BINARY DUMP/VERIFY PROGRAM (BDP/VER)

INTRODUCTION

The Binary Dump/Verify Program punches a specified area of memory in Binary Tape Format and optionally verifies the punched tape. The tape punched may then be re-loaded into memory using either Autoload, the Binary Loader (BLD) or the LAMBDA Object Loader.

The Binary Dump/Verify Program (BDP/VER) is a "binary relocatable" program and may reside any place in memory. Typically, BDP is loaded into upper memory (at location :0CF0 in 4K, :1CF0 in 8K, etc.) using Autoload or the Binary Loader.

ENVIRONMENT

Hardware Required:

1. ALPHA LSI or ALPHA 16 series computer
2. ASR 33/35 Teletype or High Speed Paper Tape Reader

Software Required:

1. Binary Loader (BLD) or Autoload

PROGRAM DESCRIPTION

The Binary Dump/Verify program will dump the contents of memory specified in a binary tape format reloadable with Autoload, the Binary Loader or LAMBDA Object Loader.

BDP is normally used to dump a series of previously loaded object programs as a single binary program. In this mode BDP allows the suppression or inclusion of transfer addresses and end-of-file (EOF) records, as required.

Binary Tape Format

The binary tape produced by BDP contains loader type codes and data or instructions grouped into records. Each record begins with a record marker (:FF), a record length (in bytes), an ORG (ABS or REL), and is terminated with a checksum (figure 1).

The tape is terminated with an end-of-file (EOF) record, consisting of a record marker and a record length of zero.

Binary Dump Type Codes

The binary dump program generates the loader type codes described below:

EOF (code = :00).	EOF is denoted by a "null" record; i.e., a record marker (:FF) with a record count of zero.
Begin Program (code = :01).	The type code is followed by two bytes of zeros.
END absolute (code = :02).	The type code is followed by two bytes containing the start address. A negative value indicates no start address.
END relocatable (code = :03).	The type code is followed by two bytes containing the start address. A negative value indicates no start address.
ORG absolute (code = :04).	The type code is followed by two bytes of absolute load address.
ORG relocatable (code = :05).	The type code is followed by two bytes of relative load address.
DATA absolute (code = :06).	The type code is followed by a two byte count of the number of data words to load. This is followed by the data, packed two bytes per word.

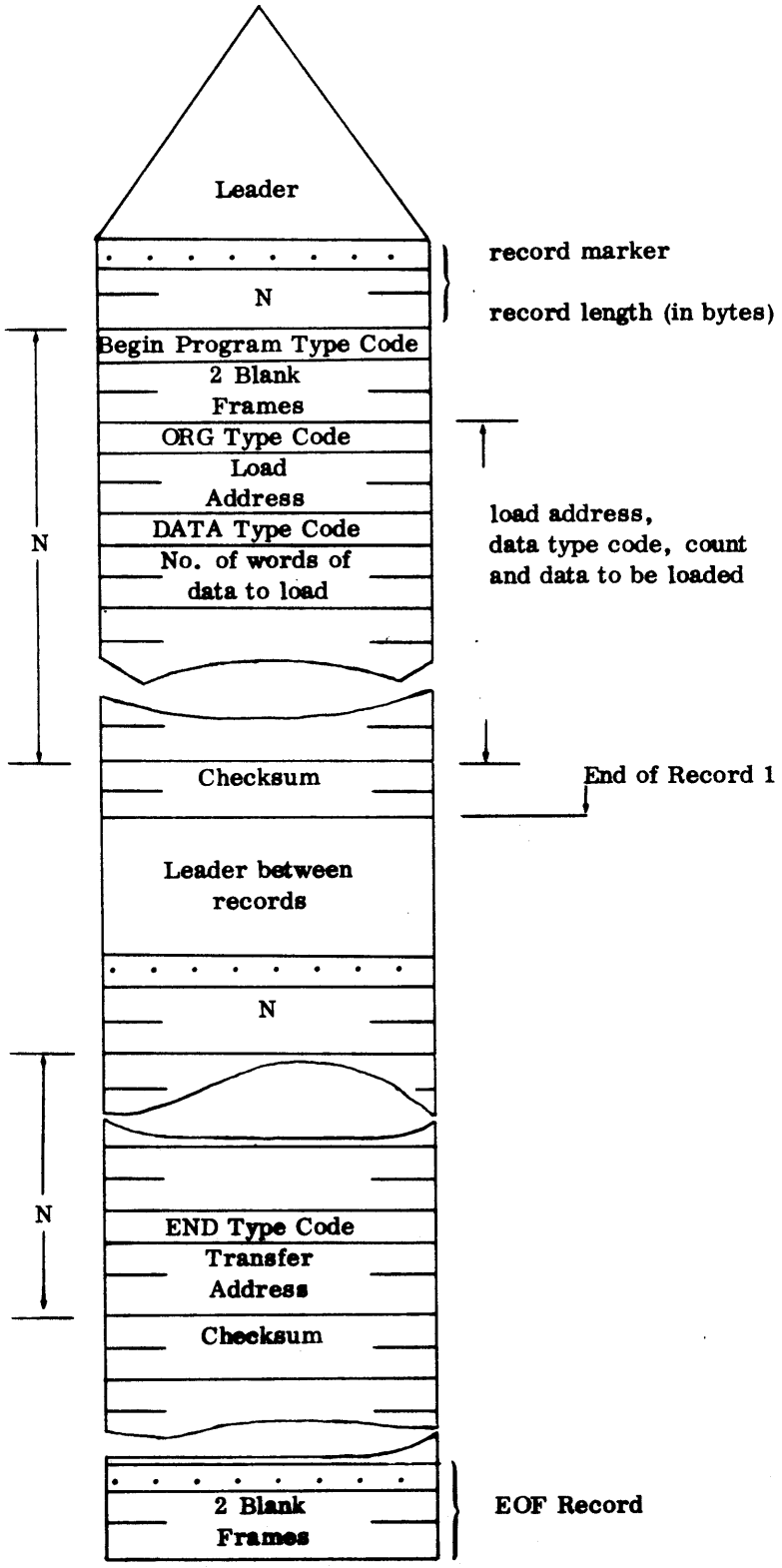


Figure 1. Binary Tape Format

Checksum

A rotating, Exclusive OR byte checksum is output for all bytes of the record except for the record marker and the checksum itself. The following is the checksum sequence:

```
LDA     BYTE        character to checksum
EMA     CKSUM       checksum word
RRA     1
JOR     $+2
ADD     H80         add hexadecimal 80
XOR     CKSUM
AND     HFF         keep low 8 bits
EMA     CKSUM       new checksum word
```

PROGRAM OPERATION

Program Loading

BDP/VER is loaded into available memory using any of the standard loaders, typically at location :nCF0.

Operation

Prior to execution of BDP/VER, the following initialization must be performed:

1. The A register must be preset to the initial memory location to be punched or verified.

2. The X register must be preset to the last memory location to be punched or verified.

3. The Console Sense Register (low order 4 Data Switches on ALPHA 16) must be preset to indicate device and mode of operation. The possible values are:

Mode \ Device		Include EOF		Suppress EOF	
		TTY	HSPT	TTY	HSPT
Punch	Abs	:0	:1	:2	:3
	Rel	:8	:9	:A	:B
Verify	Abs	:4	:5	:6	:7
	Rel	:C	:D	:E	:F

Figure 2. BDP/VER Mode and Device Selection

4. The SENSE switch should be set ON if a transfer address is desired, otherwise set off.

BDP/VER is then entered at its initial load location (nominally :nCF0). If a transfer address is desired, enter it into the A register when the processor halts (I = :0802) and press RUN.

End Action

After punching a binary tape, BDP/VER will output an end-of-file and halt (I = :0800). The tape should then be verified, after which BDP/VER will again halt (I = :0800).

End-of-File (EOF) Suppression

The end-of-file record normally produced at the end of a dump operation can be suppressed (see Figure 2). This allows the user to dump a program as multiple program segments, without intervening end-of-file records. If an execution address is to be punched, it must be entered while dumping the last segment, and a terminating end-of-file record should always be punched.

An example of this requirement is when an Autoload format tape must be produced in segments, such as around locations :30 through :3B which are required by Autoload itself.

Error Handling/Recovery

The Binary Dump/Verify program will terminate operation with a halt code displayed in the I register when an error is detected.

Entry of an initial location (A register) which is greater than the terminal location (X register) entered will cause a halt with I = :0801. BDP/VER should be restarted from the top, with corrected values in A and X.

Detection of a verify error will cause a halt (I = :0803) with the word that miscompared (2 frames) located immediately under the read station. This error indicates that the tape was either punched incorrectly or has been altered in memory since the tape was punched. Examination of memory will verify the word in question.

