



**SMOKE SIGNAL BROADCASTING**

# **SMARTBUG**

**An Intelligent Monitor for the 6800**

**COPYRIGHT © 1977  
SMOKE SIGNAL BROADCASTING**

## SMARTBUG - AN INTELLIGENT MONITOR FOR THE 6800

### INTRODUCTION

SMARTBUG is a 1024 byte monitor program which may be used in most systems using the Motorola 6800 microprocessor. It was designed primarily to replace the MIKBUG ROM used in many systems including the Southwest Technical Products 6800 microcomputer. SMARTBUG is available from SMOKE SIGNAL BROADCASTING on a 2708 EPROM. In order to implement SMARTBUG in the SWTPC 6800 microcomputer system, SMOKE SIGNAL BROADCASTING has developed the P-38 series of EPROM boards. These boards are equipped with SMARTBUG and contain room for seven more 2708's so that the user can expand the monitor at any time.

Most of the SMARTBUG subroutines start at the same address locations as the functionally equivalent MIKBUG subroutines. Thus, most programs designed to run with MIKBUG should require little, if any, modification to run with SMARTBUG.

One major advantage of SMARTBUG is that it is available on a 2708 Eraseable-Programmable Read Only Memory Chip. This means that the user may easily change the monitor to suit his individual system requirements simply by re-programming the 2708.

### WHY SMARTBUG?

SMARTBUG has several new features not found in MIKBUG which make system operation easier; however, these are not the primary reasons for SMARTBUG, but are added bonuses. MIKBUG handles serial I/O through the 6820 Parallel Interface Adapter which was designed for 8 bit parallel I/O and not serial I/O. MIKBUG requires the 6800 microprocessor to wait in timing loops while inputting or outputting data through the PIA. Thus, while the processor is writing a character, it cannot check to see if the user wishes to input a character at the same time. This limitation becomes quite noticeable to the user when trying to interrupt a program listing in BASIC (or any program that checks for user input while outputting data) by typing "CONTROL C". Many "C" keys have been worn out trying to get the program to recognize the user input. Also, while the processor is spinning its wheels in I/O timing loops, it cannot be doing any other work although this is usually unimportant except in real-time applications requiring fast servicing of interrupt requests.

SMARTBUG handles I/O through the 6850 Asynchronous Communications Interface Adapter. The 6850 was designed specifically to handle serial data. When writing data to the 6850, all

the microprocessor needs to do is check to see that the 6850 is ready to receive data and then write an 8 bit word to the 6850 in parallel form over the system data bus. This takes only a few instructions and very little time. While the 6850 is converting the parallel data received from the microprocessor to serial and sending it to the output device, it can simultaneously receive data. Thus, if you are running BASIC and type a "CONTROL C", the processor is "instantaneously" able to respond to your interrupt the first time you type "CONTROL C".

Another advantage of handling serial I/O through an ACIA is that baud rates in excess of 19,200 can be accommodated compared to a maximum baud rate of about 1200 baud that can be handled by MIKBUG.

### WHY MIKBUG?

To the experienced hobbieist who has used MIKBUG, the limitations of handling serial I/O through a 6820 parallel I/O chip are intuitively obvious. If you are a newcomer, just accept it on faith that only a very strange person would use a 6820 for serial I/O instead of a 6850 in a general purpose microcomputer system. The question that is often asked is: "Why would a big company like Motorola do such a silly thing?". The answer is that in 1974 when MIKBUG was written, the 6850 was not yet in production and the 6820 was. In order to introduce the first 6800 evaluation kit, it was necessary to handle the serial I/O through a 6820 and MIKBUG was a very clever little device used to demonstrate how easy it was to use the 6800 microprocessor.

### HARDWARE REQUIREMENTS

SMARTBUG "talks" through a 6850 ACIA which should be located at \$8008 and \$8009. It also requires RAM at \$A000 through \$A06B. SMARTBUG itself is located at \$E000 through \$E3FF. In order to have the reset and interrupt vectors operate without external ROM, it is necessary to have SMARTBUG located at \$FC00 through \$FFFF in addition to \$E000 through \$E3FF. The SMOKE SIGNAL BROADCASTING P-38 series of EPROM boards has a switch to allow SMARTBUG to occupy both of these areas or only the \$E000 through \$E3FF area when using another 2708 in the \$FC00 through \$FFFF area. To locate a 6850 ACIA at \$8008 and \$8009, owners of the SWTPC 6800 should purchase a MP-S board and place it in I/O slot number 2. The MP-C control board in slot number 1 is no longer used and should be removed from the machine.

### USE OF HIGH BAUD RATES

The maximum baud rate useable with MIKBUG is about 1200 baud. With SMARTBUG, it is possible to use baud rates of at least

19,200; however, for baud rates in excess of about 1200 baud, it may be necessary to change the crystal in the SWTPC 6800. The MC14411P baud rate generator chip used in the SWTPC 6800 is designed to use a crystal frequency of 1.8432 MHz. The crystal supplied with the SWTPC 6800 is a few percent lower in frequency due to an anomaly of MIKBUG. If you wish to take full advantage of SMARTBUG and use its high baud rate capability, you may need a crystal of the correct frequency. Also, it will be necessary to bring the desired baud rate line out from the baud rate generator chip on the CPU board in place of one of the lower baud rates that you are not using. This requires a foil cut and jumper on the CPU card. Consult the CPU card instruction manual and the MC14411P data sheet to determine the correct locations for your particular application.

## SOFTWARE OPERATION

### RESET

Pressing the reset button on the SWTPC 6800 will cause SMARTBUG to output a carriage return, line feed and an asterisk (\*) to the system terminal. As in MIKBUG, the asterisk is the prompt character; and, when it appears, SMARTBUG is waiting for the user to enter a command. One of the advantages of having your monitor in EPROM is that you are able to customize the monitor to your system. When SMARTBUG prompts with an asterisk, it is actually outputting the character string located at \$E3F0 through \$E3F6. If you are using a non-scrolling terminal such as the CT-1024, you may wish to change one of the null (00) characters in this string to an "Erase to End of Line" character (\$15 in the case of the CT-1024).

### COMMANDS

After being prompted with an asterisk, the user may enter any valid SMARTBUG command. All SMARTBUG commands are single-letter commands followed, in some cases, by address information. The valid command letters are A, B, C, D, E, G, H, I, J, K, L, M, N, P, Q, R, T, X, 4. Entering any other character will cause SMARTBUG to prompt again with an asterisk.

### "R" REGISTERS

Typing "R" will cause SMARTBUG to display the various registers in the 6800 in the following format.

\*R CC BB AA XXXX PCPC SPSP

Throughout this manual, user input is indicated by underlined characters. Output from SMARTBUG is not underlined.

CC is the two hex digits representing the contents of the  
Condition Code Register  
BB is the contents of the B Accumulator  
AA is the contents of the A Accumulator  
XXXX is the contents of the Index Register (4 hex digits)  
PCPC is the contents of the Program Counter  
SPSP is the contents of the Stack Pointer

"A" EXAMINE AND CHANGE THE A ACCUMULATOR

Entering an "A" after the \* prompt character will cause the contents of the A Accumulator to be displayed. To change the contents of the A Accumulator, simply type two hex characters. Type a carriage return to return to SMARTBUG without altering the contents of the A Accumulator. A sample format is shown below.

\*A XX YY

where XX is the old contents of the A Accumulator and YY is the new contents entered by the user.

"B" EXAMINE AND CHANGE THE B ACCUMULATOR

"B" allows the user to examine and change the contents of the B Accumulator and operates in the same manner as the "A" command.

"C" EXAMINE AND CHANGE THE CONDITION CODE REGISTER

"C" allows the user to examine and change the Condition Code Register and operates in the same manner as the "A" command.

"X" EXAMINE AND CHANGE THE INDEX REGISTER

"X" allows the user to examine and change the contents of the Index Register. This command operates in the same manner as the "A" command except that four hex characters are required for the "X" command instead of two.

"M" MEMORY EXAMINE AND CHANGE

The "M" command allows the user to examine any memory location and to change any memory location occupied by RAM memory. To examine a memory location, type "M" followed by the four hex digits of the memory location you wish to examine.

