D6D02270486208D24BDE2642049
S113E2000C811F23088D196C02862E8D1332F68023
S113E21000545424F9B7800133FEA012397EE0473C
```

```
S113E220E602A704A603C12723055F6F028D35818B
S113E2301723028617C520270BC407E70016C41846
S113E240EA008A18E701C6054869005A26FAAA01B5
S113E250A701E600CAF8E700E604EE00A600E7001E
S113E260CEA0803937CEF800E620E700088CFBD832
S113E27026F6CEFAE8E600E7F8088CFAF826F6CE99
S113E280FAE0C620E700088CFB0026F8E7F0088CCB
S113E290FB0826F833CEA08039C6204FBDE1D15A01
S113E2A026FA39B780138617B7A0838615B780007E
S113E2B07FA090CEE2BABDE07E390D0A0A0D005478
S113E2C04255470400CEE3A1BDE07E20428D46FFC7
S113E2D0A0858D4108FFA00F8D3BFEA085A600FE02
S113E2E0A00CA70008FFA00CFEA08508FFA085BC19
S113E2F0A00F26E920198D1DC640CEE394BDE07E13
S113E300FEA00CBDE0BF8D0A8D088D065A26F47E52
S113E310E0E37EE0CCBDE0478DF8397EE1D1C60C68
S113E320CEA0855A26037EE040BDE1AC810D2705D1
S113E330A7000820EE860A8DE28604A700398DDE48
S113E340BDE2994C8DD58DD38DD18DCFCEA085BD19
S113E350E07E7EE1408DC7C6048D3C810127128199
S113E3605326F48D32813926EECEE391BDE07E2032
S113E370E65A26E5CEA085BDE1ACA1002709BDE1A2
S113E380AC810A26F920D0088604A10026E97EE0A3
S113E3900A454E440D0A04B6800044424FAB68001AE
S113E3A03920533904BDE3150808FFA00FBDE31558
S113E3B0CEA00CE601E004A600A2032607C17F2339
S113E3C00A7EE0404C26FAC17F23F6E700BDE0BF99
S113E3D07EE0E347E10F4DE08552E11F50E13D4C03
S113E3E0E00A56E33E49E35554E3A54BE2F659E20D
S113E3F0CD4AE1A155E10400E000E113E005E0D0DD
S9030000FC
```

```
00001                          NAM    PLOT
00002 A092                     ORG    $A092
00003 A092 CE F800 START       LDX    #$F800    CLEAR SCREEN
00004 A095 86 20               LDA A  #$20
00005 A097 A7 00   CLRSCR      STA A  0,X
00006 A099 08                  INX
00007 A09A 8C FBF8             CPX    #$FBF8
00008 A09D 26 F8               BNE    CLRSCR
00009 A09F CE A080             LDX    #$A080
00010 A0A2 86 0D   PLOTIN      LDA A  #$0D
00011 A0A4 BD E1D1             JSR    OUTCH
00012 A0A7 BD E0CC             JSR    OUTS
00013 A0AA BD E055             JSR    BYTE      GET COL COOR FROM USER
00014 A0AD A7 05               STA A  5,X
00015 A0AF BD E0CC             JSR    OUTS
00016 A0B2 BD E055             JSR    BYTE      GET ROW COOR FROM USER
00017 A0B5 A7 06               STA A  6,X
00018 A0B7 BD E0CC             JSR    OUTS
00019 A0BA BD E055             JSR    BYTE      GET HEX FOR CHAR TO BE DISPLA
00020 A0BD E6 05               LDA B  5,X
00021 A0BF E7 02               STA B  2,X
00022 A0C1 E6 06               LDA B  6,X
00023 A0C3 E7 03               STA B  3,X
00024 A0C5 BD E21C             JSR    PLOT      PLOT CHARACTER
00025 A0C8 86 17               LDA A  #$17
00026 A0CA A7 03               STA A  3,X
00027 A0CC 6F 02               CLR    CPTR,X
00028 A0CE C6 26               LDA B  #$26      CLEAR BOTTOM LINE
00029 A0D0 BD E0CC CLRLIN      JSR    OUTS
00030 A0D3 5A                  DEC B
00031 A0D4 26 FA               BNE    CLRLIN
00032 A0D6 20 CA               BRA    PLOTIN    AND RETURN
00033      E1D1    OUTCH       EQU    $E1D1
00034      E0CC    OUTS        EQU    $E0CC
00035      E055    BYTE        EQU    $E055
00036      E21C    PLOT        EQU    $E21C
00037      0002    CPTR        EQU    2
00038                          END
```

SYMBOL TABLE

```
START  A092  CLRSCR A097  PLOTIN A0A2  CLRLIN A0D0  OUTCH  E1D1
OUTS   E0CC  BYTE   E055  PLOT   E21C  CPTR   0002
```

```
S00600004844521B
S111A092CEF8008620A700088CFBF826F8CE36
S113A0A0A080860DBDE1D1BDE0CCBDE055A705BDC6
S113A0B0E0CCBDE055A706BDE0CCBDE055E605E724
S113A0C002E606E703BDE21C8617A7036F02C62655
S10BA0D0BDE0CC5A26FA20CAB7
S9030000FC
```