I Register	Interpretation
:0800	Punch or Verify operation completed
:0801	Initial location > last location; error
:0802	Enter transfer address and press RUN
:0803	Verify error occurred

Figure 3. BDP/VER Halt Summary

Appendix A

BDP/VER OPERATION SUMMARY

The procedure for dumping and verifying binary programs in an ALPHA series processor is as follows:

1. Load BDP/VER into high memory using the Binary Loader (BLD) or Autoload; standard load location is :nCF0.

2. Enter into the A register the initial (first) memory location to be punched or verified.

3. Enter into the X register the last memory location to be punched or verified.

4. Enter into the P register the starting (first load) location of BDP/VER. 1CF0

5. Enter into the Console Sense Register (low order 4 Data Switches on ALPHA 16) the value corresponding to the mode of operation and device desired (see table A1).

Mode \ Device		Include EOF		Suppress EOF	
		TTY	HSPT	TTY	HSPT
Punch	Abs	:0	:1	:2	:3
	Rel	:8	:9	:A	:B
Verify	Abs	:4	:5	:6	:7
	Rel	:C	:D	:E	:F

Table A1. BDP/VER Mode and Device Selection

6. Set the SENSE switch ON if a transfer address is desired, otherwise off.

7. Verify the selected device is loaded and ready.

8. Press RUN to begin execution of BDP/VER.

9. If the transfer address option was selected (SENSE switch ON) the processor will immediately halt with I equal to :0802. Enter the execution address desired into the A register and press the RUN switch.

10. The binary tape will be punched on the selected device and the processor will halt with I equal to :0800. To verify the tape punched, repeat the procedure from step 2.

```
PAGE  0001  10/03/73  17:30:11  BINARY DUMP/VERIFY  96019-1000

0002                  * ALPHA 16/LSI BINARY DUMP/VERIFY (BDP/VER)
0003                  * UTILITY  96019
0004                  * COPYRIGHT 1972 BY COMPUTER AUTOMATION, INC.
0005                  * OCTOBER 24, 1972
0006                  * THIS PROGRAM IS BINARY RELOCATABLE
0007                  *
0008  0CF0                    REL   :CF0      4K LOAD LOCATION
0009  0CF0  9AAF   BDP        STA   SPNCH     START PUNCH LOCATION
0010  0CF1  0130              IXA
0011  0CF2  92AD              SUB   SPNCH
0012  0CF3  3181              JAG   $+2
0013  0CF4  0601              STOP  1         SLOC>ENDLOC.=ERROR
0014  0CF5  9AAD              STA   WDS       NO. OF DATA WORDS TO PUNCH
0015  0CF6  9AAD              STA   BFLG      BEGIN FLAG (BFLG>0=BEGIN)
0016  0CF7  5A01              ISX             DATA SWITCHES TO X
0017  0CF8  13A8              LRX   1
0018  0CF9  2201              JOS   $+2
0019  0CFA  0110              ZAR
0020  0CFB  9AA0              STA   PFLG      NON-ZERO=HSP -- ZERO=TTYP
0021  0CFC  0010              ARM
0022  0CFD  13A8              LRX   1
0023  0CFE  2201              JOS   $+2
0024  0CFF  0110              ZAR
0025  0D00  9AA8              STA   EFLG      SET EOT SUPPRESSION
0026  0D01  13A8              LRX   1
0027  0D02  EA9E              STX   RFLG      SET RELOCATION FLAG
0028  0D03  0018              AXM
0029  0D04  2201              JOS   $+2
0030  0D05  0108              ZXR
0031  0D06  EA9F              STX   VFLG      SET VERIFY FLG
0032  0D07  2401              JSR   $+2       SS SET= EXECUTE ADDR
0033  0D08  0802              STOP  2         0P END ADDR
0034  0D09  9A98              STA   EXAD      SET EXECUTE ADDRESS
0035  0D0A  C450              LXP   80        PUNCH 8 INCHES
0036  0D0B  FA79              JST   LDR
0037  0D0C  C6FF   BDP1       LAP   :FF       OUTPUT RECORD MARKER
0038  0D0D  E298              LDX   VFLG      VERIFY MODE?
0039  0D0E  3801              JXN   $+2       YES--DON'T OUTPUT RCRD MRKR
0040  0D0F  FA5B              JST   PCH
0041  0D10  C750              LAM   :50       MAX DATA WORDS IN A RECORD
0042  0D11  8A91              ADD   WDS       SUB. FROM WORDS TO PUNCH
0043  0D12  3084              JAP   BDP2
0044  0D13  0110              ZAR             NOT FULL RECORD SIZE
0045  0D14  BA8E              EMA   WDS       RESET NO. OF WORDS
0046  0D15  0048              TAX
0047  0D16  F202              JMP   BDP3
0048  0D17  9A8B   BDP2       STA   WDS       SAVE EXCESS
0049  0D18  C450              LXP   :50
0050  0D19  1328   BDP3       LLX   1         CONVERT TO BYTE COUNT
0051  0D1A  EA8A              STX   DATSAV    NO. OF DATA BYTES TO OUTPUT
0052  0D1B  B288              LDA   BFLG      .FIRST RECORD?
0053  0D1C  2101              JAZ   $+2       NO
```

```
0054  0D1D  C203          AXI  :3        YES,COMPENSATE FOR BEGIN
0055  0D1E  B284          LDA  WDS       LAST RECORD?
0056  0D1F  3101          JAN  $+2       NO
0057  0D20  C203          AXI  :3        YES,COMPENSATE FOR END
0058  0D21  C206          AXI  :6        COMPENSATE FOR ORG/DATA
0059  0D22  0110          ZAR
0060  0D23  9A79          STA  CKSUM     INITIALIZE CHECK SUM
0061  0D24  FA1D          JST  CKOUT     OUTPUT WORD COUNT
0062  0D25  B27E          LDA  BFLG      FIRST RECORD?
0063  0D26  2106          JAZ  BDP4      NO
0064  0D27  C601          LAP  1         YES, OUTPUT BEGIN TYPE CODE
0065  0D28  FA69          JST  CKSM
0066  0D29  FA41          JST  PCH
0067  0D2A  0108          ZXR
0068  0D2B  EA78          STX  BFLG      RESET FIRST RECORD FLAG
0069  0D2C  FA15          JST  CKOUT     OUTPUT 2 NULL FRAMES
0070  0D2D  C604  BDP4    LAP  :4        OUTPUT ORG  4=ABS  5=REL
0071  0D2E  E272          LDX  RFLG      ZERO=ABS    NON-ZERO=REL
0072  0D2F  2801          JXZ  $+2
0073  0D30  0150          IAR            MAKE REL
0074  0D31  FA60          JST  CKSM
0075  0D32  FA38          JST  PCH       OUTPUT ORG TYPE CODE
0076  0D33  E26C          LDX  SPNCH     PICK UP START ADDRESS
0077  0D34  FA0D          JST  CKOUT     OUTPUT START ADDRESS
0078  0D35  C606          LAP  :6        DATA TYPE CODE
0079  0D36  FA5B          JST  CKSM
0080  0D37  FA33          JST  PCH       PUT DATA TYPE CODE OUT
0081  0D38  E26C          LDX  DATSAV    NO. OF DATA BYTES TO OUTPUT
0082  0D39  13A8          LRX  1         CONVERT TO WORD COUNT
0083  0D3A  FA07          JST  CKOUT     OUTPUT DATA COUNT
0084  0D3B  B269          LDA  DATSAV    PICK UP DATA BYTE COUNT
0085  0D3C  0310          NAR
0086  0D3D  9A5D          STA  DCNT      SAVE NEGATIVE
0087  0D3E  E361  BDP5    LDX  *SPNCH    PICK UP DATA WORD
0088  0D3F  DA60          IMS  SPNCH     BUMP ADDRESS POINTER
0089  0D40  FA01          JST  CKOUT     OUTPUT DATA WORD
0090  0D41  F603          JMP  BDP5      LOOP
0091  0D42  0800  CKOUT   ENT
0092  0D43  0110          ZAR
0093  0D44  1907          LRL  8         HIGH ORDER BITS TO A REG
0094  0D45  FA4C          JST  CKSM
0095  0D46  FA24          JST  PCH       OUTPUT FIRST BYTE
0096  0D47  DA53          IMS  DCNT      BUMP DATA BYTE COUNT
0097  0D48  0110          ZAR
0098  0D49  1907          LRL  8         LOW ORDER BITS TO A REG
0099  0D4A  FA47          JST  CKSM
0100  0D4B  FA1F          JST  PCH       OUTPUT SECOND BYTE
0101  0D4C  DA4F          IMS  DCNT      BUMP DATA BYTE COUNT
0102  0D4D  F70B          RTN  CKOUT
0103  0D4E  B254          LDA  WDS       LAST RECORD?
0104  0D4F  3108          JAN  OUTCK1    NO
0105  0D50  C602          LAP  :2        DONE -- OUTPUT END TYPE CODE
```

```
0106  0D51  E24F          LDX  RFLG      PICK UP REL FLAG
0107  0D52  2801          JXZ  $+2       ZERO=ABS END
0108  0D53  0150          IAR
0109  0D54  FA3D          JST  CKSM
0110  0D55  FA15          JST  PCH       OUTPUT END CODE
0111  0D56  E24B          LDX  EXAD      PICK UP EXECUTION ADDRESS
0112  0D57  FE15          JST  CKOUT     OUTPUT EXECUTION ADDRESS
0113  0D58  E244  OUTCK1  LDX  CKSUM
0114  0D59  FE17          JST  CKOUT     OUTPUT CHECK SUM
0115  0D5A  B24E          LDA  EFLG      SEE IF EOT SUPPR.
0116  0D5B  AA47          XOR  WDS       MATCH WITH WORDS
0117  0D5C  0150          IAR
0118  0D5D  210C          JAZ  HALT      IF LAST RCRD OUT
0119  0D5E  C40A          LXP  10
0120  0D5F  FA25          JST  LDR
0121  0D60  B242          LDA  WDS
0122  0D61  2101          JAZ  $+2
0123  0D62  F056          JMP  BDP1      NOT LAST RECORD -- CONTINUE
0124  0D63  E242          LDX  VFLG      WAS THIS A VERIFY RUN?
0125  0D64  2801          JXZ  $+2       NO--OUTPUT EOT
0126  0D65  F204          JMP  HALT
0127  0D66  C0FF          LAP  :FF       OUTPUT END OF TAPE (EOT)
0128  0D67  FA03          JST  PCH
0129  0D68  C45A          LXP  90
0130  0D69  FA1B          JST  LDR
0131  0D6A  1800  HALT    STOP 0         END OF PCH AND GOOD VERIFY
0132  0D6B  3800  PCH     ENT
0133  0D6C  9A3A          STA  TEMP      SAVE CHAR TO PUNCH/VERIFY
0134  0D6D  EA24          STX  CKSM
0135  0D6E  E22D          LDX  PFLG      WHICH DEVICE?
0136  0D6F  3807          JXN  PCH2
0137  0D70  E235          LDX  VFLG      VERIFY?
0138  0D71  2803          JXZ  PCH1      NO--GO PUNCH CHAR
0139  0D72  403A          SEL  072       STEP READER
0140  0D73  7939          RBA  071       INPUT BYTE
0141  0D74  F209          JMP  COMP      GO DO COMPARE
0142  0D75  6D39  PCH1    WRA  071       TELETYPE
0143  0D76  F20C          JMP  PCH4
0144  0D77  E22F  PCH2    LDX  VFLG      VERIFY?
0145  0D78  2803          JXZ  PCH3      NO--GO PUNCH CHAR
0146  0D79  4032          SEL  062       STEP READER
0147  0D7A  7931          RBA  061       INPUT BYTE
0148  0D7B  F202          JMP  COMP      GO DO COMPARE
0149  0D7C  6D31  PCH3    WRA  061       PUNCH CHAR
0150  0D7D  F205          JMP  PCH4
0151  0D7E  E229  COMP    LDX  LFLG      LOOKING FOR LEADER?
0152  0D7F  3803          JXN  PCH4      YES--DON'T DO COMPARE
0153  0D80  9226          SUB  TEMP
0154  0D81  2101          JAZ  $+2
0155  0D82  0803          STOP 3         VERIFY ERROR
0156  0D83  E20E  PCH4    LDX  CKSM      RECOVER XREG
0157  0D84  F719          RTN  PCH
```

```
0158  0D85  0800  LDR     ENT
0159  0D86  B21F  LDR1    LDA   VFLG      VERIFY MODE?
0160  0D87  2102          JAZ   $+3       NO--PUNCH LEADER
0161  0D88  EA1F          STX   LFLG      SET  LEADER FLAG
0162  0D89  C201          AXI   :1
0163  0D8A  C110          ZAR
0164  0D8B  FE20          JST   PCH
0165  0D8C  C0FF          CAI   :FF       LOOK FOR RECORD MARKER
0166  0D8D  0528          XRP             FOUND RECORD MARKER
0167  0D8E  0048          DXR
0168  0D8F  3849          JXN   LDR1      PUNCH NO. OF BLANKS IN XREG
0169  0D90  EA17          STX   LFLG      RESET LEADER FLAG
0170  0D91  F70C          RTN   LDP
0171  0D92  0800  CKSM    ENT
0172  0D93  BA09          EMA   CKSUM
0173  0D94  11D0          RRA   1
0174  0D95  3201          JGR   $+2
0175  0D96  8A07          ADD   H80
0176  0D97  AAC5          XOR   CKSUM
0177  0D98  8206          AND   MFF
0178  0D99  FA03          EMA   CKSUM
0179  0D9A  F708          RTN   CKSM
0180  0D9B  0000  DCNT    DATA  0         NO. OF DATA BYTES THIS RCRD
0181  0D9C  0000  PFLG    DATA  0         DEVICE INDICATOR
0182  0D9D  0000  CKSUM   DATA  0         CHECK SUM
0183  0D9E  0080  H80     DATA  :80       BIT 7 (CHECK SUM)
0184  0D9F  00FF  MFF     DATA  :FF       CHECK SUM WORD MASK
0185  0DA0  0000  SPNCH   DATA  0         START DUMP ADDRESS
0186  0DA1  0000  RFLG    DATA  0         RELOCATE FLAG
0187  0DA2  0000  EXAD    DATA  0         TRANSFER(EXECUTE) ADDRESS
0188  0DA3  0000  WDS     DATA  0         NO. OF WORDS(BYTES) TO PUNCH
0189  0DA4  0000  BFLG    DATA  0         BEGIN(FIRST RECORD) FLAG
0190  0DA5  0000  DATSAV  DATA  0         TEMP STORE OF DATA COUNT
0191  0DA6  0000  VFLG    DATA  0         VERIFY FLAG
0192  0DA7  0000  TEMP    DATA  0         TEMP STORE IN VERIFY MODE
0193  0DA8  0000  LFLG    DATA  0         READ LEADER FLAG
0194  0DA9  0000  EFLG    DATA  0
0195                      END
0000 ERRORS
```

```
0009  BDP
0037  BDP1      0123
0048  BDP2      0043
0050  BDP3      0047
0070  BDP4      0063
0087  BDP5      0090
0189  BFLG      0015* 0052  0062  0068*
0091  CKOUT     0061* 0069* 0077* 0083* 0089* 0102  0112* 0114*
0171  CKSM      0065* 0074* 0079* 0094* 0099* 0109* 0134* 0156
                0179
0182  CKSUM     0060* 0113  0172* 0176  0178*
0151  COMP      0141  0148
0190  DATSAV    0051* 0081  0084
0180  DCNT      0086* 0096* 0101*
0194  EFLG      0025* 0115
0187  EXAD      0034* 0111
0183  H80       0175
0131  HALT      0118  0126
0158  LDR       0036* 0120* 0130* 0170
0159  LDR1      0168
0193  LFLG      0151  0161* 0169*
0184  NFF       0177
0113  OUTCK1    0104
0132  PCH       0040* 0066* 0075* 0080* 0095* 0100* 0110* 0128*
                0157  0164*
0142  PCH1      0138
0144  PCH2      0136
0149  PCH3      0145
0156  PCH4      0143  0150  0152
0181  PFLG      0020* 0135
0186  RFLG      0027* 0071  0106
0185  SPNCH     0009* 0011  0076  0087  0088*
0192  TEMP      0133* 0153
0191  VFLG      0031* 0038  0124  0137  0144  0159
0188  WDS       0014* 0042  0045* 0048* 0055  0103  0116  0121
0195 SOURCE LINES
```