EXAMPLE:    \*M 0100  
              \*0100 7E BD  
              \*0101 E1

In the above example, the user typed "M" followed by "0100". SMARTBUG then typed \*0100 7E. 7E was the old contents of 0100. The user then typed "BD", thus changing the contents of 0100 to BD. SMARTBUG then proceeded to show the user the contents of 0101.

To change a memory location, it is only necessary to type the two hex digits representing the new data. To return to SMARTBUG without changing the data, type a carriage return. To examine the following location without changing the present location, hit the SPACE BAR. To examine the previous memory location, type "U" for up.

#### GO TO USER'S PROGRAM

Two commands are provided to transfer control from SMARTBUG to a user's program. The "G" command which operates in the same manner as the "G" command in MIKBUG and a new "J" command.

#### "G" GO TO LOCATION CONTAINED IN \$A048 and \$A049

To use the "G" command, first use the "M" command to put the starting address of the program into memory locations \$A048 and \$A049. Then type "G". SMARTBUG will then jump to the location contained in \$A048 and \$A049. This command is useful when you will enter the program several times from SMARTBUG. When you only intend to enter the program from SMARTBUG once, the "J" command is more convenient.

#### "J" JUMP TO LOCATION XXXX

Typing "J" "XXXX" where XXXX are four hex digits will cause SMARTBUG to transfer program control to that location. EXAMPLE: \*J 01A0 will cause SMARTBUG to jump to \$01A0 and begin executing whatever program was previously stored beginning at that location.

#### "I" INSERT

FORMAT: I XXXX YYYY ZZ    EXAMPLE: \*I 0000 3FFF 3F  
This command will insert the two hex digits "ZZ" into memory locations "XXXX" through "YYYY". In the example, memory locations \$0000 through \$3FFF will now contain \$3F. In debugging a new program, it is often desirable to store \$3F (software interrupt) in all your memory prior to loading and executing the program. If the program inadvertently

transfers outside the program area, it will encounter a software interrupt, display the CPU registers and return to SMARTBUG. This command can also be used to clear blocks of memory by storing "00" into specified areas of memory.

#### "Q" QUICKSTART

This command is for the convenience of those people using the SMOKE SIGNAL BROADCASTING BFD-68 Disc System. Typing "Q" does the same thing as typing "J 8020". SMARTBUG transfers control to \$8020 which is the beginning address of the routine that boots in the disc operating system from a cold start.

#### "D" DISC

Typing "D" transfers control from SMARTBUG to \$7283 which is the warmstart address of DOS68, the disc operating system used with the BFD-68 disc system. This provides a convenient means of re-entering the DOS68 monitor from SMARTBUG when DOS68 has previously been booted in from disc and is resident in memory. Those people using the optional version of DOS68 located between D000 and DFFF will want to re-program the 2708 and change location \$E3DF from \$72 to \$D2. Typing "D" will then transfer program control to \$D283 which is the warmstart address of the optional version of DOS68.

#### "E" ECHO, "N" NO-ECHO, "H" HARDCOPY

RAM location \$A00B is a "flag" location that determines whether INEEE will echo back characters typed on the terminal and whether OUTEEE will output to the system terminal connected to I/O port number 2 (ACIA at \$8008 and \$8009) or jump to an external output routine. The external output routine would normally be a routine to drive a hardcopy printer. INEEE is a subroutine located at \$E1AC that waits for a character input from the system console and returns that character input in the A accumulator. OUTEEE is located at \$E1D1 and causes a character in the A accumulator to be transmitted to the system console (or to the external print routine).

When hitting "RESET" or otherwise entering SMARTBUG at \$E0D0, location \$A00B is cleared. Typing an "E" will also clear this location. When location \$A00B contains a "00", all input through INEEE will be echoed through the system console and calls to OUTEEE will result in output to the system console and not a jump to an external printer output routine.

NOTE: Many programs have been written that re-enter MIKBUG upon completion of the program at "START" location \$E0D0.

Normally, it is better to re-enter MIKBUG or SMARTBUG at "CONTRL" location \$E0E3. Entering at "CONTRL" will not re-initialize \$A00B to the ECHO mode, but will leave it in the mode last selected by the user or the user's program. This is usually more desirable. While MIKBUG does not have an echo control feature, there are some other reasons why it is usually better to re-enter MIKBUG or SMARTBUG at \$E0E3 rather than \$E0D0. Also, remember that hitting "RESET" restores the echo. Unless this is your desired mode of operation, you will have to type "N" or "H" after pressing "RESET".

Any positive number (\$01 through \$7F) stored in \$A00B will cause INEEE not to echo the character inputted through INEEE and OUTEEE will not jump to an external print routine. Typing "N" stores a \$4E in location \$A00B and, thus, suppresses the echo.

Any negative number (\$80 through \$FF) stored in \$A00B will cause OUTEEE to jump to \$A04A before anything is transmitted to the terminal device. Typing "H" stores a \$B8 in location \$A00B and, therefore, will cause OUTEEE to jump to \$A04A. Any user wishing to use the "H" command will have to put a jump to his printer routine location in location \$A04A, \$A04B and \$A04C prior to using this feature. Those SMARTBUG users having a SMOKE SIGNAL BROADCASTING P-38 series EPROM board will probably want to put their printer routine in EPROM. Then the printer routine will always be available without having to load it into RAM each time the system is powered up. The next EPROM location available on the P-38 board is \$E400 through \$E7FF. We suggest standardizing on \$E600 as the beginning location of the print routine. This leaves \$E400 through \$E5FF available for extended monitor routines. If you do put your printer routine at \$E600, you will probably want to change SMARTBUG location \$E1D7 from \$A0 to \$E6 and location \$E1D8 from \$4A to \$00. This will cause OUTEEE to jump directly to your routine at \$E600 instead of to \$A04A. This again points out the advantage of having the system monitor in EPROM rather than ROM. With EPROM, it is easy to customize the system monitor to your unique system requirements.

If you want OUTEEE to output both to the system console as well as to your separate hardcopy device when in the "H" mode, your print routine should end with a jump to \$E1D9. Otherwise, it should end with a "RTS" (\$39).

#### CONTROL OF THE ECHO FUNCTION FROM THE USER'S PROGRAM

Several programs such as BASIC and DOS68 turn the MIKBUG echo off prior to jumping to INEEE and restore the echo upon return. This allows the program to echo control characters and other normally non-printable characters. This is also probably the only major area where SMARTBUG and MIKBUG are not compatible. In MIKBUG, the echo is suppressed by storing a \$3C in location \$8007 and is restored by storing

\$34 in location \$8007. Running a program that suppresses the MIKBUG echo in SMARTBUG without first modifying the echo handling routine will result in the input being double echoed unless you type a "N" prior to entering these programs. For frequently used programs, it will probably be more convenient to modify them than to remember to type "N".

To modify an existing program, we suggest that you change the instructions storing a \$3C in \$8007 to an "INC \$A00B" (7C A0 0B) and that the instruction storing a \$34 in \$8007 be changed to a "DEC \$A00B" (7A A0 0B). NOP's (\$01) should be used to fill in the extra area used by the previous instructions.

In DOS68, the echo control is found in the ZLINEI routine. The jump to ZLINEI is found in the jump table at \$72B5 (or \$D2B5). Echo is turned off by the instruction sequence 86 3C B7 80 07 and turned back on by the sequence 86 34 B7 80 07. These sequences should be changed to 7C A0 0B 01 01 and 7A A0 0B 01 01 respectively. The exact location of the ZLINEI routine may vary with different versions of DOS68, but the jump table location will remain the same. This is why we ask you to go to the jump table to find ZLINEI and search through ZLINEI for this instruction sequence rather than specify the locations to be changed.

By using an increment-decrement scheme to control the echo, the user now has control of the echo even if he has selected the "H" HARDCOPY function prior to entering his program. The first part of the printer routine should test to see if \$A00B contains a \$B8. If it does, the routine should output data given it. If it contains a \$B9, the routine should do a "RTS" without outputting the data.

#### "P" PUNCH FORMATTED TAPE

EXAMPLE: \*P 0100 0150

The above example will cause SMARTBUG to punch a formatted tape containing the data in memory locations \$0100 through \$0150. The tape format is the same as the MIKBUG format and S9 is not punched at the end. This way, several areas of memory may be punched on one tape and loaded with one "L" command. At the end of the last area of memory to be punched to the tape, the user should manually type a S9 to the tape so that the "L" command will function automatically.