```
00001                            NAM    BINL
00002 A092                       ORG    $A092
00003 A092 7C A090 START         INC    RUBIG     RUBOUT CHARS NOT IGNORED
00004 A095 8D 3F   XLOOK         BSR    INDAT     LOOK FOR 'X'
00005 A097 81 58                 CMP A  #'X
00006 A099 26 FA                 BNE    XLOOK     BRANCH IF NOT 'X'
00007 A09B 8D 39                 BSR    INDAT     LOOK FOR '1 OR '9
00008 A09D 81 31                 CMP A  #'1
00009 A09F 27 09                 BEQ    LOBIN     'X1' SO LOAD
00010 A0A1 81 39                 CMP A  #'9
00011 A0A3 26 F0                 BNE    XLOOK     LOOK AGAIN
00012 A0A5 8D 3D                 BSR    LFS       'X9' JUMP AND DO NOT ECHO
00013 A0A7 7E E00A               JMP    LOAD      CONTENTS OF REMAINDER OF TAPE
00014 A0AA 7F A00A LOBIN         CLR    CKSM      CLEAR CHECKSUM
00015 A0AD 8D 27                 BSR    INDAT     GET BYTE COUNT
00016 A0AF 16                    TAB
00017 A0B0 5C                    INC B            ACC B=BYTE COUNT
00018 A0B1 8D 23                 BSR    INDAT     GET MOST SIGNIFICANT BYTE OF
00019 A0B3 B7 A00C               STA A  XHI       AND STORE
00020 A0B6 8D 1E                 BSR    INDAT     GET LEAST SIG BYTE OF ADDRESS
00021 A0B8 B7 A00D               STA A  XLOW      AND STORE
00022 A0BB FE A00C               LDX    XHI       PUT THIS ADDRESS IN X
00023 A0BE 8D 16   STIN          BSR    INDAT     GET DATA
00024 A0C0 A7 00                 STA A  0,X       AND STORE
00025 A0C2 A1 00                 CMP A  0,X       DID IT STORE?
00026 A0C4 26 0B                 BNE    ERROR     IF NOT JUMP TO ERROR
00027 A0C6 08                    INX              INCREMENT X
00028 A0C7 5A                    DEC B            DECREMENT BYTE COUNT
00029 A0C8 26 F4                 BNE    STIN      BRANCH IF BYTE COUNT NOT ZERO
00030 A0CA 8D 0A                 BSR    INDAT     GET CHECKSUM BYTE
00031 A0CC 7C A00A               INC    CKSM
00032 A0CF 27 C4                 BEQ    XLOOK     BRANCH IF CHECKSUM OK
00033 A0D1 8D 11   ERROR         BSR    LFS       OUTPUT 8 LINE FEEDS
00034 A0D3 7E E040               JMP    LOAD19    AND PRINT '?' THEN GOTO TBUG
00035 A0D6 86 FF   INDAT         LDA A  #$FF      SUBROUTINE GET 8 BITS OF DATA
00036 A0D8 BD E1AE               JSR    INXBITS    INPUT AND UPDATE CHECKSUM
00037 A0DB 36                    PSH A
00038 A0DC BB A00A               ADD A  CKSM
00039 A0DF B7 A00A               STA A  CKSM
00040 A0E2 32                    PUL A
00041 A0E3 39                    RTS
00042 A0E4 CE A0EB LFS           LDX    #LFSTR    SUBROUTINE OUTPUTS 8 LF SO AS
00043 A0E7 BD E07E               JSR    PDATA     TO CLEAR SCREEN
00044 A0EA 39                    RTS
00045 A0EB 0D      LFSTR         FCB    $D,$A,$A,$A,$A,$A,$A,$A,$A,$4
      A0EC 0A
      A0ED 0A
      A0EE 0A
      A0EF 0A
      A0F0 0A
      A0F1 0A
      A0F2 0A
      A0F3 0A
      A0F4 04
```

```
00046       A090    RUBIG   EQU     $A090
00047       A00A    CKSM    EQU     $A00A
00048       A00C    XHI     EQU     $A00C
00049       A00D    XLOW    EQU     $A00D
00050       E07E    PDATA   EQU     $E07E
00051       E1AE    INXBIT  EQU     $E1AE
00052       E040    LOAD19  EQU     $E040
00053       E00A    LOAD    EQU     $E00A
00054               END
```

SYMBOL TABLE

```
START   A092  XLOOK   A095  LOBIN   A0AA  STIN    A0BE  ERROR   A0D1
INDAT   A0D6  LFS     A0E4  LFSTR   A0EB  RUBIG   A090  CKSM    A00A
XHI     A00C  XLOW    A00D  PDATA   E07E  INXBIT  E1AE  LOAD19  E040
LOAD    E00A
```

```
  S00600004844521B
  S111A0927CA0908D3F815826FA8D39813127AC
  S113A0A009813926F08D3D7EE00A7FA00A8D2716AE
  S113A0B05C8D23B7A00C8D1EB7A00DFEA00C8D16D1
  S113A0C0A700A100260B085A26F48D0A7CA00A27B3
  S113A0D0C48D117EE04086FFBDE1AE36BBA00AB759
  S113A0E0A00A3239FEA0EBBDE07E390D0A0A0A0A45
  S108A0F00A0A0A0A043B
  S9030000FC
```