CONVERSATIONAL DEBUG (DBG)

INTRODUCTION

DEBUG (DBG) is an on-line conversational utility program which aids the user in debugging his programs on an ALPHA series computer with the aid of the ASR-33 teletype.

The user loads his programs into memory and enters the DBG program at its load address. Debugging commands are entered on the keyboard and allow the user to search, modify, and print memory; and execute portions of his program to selected breakpoints. Other functions allow the user to set or display the contents of the A, X, and O (program status) registers, initialize or copy regions of memory, transfer control to the user program, set relocation biases for subroutines, and provide a Console Interrupt (TRAP) escape from the user program to DBG.

DBG must be in the "word operand" mode of operation. The byte/word mode indicator is unconditionally set to word mode upon entry to DBG. When DBG processes a breakpoint, the status of the user's byte/word mode indicator is saved. DBG operates in word mode, and the user's status is restored prior to returning to the user's program via a jump (J) or breakpoint (B) command.

ENVIRONMENT

DBG is a binary relocatable program and may be loaded anywhere in memory using the Binary Loader (BLD) or Autoload.

Hardware Required:

1. ALPHA LSI or ALPHA 16 processor with 4K read/write memory and Teletype option.
2. ASR 33/35 Teletype.

Software Required:

1. Binary Loader (BLD) or Autoload

Additional Hardware Supported:

1. CRT in place of Teletype.
2. Line printer.

PROGRAM DESCRIPTION

Function Command Lines

Upon entry and at the completion of each command DBG will output a carriage return-line feed (CRLF), and a greater than sign (>), to notify the operator that it is ready to accept a DBG command. The first character of the command is the function identification character which is unique for each function. This character is followed by one or more address or data parameters which are separated by a single space and terminated by a period.