#### "L" LOAD FORMATTED TAPE

Typing "L" will turn on the system tape reader and read formatted tape produced by the "P" command. If the tape does not contain a S9 as an end of file indicator, it will be necessary for the user to manually type a S9 on the system console after the tape has been read in order to return to

SMARTBUG. The S9 causes SMARTBUG to be entered at "CONTRL". This is to be preferred over hitting "RESET" which causes entry at "START".

Unlike MIKBUG, SMARTBUG normally echoes the tape input. If the user wishes to suppress the echo when loading tape, he should type "N" prior to typing "L".

#### "4" JUMP TO \$E400

Typing a "4" will cause SMARTBUG to jump to \$E400. This command allows users of the SMOKE SIGNAL BROADCASTING P-38 series boards to expand their SMARTBUG monitor to include additional commands by installing another 2708 EPROM in the \$E400 through \$E7FF socket on the board. The user can accomodate additional commands by having a routine starting at \$E400 that asks for an additional character input and then executes whatever command is specified by that second character. Using this approach, all regular SMARTBUG commands would continue to be one character commands and all extended commands would be two character commands with the number "4" being the first character.

We would very much appreciate a copy of any extended commands you may develop. Naturally, we would prefer a fully-commented source listing; however, don't be embarrassed to send just the object code along with a brief functional description. After all, it seems most of us write programs first and document them later (and then, only if absolutely necessary).

#### "K" BREAKPOINT

The "K" command is a tool to allow the programmer to step through his program a few steps at a time in order to inspect his program at these intermediate steps to see if the program is, indeed, operating as it was so carefully designed to do. To use the "K" command, first load the starting address of the program into memory locations \$A048 and \$A049 using the "M" command. Next decide where you want the first breakpoint. Then type "K" followed by the four hex digits representing the address at which the breakpoint is to be inserted. After entering the fourth digit, SMARTBUG will jump to the location previously stored in \$A048 and \$A049 and execute the program until it encounters the breakpoint (if it ever does). When the breakpoint is encountered, SMARTBUG will display the contents of the registers in the same format as the "R" command. To continue the program at the point it was interrupted, simply type "G". To pick up at this point and continue to a second breakpoint, type "K" followed by a new breakpoint address.

SMARTBUG uses the "SWI" (\$3F) instruction to set a breakpoint; thus, a breakpoint may not be set in an area of Read-Only-Memory. SMARTBUG remembers the instruction stored in the breakpoint location and automatically restores that instruction

after encountering the breakpoint. If the program "gets lost" and the breakpoint is not encountered, the instruction will not be restored and will have to be manually restored by the user.

"T" TRACE MODE

Typing a "T" followed by a four digit hexadecimal address puts SMARTBUG in the single-step trace mode. This allows the user to step through a program in RAM one step at a time and to examine and change the registers after each step. Stepping to a ROM location will cause SMARTBUG to return to the regular command mode and prompt with an asterisk. After typing "T" followed by four hex digits, SMARTBUG will type the current contents of the registers followed by the specified address and the command to be executed at that address. No asterisk prompt character is issued which indicates that SMARTBUG is in the TRACE mode. Prior to executing the next instruction, the user may change the A, B, C or X registers with the A, B, C or X commands. When ready to execute the next instruction, hit the SPACE BAR. To return to the regular SMARTBUG mode, hit the carriage return. Following is the trace output from a very short program.

```
MEMORY CONTENTS:  0100 86
                   0101 43
                   0102 BD
                   0103 01
                   0104 D1
                   0105 86
                   0106 55
                   0107 3F
                   01D1 39
```

```
*T 0100
F0 33 00 E26E 0100 A049
0100 86 43
SPACEBAR
F0 33 43 E26E 0102 A049
0102 BD 01D1
SPACEBAR
F0 33 43 E26E 01D1 A047
01D1 39
B 33 48
SPACEBAR
F0 48 43 E26E 0105 A049
0105 86 55
SPACEBAR
F0 48 55 E26E 0107 A049
0107 3F
SPACEBAR
*
```

The format for the listing of the register contents is the same as in the "R" command.

## IRQ AND NMI

If the system encounters an IRQ interrupt request, it will jump to the location contained in memory locations \$A000 and \$A001. An NMI interrupt will cause SMARTBUG to jump to the location contained in memory locations \$A006 and \$A007. If the user anticipates these types of interrupts, he should initialize these locations early in his program. Alternately, he can re-program the vector locations in SMARTBUG to go to permanent interrupt handling routines in his system.

## COMPATIBILITY WITH MIKBUG

Every reasonable effort was made to keep the subroutines in SMARTBUG at the same beginning address locations as the functionally equivalent subroutines in MIKBUG so that programs written for MIKBUG would run in SMARTBUG without modification. As shown in the list below, all the locations of the most frequently used routines are maintained.

THE FOLLOWING LABELS IN SMARTBUG ARE FUNCTIONALLY EQUIVALENT TO THOSE IN MIKBUG AND ARE LOCATED AT THE SAME ADDRESS LOCATIONS.

IO	POWDWN	LOAD	LOAD3	LOAD11	LOAD15
LOAD19	LOAD21	C1	BADDR	BYTE	OUTH1
OUTH1	OUTCH	INCH	PDATA2	PDATA1	CHANGE
CHA51	INHEX	IN1HG	OUT 2H	OUT2HA	OUT4HS
OUT2HS	OUTS	START	CONTRL	SFE	INEEE
OUTEEE	IOV	BEGA	ENDA	NIO	SP
XHI	XLOW	TEMP	TW	XTEMP	STACK

THE FOLLOWING LOCATIONS IN MIKBUG ARE NOT FOUND AT THE SAME LOCATIONS IN SMARTBUG AND THERE MAY BE NO FUNCTIONALLY EQUIVALENT LABEL IN SMARTBUG.

PRINT	C2	MTAPE1	PUNCH	PUN11	PUN22
PUN23	PUN32	PUNT2	MCLOFF	MCL	SAV
IN1	IN3	IOUT	OUT1	IOUT2	IOS
DEL	DE	CKSM	BYTECT	MCONT	

## LIMITED WARRANTY

Any purchaser of SMARTBUG who is not satisfied with its performance may return his copy within 10 days from date of purchase for a full refund. This warranty is in lieu of all other warranties express or implied. SMOKE SIGNAL BROADCASTING does not warrant the suitability of SMARTBUG for any particular user application and will not be responsible for damages incidental to its use in a user system.

### LICENSE CONDITIONS

Purchase of a P-38 series board which includes SMARTBUG or purchase of a SMARTBUG listing conveys to the purchaser a license to copy SMARTBUG for his own use, and not for sale or free distribution to others. No other license, express or implied, is conveyed.

### LIMERICK

Mary had a little plane.  
She flew it high and brisk.  
Wasn't she a silly girl,  
her little \*

### USER CONTRIBUTIONS

Any user wishing to contribute program or limerick improvements should send them to:

SMOKE SIGNAL BROADCASTING  
P.O. BOX 2017  
HOLLYWOOD, CA 90028

We are particularly interested in extended monitor commands for possible inclusion in a future 2K or 4K monitor program. Worthwhile contributions will also be published in future newsletters with credit to the author.

```

00100          NAM    SMARTBUG

00120          *      "SMARTBUG" - AN INTELLIGENT MONITOR
00130          *      COPYRIGHT 1977 SMOKE SIGNAL BROADCASTING

00150          OPT    O,S
00160          8008  ACIAS EQU  $8008
00170          8009  ACIAD EQU  $8009
00180 E000        ORG    $E000

00200          * I/O INTERRUPT SEQUENCE
00210 E000 FE A000 IO    LDX    IOV
00220 E003 6E 00        JMP    X

00240          * NMI SEQUENCE
00250 E005 FE A006 POWDWN LDX    NIO    GET NMI VECTOR
00260 E008 6E 00        JMP    X    GO TO NMI LOCATION

00280          * LOAD ASCII FORMATTED TAPE
00290          E00A  LOAD    EQU    *
00300 E00A 86 55        LDA    A    #$55    READER RELAY ON, ONE STOP BIT
00310 E00C B7 8008      STA    A    ACIAS
00320 E00F 86 11        LDA    A    #$11
00330 E011 8D 62        BSR    OUTCH    AC-30 READ CTRL
00340 E013 8D 63        LOAD3 BSR    INCH    GET CHARACTER
00350 E015 81 53        CMP    A    #'S    IS IT AN "S"
00360 E017 26 FA        BNE    LOAD3    NO-LOOP TILL "S" FOUND
00370 E019 8D 5D        BSR    INCH    YES - GET NEXT CHARACTER
00380 E01B 81 39        CMP    A    #'9    IS IT A "9"
00390 E01D 27 25        BEQ    LOAD21    YES - JUMP TO CONTROL
00400 E01F 81 31        CMP    A    #'1    IS IT A "1"
00410 E021 26 F0        BNE    LOAD3    NO - TRY AGAIN
00420 E023 7F A06A      CLR    CKSM    YES - ZERO CHECKSUM
00430 E026 8D 2D        BSR    BYTE    GET A BYTE
00440 E028 80 02        SUB    A    #2
00450 E02A B7 A06B      STA    A    BYTECT    READ THIS MANY BYTES
00460          * BUILD ADDRESS
00470 E02D 8D 18        BSR    BADDR
00480          * STORE DATA
00490 E02F 8D 24        LOAD11 BSR    BYTE    READ NEXT BYTE
00500 E031 7A A06B      DEC    BYTECT    DECREMENT BYTE COUNTER
00510 E034 27 05        BEQ    LOAD15    IF 0, GET NEXT LINE
00520 E036 A7 00        STA    A    X    ELSE, STORE DATA
00530 E038 08          INX
00540 E039 20 F4        BRA    LOAD11
00550 E03B 7C A06A      LOAD15 INC    CKSM    FORM 2'S COMPLEMENT
00560 E03E 27 D3        BEQ    LOAD3    IT SHOULD BE ZERO
00570 E040 86 3F        LOAD19 LDA    A    #'?    READ ERROR - PRINT
00580 E042 8D 31        BSR    OUTCH    QUESTION MARK
00590          E044  LOAD21 EQU    *
00600 E044 7E E0E3 C1  JMP    CONTRL

00620          * BUILD ADDRESS
00630 E047 8D 0C        BADDR BSR    | BYTE    READ 2 BYTES

```

00640	E049	B7	A00C	STA	A	XHI	AND RETURN FROM THIS
00650	E04C	8D	07	BSR		BYTE	SUBROUTINE WITH BOTH
00660	E04E	B7	A00D	STA	A	XLOW	BYTES IN THE INDEX
00670	E051	FE	A00C	LDX		XHI	REGISTER.
00680	E054	39		RTS			
00700							
							* INPUT BYTE (2 HEX CHARACTERS)
00710	E055	8D	53	BYTE	BSR	INHEX	GET 1ST HEX CHAR
00720	E057	48		ASL	A		
00730	E058	48		ASL	A		
00740	E059	48		ASL	A		
00750	E05A	48		ASL	A		
00760	E05B	16		TAB			
00770	E05C	8D	4C	BSR		INHEX	GET 2ND HEX CHAR
00780	E05E	1B		ABA			
00790	E05F	16		TAB			
00800	E060	FB	A06A	ADD	B	CKSM	UPDATE CHECKSUM AND
00810	E063	F7	A06A	STA	B	CKSM	RETURN WITH BYTE IN
00820	E066	39		RTS			A ACCUMULATOR
00840	E067	44		OUTHL	LSR	A	OUT HEX LEFT BCD DIGIT
00850	E068	44		LSR	A		
00860	E069	44		LSR	A		
00870	E06A	44		LSR	A		
00890	E06B	84	0F	OUTHR	AND	A	OUT HEX RIGHT BCD DIGIT
00900	E06D	8B	30	ADD	A	#\$F	
00910	E06F	81	39	ADD	A	#\$30	
00920	E071	23	02	CMP	A	#\$39	
00930	E073	8B	07	BLS		OUTCH	
00940	E075	7E	E1D1	ADD	A	#7	
00950	E078	7E	E1AC	OUTCH	JMP	OUTEEE	OUTPUT A CHARACTER
				INCH	JMP	INEEE	INPUT A CHARACTER
00970							
							* PRINT DATA POINTED TO BY INDEX REGISTER
00980	E07B	8D	F8	PDATA2	BSR	OUTCH	
00990	E07D	08			INX		
01000	E07E	A6	00	PDATA1	LDA	A	X
01010	E080	81	04		CMP	A	#4
01020	E082	26	F7		BNE		PDATA2
01030	E084	39			RTS		END OF STRING CHARACTER
01050							
							* CHANGE MEMORY
01060	E085	8D	C0	CHANGE	BSR	BADDR	GET MEMORY ADDRESS
01070	E087	CE	E3F1	CHA51	LDX	#MCL	
01080	E08A	8D	F2		BSR	PDATA1	PRINT C/R L/F
01090	E08C	CE	A00C		LDX	#XHI	
01100	E08F	8D	37		BSR	OUT4HS	PRINT ADDRESS
01110	E091	FE	A00C		LDX	XHI	
01120	E094	8D	34		BSR	OUT2HS	PRINT OLD DATA
01130	E096	FF	A00C		STX	XHI	
01140	E099	8D	DD		BSR	INCH	INPUT A CHARACTER
01150	E09B	81	20		CMP	A	IF IT'S A SPACE
01160	E09D	27	E8		BEQ	CHA51	GET NEXT ADDRESS
01170	E09F	7E	E3AD		JMP	TDEX	ELSE - GO TO TDEX

01190	EOA2	A7	00	CHA61	STA	A	X	STORE NEW DATA
01200	EOA4	A1	00		CMP	A	X	DID IT STORE CORRECTLY?
01210	EOA6	27	DF		BEQ		CHA51	YES - GET NEXT ADDRESS
01220	EOA8	20	96		BRA		LOAD19	NO - JUMP CONTROL
01240	* INPUT HEX CHARACTER							
01250	EOAA	8D	CC	INHEX	BSR		INCH	
01260	EOAC	80	30		SUB	A	#\$30	
01270	EOAE	2B	94		BMI		C1	NOT HEX, JUMP CONTROL
01280	EOB0	81	09		CMP	A	#9	
01290	EOB2	2F	0A		BLE		IN1HG	
01300	EOB4	81	11		CMP	A	#\$11	
01310	EOB6	2B	8C		BMI		C1	NOT HEX
01320	EOB8	81	16		CMP	A	#\$16	
01330	EOBA	2E	88		BGT		C1	NOT HEX
01340	EOBC	80	07		SUB	A	#7	
01350	EOBE	39		IN1HG	RTS			
01370	EOBF	A6	00	OUT2H	LDA	A	X	OUTPUT 2 HEX CHAR
01380	EOC1	8D	A4	OUT2HA	BSR		OUTH1	OUT LEFT HEX CHAR
01390	EOC3	A6	00		LDA	A	X	
01400	EOC5	08			INX			
01410	EOC6	20	A3		BRA		OUTHR	OUTPUT RIGHT HEX CHAR
01430	EOC8	8D	F5	OUT4HS	BSR		OUT2H	OUTPUT 4 HEX CHAR AND SPACE
01440	EOCA	8D	F3	OUT2HS	BSR		OUT2H	OUTPUT 2 HEX CHAR AND SPACE
01450	EOCC	86	20	OUTS	LDA	A	#\$20	OUTPUT SPACE
01460	EOCE	20	A5		BRA		OUTCH	
01480	* POWER ON SEQUENCE							
01490	EOD0			START	EQU		*	
01500	EOD0	8E	A042		LDS		#STACK	
01510	EOD3	BF	A008		STS		SP	
01520	EOD6	7F	A00B		CLR		ECHO	ECHO ALL INPUT CHARACTERS
01530	EOD9	86	03		LDA	A	#3	MASTER RESET OF ACIA
01540	EODB	B7	8008		STA	A	ACIAS	
01550	EODE	86	15	INZ	LDA	A	#\$15	SET UP FOR 1 STOP BIT
01560	EOE0	B7	A00A	INZ1	STA	A	ACIAT	
01570	EOE3	B6	A00A	CONTRL	LDA	A	ACIAT	ALLOW FOR SOFTWARE CONTROL
01580	EOE6	B7	8008		STA	A	ACIAS	OF ACIA CONTROL REGISTER
01590	EOE9	8E	A042		LDS		#STACK	
01600	EOEC	7F	A011		CLR		TFLAG	TURN OFF TRACE MODE
01610	EOEF	CE	E3F0		LDX		#MCLOFF	
01620	EOF2	8D	8A		BSR		PDATA1	
01640	EOF4	8D	82		BSR		INCH	INPUT COMMAND CHARACTER
01650	EOF6	7F	A014		CLR		BKFLG	CLEAR BREAKPOINT INDICATOR
01660	EOF9	16			TAB			
01670	EOFA	8D	D0		BSR		OUTS	
01680	EOFC	CE	E3C3		LDX		#FUTABL	DO TABLE LOOKUP
01690	EOFF	E1	00	NXTCHR	CMP	B	0,X	FOR COMMAND FUNCTIONS
01700	E101	27	0B		BEQ		GOODCH	MATCH FOUND
01710	E103	08			INX			NO MATCH-INC TO NEXT COMMAND
01720	E104	08			INX			