If an error in entering a command is made, the user may enter a left arrow to abort the command. If an illegal command is entered, DBG will ignore the command, output a left arrow (←), CRLF, and greater than sign (>), and wait for a legal request.

Address and Data Parameters

All address and data parameters are entered and displayed as hexadecimal values (1A7E). Leading zeros need not be entered. Negative values are entered by preceding them with a minus (-) sign, (-A7E). If a relocation bias is to be added to the parameter, it is suffixed with the appropriate relocation register ID (A7ER5). The relocation registers may be preset to the appropriate values. If an error has been made in entering a parameter, it may be reset to zero and reentered by entering a slash (/), followed by the correct value (A7E/A7F). If an illegal (non-hexadecimal), character is entered, the command will be aborted and a left arrow output.

Entering Parameters

Parameters may be specified to DBG in one of two ways. They may be entered as a single value followed by a valid delimiter (13C), or as the sum or difference of two values followed by a valid delimiter (13C+2B). Each parameter is limited to the sum or difference of two values. With the exception of the pseudo registers A, X, and O, a function ID followed by a period is assumed to be an implied zero parameter. Addresses may be referenced indirectly by setting Bit 15 of the specified parameter equal to one, (Parameter + : 8000).

NOTE

If the user attempts to alter nonexistent memory, DEBUG will "hang up" since the effective address cannot be resolved.

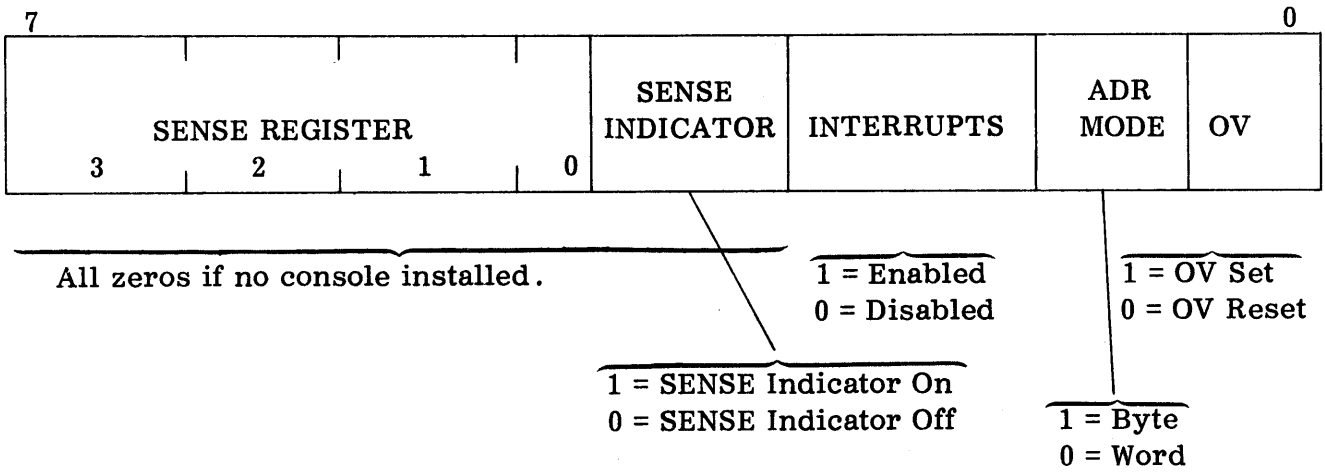
O (Status Word) Register

The O register provides the user with a means of setting or inspecting the status of the program he is debugging.

The user may preset the program status prior to an unconditional transfer to the program being debugged (J function).

The breakpoint function allows the user to inspect the status of the program being debugged by printing the contents of the O register as each breakpoint is reached.

The readout of the low-order 8 bits of the O register is:



PROGRAM OPERATION

Program Loading

DBG may be loaded by any of the standard ALPHA loaders (BLD or LAMBDA) or Autoload into whatever area of memory is available. DBG's standard load location is :0DB0 for 4K, :1DB0 for 8K, etc.

Initialization

DBG is entered at its initial location (:nDB0) and will immediately output its name and a greater than sign (>) indicating its readiness to accept a DBG command.

Command Descriptions

In the following command descriptions capital letters represent function identification characters, small letters represent hexadecimal parameters, the @ symbol represents a single space, CRLF represents a carriage return line feed and all DBG output is underlined. Letters enclosed in parenthesis denote the contents of the specified location; (aaaa)= contents of location aaaa.

Set Display/Change Relocation Registers

External subroutines are normally assembled at location :0. When loaded by the Object Loader they are offset, creating a relocation bias which must be added to the location values on the subroutine listing in order to determine the true location in memory of the subroutine. In order to alleviate this problem sixteen relocation registers are provided (designated R0 through RF), which may be set to the relocation bias of up to sixteen subroutines. When parameters are suffixed by a relocation register ID (R0 through RF), the designated bias is added to the value of the parameter. Hence, in order to refer to location 12C of a subroutine loaded at CD8 the user may set R1 to CD8 (the relocation bias), and enter 12CR1 as the parameter. The R command allows the user to set, display and/or change the values of the relocation registers.

>Rnaaaa.CRLF
>
Set relocation register n to the value aaaa (n = 0 through F).

>Rn.@vvvv@.CRLF
>
Display the value (vvvv) of relocation register n.

>Rn.@vvvv@aaaa.CRLF
>
Display the value (vvvv) of relocation register n and then change it to the value aaaa.

Fill Memory

The F function fills a given area of memory with a specified constant enabling the user to initialize tables and buffer areas.

>Faaaa@bbbb@vvvv.CRLF
>
Store the value vvvv in memory locations aaaa through bbbb inclusive.

Modify Memory

The M function allows the user to enter corrections into memory consecutively starting at a given address.

```
>Maaaa.vvvv.CRLF
>
```

Store the value vvvv at location aaaa.

```
>Maaaa.vvvv@wwww@....yyyy@zzzz.CRLF
>
```

Store the values vvvv through zzzz in memory consecutively, starting at location aaaa.

List Memory

The L function allows the user to verify the contents of tables, buffers, or programs by listing their area of memory on the Centronics line printer. Up to eight locations together with the address of the first location are printed on each line. Up to 60 lines (480 locations) are printed on each page.

```
>Laaaa@bbbb.CRLF
aaaa@(aaaa)@(aaaa+1).....(aaaa+7)CRLF
aaaa+8@(aaaa+8)@(aaaa+9)....(bbbb-1)@(bbbb)CRLF
>
```

List the contents of memory locations aaaa through bbbb inclusive. If (aaaa) contain an indirect pointer, the contents of the effective address are printed.

NOTES

The list operation can be terminated by holding down the BREAK key until DBG responds with the query (>) indicator.

The listing may be taken on the Data Products line printer (model 2310 or equivalent) by replacing the instruction at label NOP:ME (see listing) with a NOP (:0000) instruction.

Print Memory

The P function allows the user to verify the contents of tables, buffers, or programs by printing their area of memory on the Teletype printer. Up to eight locations together with the address of the first location are printed on each line.

```
>Paaaa@bbbb.CRLF
aaaa@(aaaa)@(aaaa+1)....(aaaa+7)CRLF
aaaa+8@(aaaa+8)@(aaaa+9)....(bbbb-1)@(bbbb)CRLF
>
```

Print the contents of memory locations aaaa through bbbb inclusive. If (aaaa) contains an indirect pointer the contents of the effective address are printed.

NOTE

The print operation can be terminated by holding down the BREAK key until DBG responds with the query (>) indicator.

Inspect/Change Memory

The I function allows the user to display individual memory locations and change their values if desired. The user enters the address of the first word to be inspected and DBG will return its address and contents. If the value is to be changed, a new value is entered. If the next consecutive location is to be inspected, a space is entered. If the preceding location is to be inspected a comma (,) is entered. DBG will terminate the Inspect Function if the user attempts to decrement past location : 0. When all corrections have been made, a period is entered to terminate inspection.

>Iaaaa.CRLF	Inspect location aaaa.
aaaa@(aaaa)@@CRLF	
aaaa+1@(aaaa+1)@vvvv,CRLF	Change the contents of location aaaa+1 to vvvv and go back 1 location.
aaaa@(aaaa)@,CRLF	Do not change. Continue back.
aaaa-1@(aaaa-1)@xxxx.CRLF	Change the contents of location aaaa-1 to xxxx and terminate inspection.
>	

If (aaaa) contains an indirect pointer the contents of the effective address are printed.