01730	E105	08		INX		
01740	E106	8C	E3F0	CPX	#TBLEND	END OF COMMAND TABLE?
01750	E109	26	F4	BNE	NXTCHR	NO - GET NEXT CHARACTER
01760	E10B	7E	E2D9	JMP	CKCBA	YES - CHECK FOR A,B,C,X CMNDS
01770	E10E	EE	01	GOODCH LDX	1,X	GET COMMAND LOCATION
01780	E110	6E	00	JMP	0,X	AND JUMP THERE
01790	E112	01		NOP		KEEP SFE AT \$E113

01810						* ENTER FROM SOFTWARE INTERRUPT	
01820	E113	BF	A008	SFE	STS	SP	SAVE PROGRAM'S STACK POINTER
01830							* DECREMENT PROGRAM COUNTER
01840	E116	30		TSX			
01850	E117	6D	06	TST	6,X		
01860	E119	26	02	BNE	*+4		
01870	E11B	6A	05	DEC	5,X		
01880	E11D	6A	06	DEC	6,X		
01890	E11F	7D	A011	TST	TFLAG		
01900	E122	27	63	BEQ	PRNT		IF TRACE IS OFF
01910	E124	7E	E38C	JMP	SWTURN		IF TRACE IS ON

01930 \* PUNCH - OUTPUT HEX FORMATTED TAPE

01950	E127	8D	74	PUNCH	BSR	LIMITS	GET LIMITS
01960	E129	86	12		LDA	A #12	AC-30 CONTRL
01970	E12B	BD	E075		JSR	OUTCH	
01980	E12E	FE	A002		LDX	BEGA	THE "P" COMMAND JUMPS TO
01990	E131	FF	A00F		STX	TW	PUNCH AFTER USING THE LIMITS
02000	E134	B6	A005	PUN11	LDA	A ENDA+1	SUBROUTINE TO ENTER THE
02010	E137	B0	A010		SUB	A TW+1	START AND STOP ADDRESSES
02020	E13A	F6	A004		LDA	B ENDA	
02030	E13D	F2	A00F		SBC	B TW	
02040	E140	26	04		BNE	PUN22	
02050	E142	81	10		CMP	A #16	
02060	E144	25	02		BCS	PUN23	
02070	E146	86	0F	PUN22	LDA	A #15	
02080	E148	8B	04	PUN23	ADD	A #4	
02090	E14A	B7	A064		STA	A MCONT	FRAME COUNT THIS RECORD
02100	E14D	80	03		SUB	A #3	
02110	E14F	B7	A00E		STA	A TEMP	BYTE COUNT THIS RECORD
02120							* PUNCH C/R,L/F,NULL,S,1
02130	E152	8D	77		BSR	CRLF	
02140	E154	08			INX		
02150	E155	8D	77		BSR	PDAT1	
02160	E157	5F			CLR	B	ZERO CHECKSM
02170							* PUNCH FRAME COUNT
02180	E158	CE	A064		LDX	#MCONT	
02190	E15B	8D	25		BSR	PUNT2	PUNCH 2 HEX CHAR
02200							* PUNCH ADDRESS
02210	E15D	CE	A00F		LDX	#TW	
02220	E160	8D	20		BSR	PUNT2	
02230	E162	8D	1E		BSR	PUNT2	
02240							* PUNCH DATA
02250	E164	FE	A00F		LDX	TW	
02260	E167	8D	19	PUN32	BSR	PUNT2	PUNCH ONE BYTE

02270	E169	7A	A00E		DEC	TEMP	DECREMENT ONE BYTE
02280	E16C	26	F9		BNE	PUN32	
02290	E16E	FF	A00F		STX	TW	
02300	E171	53			COM	B	
02310	E172	37			PSH	B	
02320	E173	30			TSX		
02330	E174	8D	0C		BSR	PUNT2	PUNCH CHECKSUM
02340	E176	33			PUL	B	RESTORE STACK
02350	E177	FE	A00F		LDX	TW	
02360	E17A	09			DEX		
02370	E17B	BC	A004		CPX	ENDA	
02380	E17E	26	B4		BNE	PUN11	
02390	E180	20	47		BRA	C3	GO TO CONTROL
02400	E182	EB	00	PUNT2	ADD	B	X
02410	E184	7E	E0BF		JMP	OUT2H	
02420	E187	20	61	PRNT	BRA	PRINT	
02440	E189	8D	36	BKPNT	BSR	BAD2	GET BREAKPOINT ADDRESS
02450	E18B	FF	A068		STX	PB2	
02460	E18E	A6	00		LDA	A	X
02470	E190	B7	A014		STA	A	BKFLG
02480	E193	86	3F		LDA	A	#\$3F
02490	E195	A7	00		STA	A	X
02500	E197	8D	32		BSR	CRLF	SET BREAKPOINT
02510	E199	BE	A008	CONTG	LDS	SP	RESTORE PGM'S STACK POINTER
02520	E19C	3B			RTI		GO TO USER'S PROGRAM
02540	E19D	8D	22	LIMITS	BSR	BAD2	GET FIRST ADDRESS
02550	E19F	FF	A002		STX	BEGA	
02560	E1A2	8D	05		BSR	OUS	OUTPUT A SPACE
02570	E1A4	8D	1B		BSR	BAD2	GET SECOND ADDRESS
02580	E1A6	FF	A004		STX	ENDA	
02590	E1A9	7E	E0CC	OUS	JMP	OUTS	OUTPUT A SPACE & RETURN
02610							* INPUT ONE CHARACTER INTO A ACCUMULATOR
02620	E1AC	B6	8008	INEEE	LDA	A	ACIAS
02630	E1AF	47			ASR	A	
02640	E1B0	24	FA		BCC	INEEE	
02650	E1B2	B6	8009		LDA	A	ACIAD
02660	E1B5	84	7F		AND	A	#\$7F
02670	E1B7	81	7F		CMP	A	#\$7F
02680	E1B9	27	F1		BEQ	INEEE	IGNORE RUBOUTS
02690	E1BB	7D	A00B		TST	ECHO	
02700	E1BE	2F	11		BLE	OUTEEE	
02710	E1C0	39			RTS		
02730	E1C1	7E	E047	BAD2	JMP	BADDR	GET ADDRESS
02750	E1C4	5F		ECHON	CLR	B	ECHO ALL INPUT CHARACTERS
02760	E1C5	50		PRNTON	NEG	B	TURN PRINTER ON
02770	E1C6	F7	A00B	ECHOFF	STA	B	ECHO
02780	E1C9	20	41	C3	BRA	C2	GO TO CONTROL
02800	E1CB	CE	E3A4	CRLF	LDX	#\$CRLFAS	C/R L/F WITHOUT * PROMPT

```

02810 E1CE 7E E07E PDAT1 JMP PDATA1 SIGNIFIES TRACE MODE

02830 * OUTPUT ONE CHARACTER FROM A-REG
02840 E1D1 7D A00B OUTEEE TST ECHO IF ECHO IS NEGATIVE,
02850 E1D4 2C 03 BGE OUTCH2 GO TO PRINTER ROUTINE.
02860 E1D6 7E A04A JMP PRINTR
02870 E1D9 37 OUTCH2 PSH B
02880 E1DA F6 8008 OUTCH1 LDA B ACIAS TEST TRANSMIT DATA
02890 E1DD 57 ASR B REGISTER EMPTY FLAG
02900 E1DE 57 ASR B AND LOOP TILL SET
02910 E1DF 24 F9 BCC OUTCH1
02920 E1E1 B7 8009 STA A ACIAD OUTPUT DATA TO ACIA
02930 E1E4 33 PUL B RESTORE B-REG
02940 E1E5 39 RTS

02960 E1E6 8D D9 JUMP BSR BAD2 GET LOCATION OF JUMP
02970 E1E8 6E 00 JMP X GO TO USER'S PROGRAM

02990 * PRINT CONTENTS OF STACK
03000 E1EA FE A008 PRINT LDX SP
03010 E1ED 08 INX
03020 E1EE 8D 44 BSR OUT2 CONDITION CODES
03030 E1F0 8D 42 BSR OUT2 B ACCUMULATOR
03040 E1F2 8D 40 BSR OUT2 A ACCUMULATOR
03050 E1F4 8D 3C BSR OUTT4 INDEX REGISTER
03060 E1F6 8D 3A BSR OUTT4 PROGRAM COUNTER
03070 E1F8 CE A008 LDX #SP
03080 E1FB 7D A011 TST TFLAG
03090 E1FE 26 21 BNE PRINTS IF IN TRACE MODE
03100 E200 8D 30 BSR OUTT4 STACK POINTER
03110 E202 B6 A014 LDA A BKFLG GET INSTR TO REPLACE BKPNT
03120 E205 27 05 BEQ C2 NO BREAKPOINT SET
03130 E207 FE A068 LDX PB2
03140 E20A A7 00 STA A X REPLACE BREAKPOINT
03150 E20C 7E E0E3 C2 JMP CONTRL

03170 E20F 8D 8C IFILL BSR LIMITS GET START & END ADDRESSES
03180 E211 8D 7F BSR BYT GET DESIRED CONTENTS
03190 E213 FE A002 LDX BEGA 1ST ADDRESS TO INDEX REG
03200 E216 09 DEX
03210 E217 08 FILLOP INX
03220 E218 A7 00 STA A X FILL MEMORY FROM A REG
03230 E21A BC A004 CPX ENDA
03240 E21D 26 F8 BNE FILLOP LOOP UNTIL DONE
03250 E21F 20 EB C5 BRA C2 GO TO CONTROL

03270 E221 E6 00 PRINTS LDA B X WHEN IN TRACE MODE
03280 E223 A6 01 LDA A 1,X DISPLAY S-POINTER THAT
03290 E225 8B 07 ADD A #7 WILL BE USED WHEN EXECUTING
03300 E227 C9 00 ADC B #0 THE DISPLAYED INSTRUCTION
03310 E229 F7 A00E STA B TEMP
03320 E22C B7 A00F STA A TEMP+1
03330 E22F CE A00E LDX #TEMP
03340 E232 20 63 OUTT4 BRA OUT4

```

03350 E234 7E EOCA OUT2 JMP OUT2HS

```

03370          * TRACE ROUTINE
03380 E237 8D 88 TRACE BSR BAD2 GET START ADDRESS OF TRACE
03390 E239 8D 90          BSR CRLF AND SAVE IN XHI & XLOW
03400 E23B FE A008          LDX SP
03410 E23E F6 A00C          LDA B XHI PUT START ADDRESS IN
03420 E241 E7 06          STA B 6,X PROGRAM COUNTER POSITION
03430 E243 B6 A00D          LDA A XLOW IN STACK
03440 E246 A7 07          STA A 7,X
03450 E248 7C A011          INC TFLAG SET TRACE FLAG
03460 E24B 8E A060 RETURN LDS #TSTACK SEPARATE STACK FOR TRACE
03470 E24E 8D 9A          BSR PRINT DISPLAY ALL REGISTERS
03480 E250 7F A065          CLR BFLAG CLEAR BRANCH FLAG
03490 E253 FE A008          LDX SP
03500 E256 EE 06          LDX 6,X GET PROGRAM COUNTER FROM STAC
03510 E258 FF A00C          STX XHI AND SAVE IN XHI AND XLOW
03520 E25B BD E1CB          JSR CRLF
03530 E25E CE A00C          LDX #XHI
03540 E261 8D 34          BSR OUT4 DISPLAY PROGRAM COUNTER
03550 E263 FE A00C          LDX XHI AND FIRST BYTE OF
03560 E266 E6 00          LDA B X INSTRUCTION
03570 E268 8D CA          BSR OUT2
03580 E26A A6 00          LDA A X STORE 2ND BYTE OF INSTRUCTION
03590 E26C B7 A068          STA A PB2 IN PB2 AND 3RD BYTE IN PB3
03600 E26F A6 01          LDA A 1,X IF INSTRUCTION IS LONGER
03610 E271 B7 A069          STA A PB3 THAN ONE BYTE
03620 E274 F7 A067          STA B PB1
03630 E277 C1 8D          CMP B #$8D BSR? TEST FOR SPECIAL CODES
03640 E279 27 12          BEQ BBR
03650 E27B C1 8C          CMP B #$8C CPX?
03660 E27D 27 25          BEQ BYT3
03670 E27F C1 8E          CMP B #$8E LDS?
03680 E281 27 21          BEQ BYT3
03690 E283 C1 CE          CMP B #$CE LDX?
03700 E285 27 1D          BEQ BYT3
03710 E287 C4 F0          AND B #$F0
03720 E289 C1 20          CMP B #$20 TEST FOR RELATIVE BRANCH
03730 E28B 26 0D          BNE NOTB TYPE INSTRUCTIONS
03740 E28D 7C A065 BBR INC BFLAG SET BRANCH FLAG
03750 E290 20 16          BRA BYT2 TWO BYTE INSTRUCTION
03760 E292 7E E055 BYT JMP BYTE
03770 E295 20 88 C4 BRA C5 GO TO CONTROL
03780 E297 7E E0C8 OUT4 JMP OUT4HS
03790 E29A C1 60 NOTB CMP B #$60 IS CODE LESS THAN 60?
03800 E29C 25 0C          BCS BYT1 YES - 1 BYTE INSTRUCTION
03810 E29E C4 30          AND B #$30
03820 E2A0 C1 30          CMP B #$30
03830 E2A2 26 04          BNE BYT2 ONLY 3 BYTE WILL FALL THRU
03840 E2A4 8D F1 BYT3 BSR OUT4 DISPLAY 2 BYTE OPERAND
03850 E2A6 20 02          BRA BYT1
03860 E2A8 8D 8A BYT2 BSR OUT2 DISPLAY 1 BYTE OPERAND
03870 E2AA FF A00C BYT1 STX XHI SAVE LOCATION OF NEXT INSTR
03880          * XHI NOW CONTAINS NEXT INS LOCATION