Search Memory

The S function enables the user to search a given area of memory for a specified key value. The location and contents of each key value found in the given area are listed.

```
>Saaaa@bbbb@vvvv.CRLF
cccc@vvvvCRLF
dddd@(dddd)CRLF
>
```

Search memory from location aaaa through location bbbb inclusive for value vvvv and list the location(s) where it is found. The search test is satisfied if (aaaa+n) = vvvv. In this example the value vvvv was found at locations cccc and dddd.

```
>Saaaa@bbbb@vvvv@mmmm.CRLF
cccc@(cccc)CRLF
dddd@(dddd)CRLF
eeee@(eeee)CRLF
>
```

Search memory from location aaaa through location bbbb for the value vvvv and compare only those bits which have a corresponding one bit in the mask word mmmm. The search test is satisfied if (aaaa+n) XOR mmmm= vvvv XOR mmmm. In this example three locations cccc, dddd, and eeee were found.

Copy Memory

The C function copies one area of memory to another, allowing the user to move tables or relocatable programs to a new area.

```
>Caaaa@bbbb@cccc.CRLF
>
```

Copy locations aaaa through bbbb inclusive to location cccc and following. aaaa≤cccc≤bbbb is illegal!

Set Display/Change Pseudo Registers

In debugging the users program it is often necessary to preset the hardware registers with parameters or initial values in order to test specific functions. Also when reaching a breakpoint it is necessary to save the values of the hardware registers to verify the accuracy of the tested program module. Three pseudo registers, A, X, and O, corresponding to the Accumulator, Index, and Status Word registers are provided for this purpose. An exit from DBG causes the hardware registers to be loaded from the pseudo registers, while a return to DBG from the users program causes the hardware registers to be stored in the pseudo registers.

```
>Aaaaa.CRLF
>Xxxxx.CRLF
>O0.CRLF
>
```

Set the pseudo Accumulator to the value aaaa. Set the pseudo Index register to the value xxxx. Set the pseudo Status Word to the value 0.

```
>A.@aaaa@.CRLF
>X.@xxxx@.CRLF
>O.@oooo@.CRLF
>
```

Display the contents of the pseudo registers A, X and O.

>A.@aaaa@vvvv.CRLF
>
Display the value of the pseudo register A (X,O) and then change it to the value vvvv.

Set Breakpoint and Transfer Control

The B function allows the user to establish one or two breakpoints, then transfer control to his program. When his program reaches either breakpoint it will return control to the DBG program. DBG will save the contents of the A, X, and O registers, restore both breakpoints, print the address of the breakpoint it reached and the contents of the A, X, and O registers as of the last executed instruction before the breakpoint. As larger and larger program modules are tested and corrected, breakpoints may be set at longer intervals until the entire program is debugged.

NOTE

DBG uses one location in the users program for each breakpoint set and location :FF in page zero as return linkage for breakpoint functions.

>Bvvvv@cccc,dddd.CRLF
cccc@aaaa@xxxx@ooooCRLF
>
Set a breakpoint at cccc and dddd. Save the user instructions and store a jump and store indirect to DBG at these locations. Set the hardware registers from the corresponding pseudo registers and transfer control to the users program at location vvvv. When either breakpoint is reached, save the hardware registers in the pseudo registers, restore both breakpoints, print the addresss of the breakpoint reached and values of the pseudo registers A, X, and O. In the case illustrated above the breakpoint at location cccc was reached.

>Bbbbb,cccc.CRLF
bbbb@aaaa@xxxx@ooooCRLF
>
Set two new breakpoints. Transfer control to the user program at the location of the last breakpoint reached. When either breakpoint is reached, respond as above.

>Bbbbb,cccc.CRLF
cccc@aaaa@ooooCRLF
>

Set one breakpoint at location cccc. Transfer control to the users program at location bbbb. When the breakpoint is reached, respond as above.

CAUTION

If the users program does not reach the breakpoint, DBG should be reentered in order to restore the instruction at the breakpoint. DBG will respond as if a breakpoint had been reached. Unless a Trap escape from the users program had been used, the information printed out will have no meaning.

NOTE

If an error is made terminating the third parameter of a multiple breakpoint function, DBG will respond as if a breakpoint had been reached to clear the first breakpoint set.

Transfer Control

The J function allows the user to preset the hardware registers from the pseudo registers and transfer control to his program unconditionally. This function is employed when a great deal of faith in the object program has been obtained.

>Jaaaa.

Set the hardware registers from the pseudo registers and jump to location aaaa.

Trap Function

The T function allows the user to preset a Console Interrupt escape from his program prior to conditionally (B function) or unconditionally transferring control (J function) to his program from DBG.

If, for some reason, the program begins to 'run away', or does not respond correctly, the Console Interrupt Switch may be depressed, providing a Breakpoint Return to DBG. DBG will print the contents of the A, X, and O registers at the time the Console Interrupt occurred and the location of the next instruction to be executed had the interrupt not occurred. DBG will retain control for further debugging operations. TRAP will remain set until DBG is reentered at its intial entry location.

One of the following three conditions must exist for the Console Interrupt Escape to operate successfully:

1. The users program must enable interrupts during its execution, and interrupts must not have been disabled or surpressed prior to exercising the TRAP escape (Console Interrupt).

2. The O (Status Word) register must have been preset to enable interrupts prior to exiting DBG (J function), and the users program must not have disabled or surpressed interrupts prior to exercising the TRAP escape (Console Interrupt).

3. Console Interrupt Enable must be outside Enable Interrupts (EIN) and Disable Interrupts (DIN) control.

TRAP may be set in one of two modes.

MODE 1:

If Power Fail Enable (PFE) and Console Interrupt Enable (CIE) have been placed outside Enable Interrupts (EIN) control, or, if the users program controls interrupts, TRAP may be set as follows:

T.

Under this condition a Jump and Store (JST) to DBG instruction will be placed at Console Interrupt location : 1E and a Console Interrupt instruction will be executed prior to each conditional (B function) or unconditional (J function) exit from DBG.

MODE 2:

If Power Fail (PFE) and Console Interrupt (CIE) are under Enable Interrupts (EIN) control, the user may request DBG to enable both Console Interrupts (CIE) and Enable Interrupts (EIN) on each conditional (B function) or unconditional (J function) exit from DBG. In the standard configuration for ALPHA series Processors, Console Interrupt is placed under EIN/DIN control.

Tn.

n may be any valid hexadecimal character (1-F).

Under this condition, a Jump and Store (JST) to DBG instruction will be placed at Console Interrupt location : 1E. A Console Interrupt (CIE) and Enable Interrupt (EIN) instruction will be executed prior to each conditional (B function) or unconditional (J function) exit from DBG.

When the TRAP Escape is utilized, the O (Status Word) register will not reflect the true interrupt state of the users program, since interrupts are disabled by execution of the JST instruction to DBG.

NOTE

If the Interrupt lines have been displaced, (i.e., Console Interrupt line location is : 11E), the data constant TRP1 must be changed to : 11E for TRAP to function normally.

TRAP uses location : FE in Page 0 to store the address of the TRAP Return Processor. If the Breakpoint link location : FF is moved, the TRAP link will also move (i.e., if the Breakpoint link (: FF) is moved to location n, the TRAP link will be at location (n-1).

Appendix A

DEBUG OPERATION SUMMARY

The procedure for loading and executing DBG in an ALPHA series processor is as follows:

1. Load DBG into memory using the Binary Loader (BLD) or Autoload; standard load location is : nDB0.

2. Enter DBG's initial load location into the Program Counter (P) register and depress RUN.

3. DBG will respond by printing its name followed by the query character (>). At that time any of the commands summarized in Appendix B may be entered.