```

03890	E2AD	7D	A065	TST	BFLAG	IS IT A BRANCH?
03900	E2B0	27	19	BEQ	NOTBB	NO
03910	E2B2	4F		CLR	A	YES, COMPUTE TARGET LOCATION
03920	E2B3	F6	A068	LDA	B PB2	
03930	E2B6	2C	02	BGE	DPOS	TEST FOR BRANCH BACK
03940	E2B8	86	FF	LDA	A #\$FF	FF FOR BACKWARD BRANCH
03950	E2BA	FB	A00D	DPOS	ADD B XLOW	ADD OPERAND TO LOWER
03960	E2BD	B9	A00C	ADC	A XHI	8 BITS OF PROGRAM COUNTER
03970	E2C0	B7	A061	STA	A BPOINT	SAVE TARGET ADDRESS
03980	E2C3	F7	A062	STA	B BPOINT+1	
03990	E2C6	CE	A061	LDX	#BPOINT	DISPLAY TARGET ADDRESS
04000	E2C9	8D	CC	BSR	OUT4	
04010	E2CB	BD	E1CB	NOTBB	JSR	CRLF
04020	E2CE	BD	E1AC	JSR	INEEE	GET COMMAND
04030	E2D1	16		TAB		SAVE IN B REGISTER
04040	E2D2	BD	E0CC	JSR	OUTS	
04050	E2D5	C1	20	CMP	B #\$20	IF SPACE EXECUTE THE
04060	E2D7	27	35	BEQ	DOT	INSTRUCTION. IF NOT A
04070	E2D9	FE	A008	CKCBA	LDX	SP
04080	E2DC	08		INX		SPACE, TEST FOR A CHANGE
04090	E2DD	C1	43	CMP	B #'C	REGISTER COMMAND. NOTE, THIS
04100	E2DF	27	0A	BEQ	RDC	PART OF MEMORY IS SHARED
04110	E2E1	08		INX		WITH THE CHANGE REGISTER
04120	E2E2	C1	42	CMP	B #'B	COMMANDS WHEN NOT IN TRACE
04130	E2E4	27	05	BEQ	RDC	MODE. IF IT IS A CHANGE
04140	E2E6	08		INX		REGISTER COMMAND WHILE IN
04150	E2E7	C1	41	CMP	B #'A	TRACE MODE, RETURN TO
04160	E2E9	26	0A	BNE	CHKX	NOTBB FOR NEXT COMMAND.
04170	E2EB	BD	E0CA	RDC	JSR	OUT2HS
04180	E2EE	09		DEX		DISPLAY REGISTER CONTENTS
04190	E2EF	8D	A1	BSR	BYT	SAVED IN STACK
04200	E2F1	A7	00	STA	A X	GET NEW CONTENTS
04210	E2F3	20	12	BRA	RETDID	AND STORE IN STACK
04220	E2F5	C1	58	CHKX	CMP	B #'X
04230	E2F7	26	9C	BNE	C4	
04240	E2F9	08		INX		
04250	E2FA	8D	9B	BSR	OUT4	DISPLAY INDEX CONTENTS
04260	E2FC	8D	94	BSR	BYT	GET HIGH 8 BITS
04270	E2FE	FE	A008	LDX	SP	
04280	E301	A7	04	STA	A 4,X	STORE IN STACK
04290	E303	8D	8D	BSR	BYT	GET LOWER 8 BITS
04300	E305	A7	05	STA	A 5,X	STORE
04310	E307	7D	A011	RETDID	TST	TFLAG
04320	E30A	26	BF	BNE	NOTBB	IN TRACE?
04330	E30C	20	87	RETNOT	BRA	C4
04340	E30E	C6	3F	DOT	LDA	B #\$3F
04350	E310	B6	A067	LDA	A PB1	SWI CODE TO B-REG
04360	E313	81	8D	CMP	A #\$8D	GET INSTRUCTION
04370	E315	26	0B	BNE	TSTB	IS IT A BSR?
04380	E317	FE	A061	LDX	BPOINT	IF YES, NEXT INSTRUCTION
04390	E31A	FF	A00C	STX	XHI	WILL BE AT ADDRESS STORED
04400	E31D	7F	A065	CLR	BFLAG	IN BPOINT.
04410	E320	20	59	BRA	EXEC	ONLY ONE SWI NEED BE SET
04420	E322	7D	A065	TSTB	TST	SET BKPOINT AND EXECUTE INST
					BFLAG	IS IT CONDITIONAL BRANCH?

04430	E325	27	OC	BEQ	TSTJ	YES, SET BREAKPOINT AT
04440	E327	FE	A061	LDX	BPOINT	TARGET ADDRESS IN CASE
04450	E32A	A6	00	LDA A	X	PROGRAM GOES THERE.
04460	E32C	B7	A063	STA A	BPOINT+2	SAVE INSTRUCTION
04470	E32F	E7	00	STA B	X	SET SWI AT TARGET ADDRESS
04480	E331	20	48	BRA	EXEC	
04490	E333	81	6E	TSTJ	CMP A	INDEXED JUMP INSTRUCTION?
04500	E335	27	14	BEQ	ISX	
04510	E337	81	AD	CMP A	#\$AD	INDEXED JSR?
04520	E339	27	10	BEQ	ISX	
04530	E33B	81	7E	CMP A	#\$7E	STRAIGHT JUMP?
04540	E33D	27	04	BEQ	ISJ	
04550	E33F	81	BD	CMP A	#\$BD	STRAIGHT JSR?
04560	E341	26	1C	BNE	NOTJ	
04570	E343	FE	A068	ISJ	LDX	PUT NEXT INSTRUCTION
04580	E346	FF	A00C	STX	XHI	ADDRESS IN XHI & XLOW
04590	E349	20	30	BRA	EXEC	
04600	E34B	FE	A008	ISX	LDX	COMPUTE NEXT INST ADDRESS
04610	E34E	A6	05	LDA A	5,X	FOR INDEXED JUMPS
04620	E350	BB	A068	ADD A	PB2	
04630	E353	B7	A00D	STA A	XLOW	
04640	E356	A6	04	LDA A	4,X	
04650	E358	89	00	ADC A	#0	
04660	E35A	B7	A00C	STA A	XHI	
04670	E35D	20	1C	BRA	EXEC	
04680	E35F	FE	A008	NOTJ	LDX	
04690	E362	81	39	CMP A	#\$39	IS INSTRUCTION AN RTS?
04700	E364	26	04	BNE	NOTRTS	NO
04710	E366	EE	08	LDX	8,X	YES, PULL RETURN ADDRESS
04720	E368	20	06	BRA	EXR	FROM STACK AND STORE IN
04730	E36A	81	38	NOTRTS	CMP A	NEXT INSTRUCTION POINTER.
04740	E36C	26	05	BNE	NOTRTI	
04750	E36E	EE	0D	LDX	13,X	
04760	E370	FF	A00C	EXR	STX	
04770	E373	81	3F	NOTRTI	CMP A	SWI?
04780	E375	27	95	BEQ	RETNOT	YES, RETURN TO CONTROL
04790	E377	81	3E	CMP A	#\$3E	WAI?
04800	E379	27	91	BEQ	RETNOT	YES, RETURN TO CONTROL
04810	E37B	FE	A00C	EXEC	LDX	SET BREAKPOINT AT NEXT
04820	E37E	A6	00	LDA A	X	INSTRUCTION LOCATION AND SAVE
04830	E380	B7	A066	STA A	OPSAVE	OP CODE.
04840	E383	E7	00	STA B	X	STORE SWI AT BREAKPOINT &
04850	E385	E1	00	CMP B	X	VERIFY THAT IT'S WITHIN RAM
04860	E387	26	83	BNE	RETNOT	IF ROM, GO TO CONTROL
04870						
04880	E389	7E	E199		JMP	RTI TO EXECUTE INSTRUCTION
04900						
04910	E38C	FE	A00C	*RETURN HERE ON SWI IF	SWTURN	TRACE FLAG ON
04920	E38F	B6	A066	LDX	XHI	
04930	E392	A7	00	LDA A	OPSAVE	
04940	E394	7D	A065	STA A	X	REPLACE SWI'S WITH PREVIOUS
04950	E397	27	08	TST	BFLAG	CONTENTS. IF BFLAG IS CLEAR,
04960	E399	FE	A061	BEQ	DISPLY	THEN ONLY ONE BREAKPOINT
				LDX	BPOINT	WAS SET.

```

04970 E39C B6 A063      LDA A  BPOINT+2
04980 E39F A7 00        STA A  X
04990 E3A1 7E E24B DISPLY JMP  RETURN  DISPLAY REGISTER STATUS

05010 E3A4 0D          CRLFAS FCB  $D,$A,0,0,0,4,'S','1,4
      E3A5 0A
      E3A6 00
      E3A7 00
      E3A8 00
      E3A9 04
      E3AA 53
      E3AB 31
      E3AC 04

05030 E3AD 81 55      TDEX  CMP A  #'U      IF IT'S A "U"
05040 E3AF 27 0A      BEQ   CHA71  GET PREVIOUS ADDRESS
05050 E3B1 BD E0AC      JSR   INHEX+2 IF NOT HEX, JMP CONTROL
05060 E3B4 BD E057      JSR   BYTE+2  ELSE, GET NEW DATA
05070 E3B7 09          DEX
05080 E3B8 7E E0A2      JMP   CHA61  STORE NEW DATA
05090 E3BB 09          CHA71 DEX      GET PREVIOUS ADDRESS
05100 E3BC 09          DEX
05110 E3BD FF A00C      STX   XHI
05120 E3C0 7E E087      JMP   CHA51  PRINT PREVIOUS ADDRESS

05140      E3C3      FUTABL EQU  *      COMMAND LOOKUP TABLE
05150 E3C3 4D          FCC   /M/
05160 E3C4 E085      FDB   CHANGE  MEMORY EXAMINE
05170 E3C6 47          FCC   /G/
05180 E3C7 E199      FDB   CONTG   GO TO $A048
05190 E3C9 52          FCC   /R/
05200 E3CA E1EA      FDB   PRINT   PRINT REGISTERS
05210 E3CC 54          FCC   /T/
05220 E3CD E237      FDB   TRACE   TRACE ROUTINE
05230 E3CF 49          FCC   /I/
05240 E3D0 E20F      FDB   IFILL   MEMORY FILL
05250 E3D2 4B          FCC   /K/
05260 E3D3 E189      FDB   BKPNT   SET BREAKPOINT
05270 E3D5 34          FCC   /4/
05280 E3D6 E400      FDB   $E400  GO TO $E400
05290 E3D8 4A          FCC   /J/
05300 E3D9 E1E6      FDB   JUMP    JUMP TO ADDRESS ENTERED
05310 E3DB 51          FCC   /Q/
05320 E3DC 8020      FDB   $8020  QUICKSTART - BOOT DISC
05330 E3DE 44          FCC   /D/
05340 E3DF 7283 >283 FDB   $7283  DISC WARMSTART
05350 E3E1 48          FCC   /H/
05360 E3E2 E1C5      FDB   PRNTON  SET HARDCOPY FLAG
05370 E3E4 4C          FCC   /L/
05380 E3E5 E00A      FDB   LOAD    LOAD ASCII FORMATTED TAPE
05390 E3E7 50          FCC   /P/
05400 E3E8 E127      FDB   PUNCH   PUNCH ASCII FORMATTED TAPE
05410 E3EA 45          FCC   /E/
05420 E3EB E1C4      FDB   ECHON   TURN INPUT ECHO ON