Appendix B

DEBUG COMMAND SUMMARY

≥A.	Display pseudo A register.
≥Av.	Set pseudo A register to value v.
≥Ba.	Continue breakpoint to location a.
≥Ba,b.	Continue breakpoint to location (a or b).
≥Ba@b.	Breakpoint from location a to b.
≥Ba@b,c.	Breakpoint from location a to location (b or c).
≥Ca@b@c.	Copy the contents of locations a through b at c and following.
≥Fa@b@v.	Fill locations a through b with value v.
≥Ia.	Inspect location a.
≥Ja.	Jump to location a.
≥La@b.	List the contents of locations a through b.
≥Ma.	Modify memory starting at location a.
≥O.	Display pseudo O register.
≥Ov.	Set pseudo O register to value v.
≥Pa@b.	Print the contents of locations a through b.
≥Rn.	Display relocation register Rn.
≥Rnv.	Set relocation register Rn to value v.
≥Sa@b@v.	Search locations a through b for value v.
≥Sa@b@v@m.	Search locations a through b for value v using mask word m.
≥T.	Enable Console Interrupt (TRAP).
≥Tn.	Enable Console Interrupt and Enable Interrupts.
≥X.	Display pseudo X register.
≥Xv.	Set pseudo X register to value v.

```
0002                *       COPYRIGHT 1972, COMPUTER AUTOMATION, INC.
0003                *
0004                *       DEBUG, A HEXIDECIMAL DEBUGGING PACKAGE.
0005                *       THIS PROGRAM IS BINARY RELOCATABLE.
0006                *       THE OBJECT TAPE PRODUCED BY ASSEMBLING
0007                *       THIS PROGRAM MAY BE LOADED USING THE
0008                *       BINARY LOADER (BLD) OR AUTOLOAD.
0009                *
0010                *       NOTE: THE BREAKPOINT FUNCTION USES ONE
0011                *       LOCATION IN THE USERS PROGRAM AND LOC :FF
0012                *       IN THE BASE PAGE AS RETURN LINKAGE. IF
0013                *       LOCATION :FF IS UNDESIREABLE, THE DATA
0014                *       CONSTANT BSET FOUND IN THE DATA TABLE
0015                *       MAY BE CHANGED TO THE DESIRED LOCATION.
0016                *       THE TRAP FUNCTION USES CONSOLE INTERRUPT
0017                *       LOCATION :1E AND BASE PAGE LOCATION :FE
0018                *       AS TRAP RETURN LINKAGE LOCATIONS. IF THE
0019                *       INTERRUPT LINES HAVE BEEN DISPLACED, THE
0020                *       DATA CONSTANT TRP1 MAY BE CHANGED TO :11E
0021                *       AND TRAP WILL FUNCTION NORMALLY. THE TRAP
0022                *       FUNCTION RETURN ADDRESS LOCATION (:FE),
0023                *       HOWEVER, WILL ALWAYS BE ONE LESS THAN THE
0024                *       BREAKPOINT RETURN ADDRESS LOCATION (:FF).
0025                *
```

```
0027                *          DEBUG COMMAND SUMMARY                *
0028                *                                               *
0029                *     THE CHARACTER '>' IS TYPED BY DEBUG TO    *
0030                *     SIGNAL 'READY FOR COMMAND'.               *
0031                *     THE '@' SYMBOL DENOTES BLANK.             *
0032                *     ALL OTHER CHARACTERS MUST BE TYPED BY     *
0033                *     THE USER.                                 *
0034                *                                               *
0035                *     >A.           DISPLAY PSEUDO A REGISTER.  *
0036                *     >AV.          SET PSEUDO A REGISTER TO    *
0037                *                   VALUE V.                    *
0038                *     >BA.          CONTINUE BREAKPOINT TO      *
0039                *                   LOCATION A.                 *
0040                *     >BA,B.        CONTINUE BREAKPOINT TO      *
0041                *                   LOCATION (A OR B).          *
0042                *     >BA@B.        BREAKPOINT FROM LOCATION    *
0043                *                   A TO B.                     *
0044                *     >BA@B,C.      BREAKPOINT FROM LOCATION    *
0045                *                   A TO LOCATION (B OR C).     *
0046                *     >CA@B@C.      COPY LOCATIONS A THROUGH    *
0047                *                   B AT C AND FOLLOWING.       *
0048                *     >FA@B@V.      FILL LOCATIONS A THROUGH    *
0049                *                   B WITH VALUE V.             *
0050                *     >IA.          INSPECT LOCATION A.         *
0051                *     >JA.          JUMP TO LOCATION A.         *
0052                *     >LA@B.        LIST THE CONTENTS OF        *
0053                *                   LOCATIONS A THROUGH B.      *
0054                *     >MA.          MODIFY MEMORY STARTING AT   *
0055                *                   LOCATION A.                 *
0056                *     >O.           DISPLAY PSEUDO O REGISTER.  *
0057                *     >OV.          SET PSEUDO O REGISTER TO    *
0058                *                   VALUE V.                    *
0059                *     >PA@B.        PRINT LOCATIONS A THROUGH   *
0060                *                   B.                          *
0061                *     >RN.          DISPLAY RELOCATION REGIS-   *
0062                *                   TER N.                      *
0063                *     >RNV.         SET RELOCATION REGISTER     *
0064                *                   RN TO VALUE V.              *
0065                *     >SA@B@V.      SEARCH LOCATIONS A THROUGH  *
0066                *                   B FOR VALUE V.              *
0067                *     >SA@B@V@M.    SEARCH FOR VALUE V USING    *
0068                *                   MASK WORD M.                *
0069                *     >T.           ENABLE CONSOLE INTERRUPT    *
0070                *                   (TRAP) ONLY.                *
0071                *     >TN.          ENABLE CONSOLE INTERRUPT    *
0072                *                   (TRAP) AND ENABLE INTERR-   *
0073                *                   UPTS ON EACH EXIT.          *
0074                *     >X.           DISPLAY PSEUDO X REGISTER.  *
0075                *     >XV.          SET PSEUDO X REGISTER TO    *
0076                *                   VALUE V.                    *
```

```
0078  0DB0                    REL   :DB0      STANDARD LOAD ADDRESS
0079                    *
0080        0007        TDA     EQU  7        STANDARD TTY DEVICE ADDRESS
0081                    *
0082                    *       THE JST TO BASE INSTRUCTION IS TO
0083                    *       INSURE THE BINARY RELOCATIBILITY OF
0084                    *       THIS PROGRAM.
0085  0DB0  FA00        DBUG    JST  BASE     SET LOCATION OF DEBUG
0086  0DB1  0000        BASE    DATA 0        CONTAINS CURRENT LOC OF DBG
0087  0DB2  0F0A              SWM
0088  0DB3  0110              ZAR
0089  0DB4  9A2A              STA  TFND     SET TRAP STATUS
0090  0DB5  3010              ARM
0091  0DB6  483F              SSN  TDA,7    FULL DUPLEX?
0092  0DB7  9A28              STA  DUPLEX   YES
0093  0DB8  9AFC              STA  LP       RESET LINE PRINTER FUNCTION
0094  0DB9  FAFC              JST  CRLF     NEW LINE.
0095  0DBA  C6C4              LAP  'D'
0096  0DBB  FAED              JST  TYP
0097  0DBC  C6C2              LAP  'B'
0098  0DBD  FAE8              JST  TYP
0099  0DBE  C6C7              LAP  'G'
0100  0DBF  FAE9              JST  TYP      OUTPUT NAME
0101  0DC0  B2DF              LDA  BFLG     GET BREAKPOINT FLG
0102  0DC1  3115              JAN  TRTN+1   FIX BREAKPOINT ERROR
0103  0DC2  F29C              JMP  DBG2     GET FUNCTION REQUEST
0104                    *
0105                    *       DATA CELLS AR,XR,OV,AND R0 THRU RF
0106                    *                 MUST ALWAYS BE CONTIGUOUS.
0107  0DC3  0000        AR      DATA 0        PSEUDO AR
0108  0DC4  0000        XR      DATA 0        PSEUDO XR
0109  0DC5  0000        OV      DATA 0        PSEUDO OV
0110                    *       THE FOLLOWING 16 LOCATIONS ARE RESERVED
0111                    *            FOR THE RELOCATION REGS
0112  0DC6  0000        R0      DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      0DC7  0000
      0DC8  0000
      0DC9  0000
      0DCA  0000
      0DCB  0000
      0DCC  0000
      0DCD  0000
      0DCE  0000
      0DCF  0000
      0DD0  0000
      0DD1  0000
      0DD2  0000
      0DD3  0000
      0DD4  0000
      0DD5  0000
0113                    *
0114                    *       TRTN -- FORM TRAP RETURN ADDRESS
```

```
0115  0DD6  0800  TRTN   ENT
0116  0DD7  6807         SIN  6
0117  0DD8  9E15         STA  AR        SAVE A REGISTER
0118  0DD9  DE03         IMS  TRTN      BUMP ADDRESS
0119  0DDA  B604         LDA  TRTN
0120  0DDB  9A4B         STA  RTN       LOCATION OF BRKPNT RTN PROC
0121  0DDC  F24D         JMP  RTN3      PROCESS RETURN
0122                *
0123                *      THE FOLLOWING FOUR DATA STATEMENTS HAVE BEEN
0124                *      LOCATED HERE TO INSURE NO PAGE ZERO REFS.
0125                **
0126  0DDD  0000  PARCTL DATA 0          BRKPNT PARAM CNTL
0127  0DDE  001E  TRP1   DATA :1E       CONSOLE INTRPT LOCATION
0128  0DDF  0000  TFND   DATA 0         TRAP STATUS WORD
0129  0DE0  0000  DUPLEX DATA 0         TTY DUPLEX MODE (-1=FULL)
0130                *
0131                *      TRAP FUNCTION
0132                *
0133  0DE1  B2BF  TRAP1  LDA  BSET      BRKPNT LINK ADDR
0134  0DE2  00D0         DAR
0135  0DE3  0048         TAX
0136  0DE4  8AB8         ADD  RJMP      RETURN JUMP INSTR
0137  0DE5  9F07         STA  *TRP1     PUT IN CI LOCATION
0138  0DE6  C625         LAP  TRTN-BASE
0139  0DE7  8E36         ADD  BASE
0140  0DE8  9C00         STA  @0        PUT IN LOCATION :FE
0141  0DE9  E2B8         LDX  ADDR
0142  0DEA  3801         JXN  $+2       ENABLE INTERRUPTS ALSO
0143  0DEB  0010         ARM
0144  0DEC  9E0D         STA  TFND      YES
0145  0DED  F274         JMP  DBG       NEXT FUNC
0146                *
0147                *      JMP   JUMP FUNCTION
0148                *
0149  0DEE  E2B3  JMP    LDX  ADDR      GET AND SET EXIT ADDR
0150  0DEF  EAAB         STX  EXIT
0151  0DF0  FAC5  JMP1   JST  CRLF      NEW LINE
0152  0DF1  E62D         LDX  XR        SET XREG
0153  0DF2  B62D         LDA  OV        GET INTERRUPT STATUS
0154  0DF3  11D2         RRA  3
0155  0DF4  0C00         DIN
0156  0DF5  B616         LDA  TFND      SEE IF TRAP WANTED
0157  0DF6  2101         JAZ  $+2       NO IF ZERO
0158  0DF7  4005         CIE
0159  0DF8  3181         JAG  $+2       INTERRUPTS WANTED
0160  0DF9  3201         JOR  $+2       INTERRUPTS NOT WANTED
0161  0DFA  0A00         EIN
0162  0DFB  6805         SIN  4         KEEP OUT BYTE MODE
0163  0DFC  C601         LAP  1         SET FULL DUPLEX COMMAND
0164  0DFD  DE1D         IMS  DUPLEX
0165  0DFE  F201         JMP  $+2       HALF DUPLEX
0166  0DFF  443C         SEA  TDA,4     ISSUE FULL COMMAND
```

```
0167  0E00  6805          SIN  4
0168  0E01  B63C          LDA  OV        GET AND OUTPUT STATUS
0169  0E02  6C00          SOA
0170  0E03  B640          LDA  AR        SET AREG
0171  0E04  F396          JMP  *EXIT     EXIT TO USER
0172                *
0173                *     BRK  BREAKPOINT FUNCTION
0174                *
0175  0E05  E298    BRK   LDX  VALU      GET ADDR
0176  0E06  C0A0          CAI  '         '       IF NEW BREAK
0177  0E07  F201          JMP  $+2
0178  0E08  F204          JMP  BRK1      IF MORE THAN 1 BRK
0179  0E09  EA91          STX  EXIT      ELSE SET EXIT
0180  0E0A  FAB6          JST  INP       GET 1ST BRKPNT ADDR
0181  0E0B  C0A0          CAI  '         '
0182  0E0C  F281          JMP  ERR       ERROR IF 3 PARAMS
0183  0E0D  DE30    BRK1  IMS  PARCTL    COUNT BREAKPOINTS
0184  0E0E  EA8D          STX  RTRN      SAVE BRK ADDR
0185  0E0F  9A90          STA  BFLG      SET BREAKPOINT FLAG
0186  0E10  B28C          LDA  RJMP      GET RETURN JMP INSTR
0187  0E11  8A8F          ADD  BSET      ADD PAGE 0 LOC
0188  0E12  BC00          EMA  *0        SET AND GET USERS INSTR
0189  0E13  E282          LDX  MB        ADDR OF SAVE TBL
0190  0E14  0128          IXR
0191  0E15  9C00          STA  *0        SAVE USERS INSTR
0192  0E16  0128          IXR
0193  0E17  B284          LDA  RTRN      ADDR OF USERS INSTR
0194  0E18  9C00          STA  *0        SAVE ADDR
0195  0E19  EA7C          STX  MB
0196  0E1A  C676          LAP  RTN-BASE
0197  0E1B  8E6A          ADD  BASE
0198  0E1C  9B84          STA  *BSET     SET UP RETURN ADDRESS IN PAGE 0
0199  0E1D  B282          LDA  BFLG      GET TERMINATOR
0200  0E1E  C0AC          CAI  ','
0201  0E1F  F201          JMP  NOGO      ANOTHER BRK NEEDED
0202  0E20  F630          JMP  JMP1      GO SET UP REGS AND EXIT
0203  0E21  FA9F    NOGO  JST  INP       GET 2ND BREAK ADDR
0204  0E22  C0AC          CAI  ','
0205  0E23  F26A          JMP  ERR       ERROR---CLEAR BREAKPOINT
0206  0E24  C0A0          CAI  '         '
0207  0E25  F268          JMP  ERR       ERROR---CLEAR BREAKPOINT
0208  0E26  F619          JMP  BRK1
0209                *
0210                *     RTN  BREAKPOINT RETURN PROCESSOR
0211                *
0212  0E27  0800    RTN   ENT
0213  0E28  6802    RTN2  SIN  1
0214  0E29  9E66          STA  AR        SAVE AREG
0215  0E2A  5800    RTN3  SIA
0216  0E2B  9E66          STA  OV        SAVE STATUS
0217  0E2C  EE68          STX  XR        SAVE XREG
0218  0E2D  0010          ARM
```

```
0219  0E2E  483F          SSN   TDA,7     GET DUPLEX MODE
0220  0E2F  9E4F          STA   DUPLEX    FULL
0221  0E30  B653          LDA   PARCTL    CONSOLE INTRPT RETURN?
0222  0E31  210E          JAZ   PRINT     YES---GO PRINT REGS
0223  0E32  E263  RTN1    LDX   MB
0224  0E33  B400          LDA   *0        GET USERS INST ADDR
0225  0E34  9A69          STA   VALU      SAVE
0226  0E35  00A8          DXR
0227  0E36  B400          LDA   *0        GET USERS INSTR
0228  0E37  00A8          DXR
0229  0E38  EA5D          STX   MB        SAVE PLACE IN TBL
0230  0E39  E264          LDX   VALU
0231  0E3A  9C00          STA   *0        RESTORE USERS INSTR
0232  0E3B  B65E          LDA   PARCTL    HOW MANY BRKPNTS ?
0233  0E3C  00D0          DAR
0234  0E3D  9E60          STA   PARCTL    RESET BRKPNT CNTR
0235  0E3E  9A61          STA   BFLG      RESET BRKPNT FLAG
0236  0E3F  F60E          JMP   RTN1-1
0237  0E40  FA75  PRINT   JST   CRLF      NEW TTY LINE
0238  0E41  B61A          LDA   RTN       GET BREAK ADDR
0239  0E42  00D0          DAR
0240  0E43  9A57          STA   EXIT      NEW EXIT ADDR
0241  0E44  FABA          JST   HEX       PRINT VALUES
0242  0E45  B682          LDA   AR        TYPE REGISTER VALUES
0243  0E46  FAB8          JST   HEX
0244  0E47  B683          LDA   XR
0245  0E48  FAB6          JST   HEX
0246  0E49  B684          LDA   OV
0247  0E4A  FAB4          JST   HEX
0248  0E4B  F216          JMP   DBG       NEXT FUNCTION
0249                *
0250                *     REG   DISPLAY/CHANGE PSEUDO REGISTERS
0251                *
0252  0E4C  8AC4  REG     ADD   SIZE      ADD REL REG NO.
0253  0E4D  0150          IAR
0254  0E4E  0150  OREG    IAR
0255  0E4F  0150  XREG    IAR
0256  0E50  9A51  AREG    STA   ADDR      SAVE ADDR OF REG
0257  0E51  B24D          LDA   PFLG      IF VALUE SUPPLIED
0258  0E52  3109          JAN   REG1      GO CHANGE REG
0259  0E53  C6A0          LAP   '         '
0260  0E54  FA54          JST   TYP       ELSE TYPE BLANK
0261  0E55  E24C          LDX   ADDR      GET ADDR OF REG
0262  0E56  B412          LDA   *AR-BASE  GET REG VALUE
0263  0E57  FAA7          JST   HEX       AND TYPE
0264  0E58  FA68          JST   INP       GET CHANGE
0265  0E59  FAD6          JST   COMMA     ERROR IF COMMA
0266  0E5A  B244          LDA   PFLG      IF NO CHANGE