```

05430	E3ED	4E		FCC	/N/	
05440	E3EE	E1C6		FDB	ECHOFF	TURN INPUT ECHO OFF
05450		E3F0	TBLEND	EQU	*	

05470	E3F0	13	MCLOFF	FCB	\$13	
05480	E3F1	0D	MCL	FCB	\$D,\$A,\$14,0,0,'*,4	
	E3F2	0A				
	E3F3	14				
	E3F4	00				
	E3F5	00				
	E3F6	2A				
	E3F7	04				

05500	E3F8	E000		FDB	IO	IRQ VECTOR
05510	E3FA	E113		FDB	SFE	SWI VECTOR
05520	E3FC	E005		FDB	POWDWN	NMI VECTOR
05530	E3FE	E0D0		FDB	START	RESET VECTOR

05550 \* RAM STORAGE LOCATIONS

05570	A000		ORG	\$A000		
05580	A000	0002	IOV	RMB	2	I/O INTERRUPT POINTER
05590	A002	0002	BEGA	RMB	2	BEGINNING ADDRESS
05600	A004	0002	ENDA	RMB	2	ENDING ADDRESS
05610	A006	0002	NIO	RMB	2	NMI INTERRUPT POINTER
05620	A008	0002	SP	RMB	2	TARGET STACK POINTER
05630	A00A	0001	ACIAT	RMB	1	ACIA STATUS WORD
05640	A00B	0001	ECHO	RMB	1	ECHO FLAG
05650	A00C	0001	XHI	RMB	1	INDEX REG HI
05660	A00D	0001	XLOW	RMB	1	INDEX REG LOW
05670	A00E	0001	TEMP	RMB	1	TEMP
05680	A00F	0002	TW	RMB	2	TEMP
05690	A011	0001	TFLAG	RMB	1	TRACE FLAG
05700	A012	0002	XTEMP	RMB	2	X-REG TEMP STORAGE
05710	A014	0001	BKFLG	RMB	1	BREAKPOINT FLAG
05720	A015	002D		RMB	45	SMARTBUG STACK
05730	A042	0001	STACK	RMB	1	STACK POINTER
05740	A043	001D		RMB	29	
05750	A060	0001	TSTACK	RMB	1	TRACE MODE STACK
05760	A061	0003	BPOINT	RMB	3	BRANCH POINT ADDR & CODE
05770	A064	0001	MCONT	RMB	1	TEMP
05780	A065	0001	BFLAG	RMB	1	BRANCH FLAG (TRACE)
05790	A066	0001	OPSAVE	RMB	1	OPERAND (TRACE)
05800	A067	0001	PB1	RMB	1	TRACE TEMP
05810	A068	0001	PB2	RMB	1	TRACE TEMP
05820	A069	0001	PB3	RMB	1	TRACE TEMP
05830	A06A	0001	CKSM	RMB	1	CHECKSUM
05840	A06B	0001	BYTECT	RMB	1	BYTE COUNT
05850		A04A	PRINTR	EQU	\$A04A	USER PRINT ROUTINE

05870  
 ACIAS 8008  
 ACIAD 8009  
 IO E000  
 POWDWN E005  
 LOAD E00A  
 LOAD3 E013  
 LOAD11 E02F  
 LOAD15 E03B  
 LOAD19 E040  
 LOAD21 E044  
 C1 E044  
 BADDR E047  
 BYTE E055  
 OUTHL E067  
 OUTHR E06B  
 OUTCH E075  
 INCH E078  
 PDATA2 E07B  
 PDATA1 E07E  
 CHANGE E085  
 CHA51 E087  
 CHA61 E0A2  
 INHEX E0AA  
 IN1HG E0BE  
 OUT2H E0BF  
 OUT2HA E0C1  
 OUT4HS E0C8  
 OUT2HS E0CA  
 OUTS E0CC  
 START E0D0  
 INZ E0DE  
 INZ1 E0E0  
 CONTRL E0E3  
 NXTCHR E0FF  
 GOODCH E10E  
 SFE E113  
 PUNCH E127  
 PUN11 E134  
 PUN22 E146  
 PUN23 E148  
 PUN32 E167  
 PUNT2 E182  
 PRNT E187  
 BKPNT E189  
 CONTG E199  
 LIMITS E19D  
 OUS E1A9  
 INEEE E1AC  
 BAD2 E1C1  
 ECHON E1C4  
 PRNTON E1C5  
 ECHOFF E1C6  
 C3 E1C9

END

CRLF E1CB  
 PDAT1 E1CE  
 OUTEEE E1D1  
 OUTCH2 E1D9  
 OUTCH1 E1DA  
 JUMP E1E6  
 PRINT E1EA  
 C2 E20C  
 IFILL E20F  
 FILLOP E217  
 C5 E21F  
 PRINTS E221  
 OUTT4 E232  
 OUT2 E234  
 TRACE E237  
 RETURN E24B  
 BBR E28D  
 BYT E292  
 C4 E295  
 OUT4 E297  
 NOTB E29A  
 BYT3 E2A4  
 BYT2 E2A8  
 BYT1 E2AA  
 DPOS E2BA  
 NOTBB E2CB  
 CKCBA E2D9  
 RDC E2EB  
 CHKX E2F5  
 RETDID E307  
 RETNOT E30C  
 DOT E30E  
 TSTB E322  
 TSTJ E333  
 ISJ E343  
 ISX E34B  
 NOTJ E35F  
 NOTRTS E36A  
 EXR E370  
 NOTRTI E373  
 EXEC E37B  
 SWTURN E38C  
 DISPLY E3A1  
 CRLFAS E3A4  
 TDEX E3AD  
 CHA71 E3BB  
 FUTABL E3C3  
 TBLEND E3F0  
 MCLOFF E3F0  
 MCL E3F1  
 IOV A000  
 BEGA A002  
 ENDA A004  
 NIO A006

SP A008  
 ACIAT A00A  
 ECHO A00B  
 XHI A00C  
 XLOW A00D  
 TEMP A00E  
 TW A00F  
 TFLAG A011  
 XTEMP A012  
 BKFLG A014  
 STACK A042  
 TSTACK A060  
 BPOINT A061  
 MCONT A064  
 BFLAG A065  
 OPSAVE A066  
 PB1 A067  
 PB2 A068  
 PB3 A069  
 CKSM A06A  
 BYTECT A06B  
 PRINTR A04A

TOTAL ERRORS 00000