0267  0E5B  2106          JAZ   DBG       NEXT FUNCTION
0268  0E5C  E245  REG1    LDX   ADDR      ELSE GET ADDR OF REG
0269  0E5D  B240          LDA   VALU      GET CHANGE
0270  0E5E  9C12          STA   *AR-BASE  AND SET NEW VALUE
```

```
0271                    *
0272                    *       DBG   GET FUNCTION ID AND FIRST PARAMETER
0273                    *
0274  0E5F  C6E5  DBG2    LAP   MB-BASE
0275  0E60  8EAF          ADD   BASE
0276  0E61  9A34          STA   MB          BRKPNT SAVTBL LOC CALC.
0277  0E62  FA53  DBG     JST   CRLF        NEW LINE
0278  0E63  C6BE          LAP   '>'
0279  0E64  FA44          JST   TYP         TYPE > (READY FOR COMMAND)
0280  0E65  FA93          JST   RKB         GET FUNCTION ID
0281  0E66  9AAB          STA   KEY         AND SAVE
0282  0E67  C0D2          CAI   'R'
0283  0E68  F201          JMP   $+2
0284  0E69  F203          JMP   DBG1        IF REL REG...
0285  0E6A  FA8F          JST   RKB         GET REG NO.
0286  0E6B  FA81          JST   CHK         CHK IF HEX DIGIT
0287  0E6C  9AA4          STA   SIZE        SAVE REG NO.
0288  0E6D  FA53  DBG1    JST   INP         GET 1ST PARAMETER
0289  0E6E  EA33          STX   ADDR        SET ADDR
0290  0E6F  E2A2          LDX   KEY         GET FUNCTION ID
0291  0E70  C1C2          CXI   'B'
0292  0E71  F66C          JMP   BRK         BREAKPOINT
0293  0E72  C1C3          CXI   'C'
0294  0E73  F2EF          JMP   COPY
0295  0E74  C1C6          CXI   'F'
0296  0E75  F2E6          JMP   FILL
0297  0E76  C1CC          CXI   'L'
0298  0E77  F2C3          JMP   LIST
0299  0E78  C1D0          CXI   'P'
0300  0E79  F2C5          JMP   PRNT        PRINT
0301  0E7A  C1D3          CXI   'S'
0302  0E7B  F219          JMP   ERR2        SEARCH
0303  0E7C  FADB          JST   PERD        ERROR IF TWO PARAMS AT THIS POIN
0304  0E7D  C1D4          CXI   'T'
0305  0E7E  F69D          JMP   TRAP1       CONSOLE INTRPT WANTED
0306  0E7F  C1CA          CXI   'J'
0307  0E80  F692          JMP   JMP         JUMP
0308  0E81  C1C9          CXI   'I'
0309  0E82  F290          JMP   INSP        INSPECT
0310  0E83  C1CD          CXI   'M'
0311  0E84  F2AF          JMP   MOD         MODIFY
0312  0E85  B6D4          LDA   BASE        ELSE PSEUDO REG, GET ORIGIN
0313  0E86  C1D2          CXI   'R'
0314  0E87  F63B          JMP   REG         RELOCATION REG
0315  0E88  C1C1          CXI   'A'
0316  0E89  F639          JMP   AREG        A REG
0317  0E8A  C1D8          CXI   'X'
0318  0E8B  F63C          JMP   XREG        XREG
0319  0E8C  C1CF          CXI   'O'
0320  0E8D  F63F          JMP   OREG        OV REG
0321                    *
0322                    *       ERR   FORMAT ERROR
```

```
0323                    *
0324  0E8E  C6DF  ERR     LAP   :DF
0325  0E8F  9A25          STA   LP
0326  0E90  FA18          JST   TYP       TYPE LEFT ARROW
0327  0E91  86B4          LDA   PARCTL    HAS A BREAKPOINT BEEN SET?
0328  0E92  3101          JAN   $+2       YES
0329  0E93  F631          JMP   DBG       NEXT FUNCTION
0330  0E94  F66C          JMP   RTN2      GO FIX BREAKPOINT---ERROR
0331  0E95  F2DA  ERR2    JMP   SRCH      LINK TO SEARCH ROUTINE
0332                    *
0333                    *       DATA AREA FOR DBUG
0334                    *
0335  0E96  0000  MB      DATA  0         INSURES BINARY RELOC.
0336                    *                 OF BRKPOINT SAVTBL.
0337  0E97  0000          DATA  0,0,0,0   START OF BRKPNT SAVTBL
      0E98  0000
      0E99  0000
      0E9A  0000
0338  0E9B  0000  EXIT    DATA  0         EXIT ADDR FOR JMP/BRK
0339  0E9C  0000  RTRN    DATA  0         RETURN (BREAKPOINT) ADDR
0340  0E9D  F900  RJMP    JST   *0        PAGE ZERO BRKPNT
0341                    *       THE VALUE OF RLOC(RTN-BASE) CANNOT BE
0342                    *                 GREATER THAN 255
0343  0E9E  0000  VALU    DATA  0         PARAMETER VALUE
0344  0E9F  0000  PFLG    DATA  0         AND FLAG PO X NO PARAMETER)
0345  0EA0  0000  BFLG    DATA  0         BREAKPOINT FLAG
0346  0EA1  00FF  BSET    DATA  :FF       PAGE 0 BREAKPOINT LINK ADDRESS
0347  0EA2  0000  ADDR    DATA  0         1ST PARAMETER
0348                    *
0349                    *     INP   INPUT A PARAMETER
0350                    *
0351  0EA3  FA55  INP3      JST   RKB     GET REL. REG. NO.
0352  0EA4  FA48            JST   CHK
0353  0EA5  8EF4            ADD   BASE
0354  0EA6  3048            TAX
0355  0EA7  B415            LDA   @RO-BASE GET REL. VALUE
0356  0EA8  F230            JMP   INP2
0357                    *
0358                    *     TYP   TYPE AR ON TELETYPE
0359                    *
0360  0EA9  0800  TYP       ENT
0361  0EAA  BA0A            EMA   LP      SEE IF LINE PRINTER OUTPUT
0362  0EAB  2106            JAZ   TYP1    IF YES
0363  0EAC  BA08            EMA   LP
0364  0EAD  403C            SEL   TDA,4   SET OUTPUT SIDE
0365  0EAE  6C38            OTA   TDA,0   OUTPUT
0366  0EAF  4939            SEN   TDA,1   WAIT TILL DONE
0367  0EB0  F601            JMP   $-1
0368  0EB1  F708            RTN   TYP     RETURN
0369  0EB2  BA02  TYP1      EMA   LP
0370  0EB3  6D21            WRA   :21
0371  0EB4  F70B            RTN   TYP
```

```
0372  0EB5  0000  LP      DATA 0
0373                *
0374                *     CRLF  CARRIAGE RETURN/LINE FEED
0375                *
0376  0EB6  0800  CRLF    ENT
0377  0EB7  B2F1          LDA   TCON
0378  0EB8  00D0          DAR                TIME OUT
0379  0EB9  3141          JAN   $-1
0380  0EBA  C68D          LAP   :8D
0381  0EBB  FE12          JST   TYP          TYPE CR
0382  0EBC  B607          LDA   LP           SEE IF LINE PRINTER OUTPUT
0383  0EBD  2102          JAZ   $+3
0384  0EBE  C68A          LAP   :8A
0385  0EBF  FE16  NOP:ME  JST   TYP          TYPE LF
0386  0EC0  F70A          RTN   CRLF         EXIT
0387                *
0388  0EC1  0800  INP     ENT
0389  0EC2  0108          ZXR
0390  0EC3  EACB          STX   TEST         INPUT PARAMETER 1
0391  0EC4  EE0E          STX   CRLF         ASSUME + CONNECTOR
0392  0EC5  EE27          STX   VALU         READY FOR NEW PARAM.
0393  0EC6  EE27          STX   PFLG         NO PARAM YET
0394  0EC7  FA31  INP1    JST   RKB          IF MINUS GET SECOND CHAR
0395  0EC8  C0AF          CAI   '/'
0396  0EC9  F607          JMP   INP+1        START OVER IF /
0397  0ECA  C0A0          CAI   '  '
0398  0ECB  F211          JMP   INP5         PROCESS
0399  0ECC  C0AF          CAI   '.'
0400  0ECD  F20F          JMP   INP5         PROCESS
0401  0ECE  C0D2          CAI   'R'
0402  0ECF  F62C          JMP   INP3
0403  0ED0  C0AC          CAI   ','
0404  0ED1  F208          JMP   INP5         PROCESS
0405  0ED2  C0AB          CAI   '+'
0406  0ED3  F209          JMP   INP5
0407  0ED4  C0AD          CAI   '-'
0408  0ED5  F207          JMP   INP5
0409  0ED6  FA16          JST   CHK          CHECK HEXADECIMAL
0410  0ED7  8E21          EMA   CRLF         BUILD HEX VALUE
0411  0ED8  1353          LLA   4            SHIFT LEFT 1 HEX DIGIT
0412  0ED9  8E23  INP2    ADD   CRLF         ADD DIGIT
0413  0EDA  9E24          STA   CRLF         SAVE VALUE
0414  0EDB  DE3C          IMS   PFLG         PARAMETER ENTERED
0415  0EDC  F615          JMP   INP1         GET NEXT CHAR
0416  0EDD  BAB1  INP5    EMA   TEST         GET LAST SIGN (+-)
0417  0EDE  E628          LDX   CRLF         GET WORKING VALUE
0418  0EDF  C0AD          CAI   '-'          NEGATE?
0419  0EE0  0508          NXR                IF MINUS NEGATE
0420  0EF1  0030          TXA
0421  0EE2  8E44          ADD   VALU         ADD OLD VALUE
0422  0EF3  9E45          STA   VALU
0423  0EE4  0048          TAX                SAVE
```

```
PAGE  0010  09/17/73  16:29:29  ** DBG (DEBUG) 96004-10C1 **

0424  0LF5  0110          ZAR             INITIALIZE FOR NEXT
0425  0LF6  9E30          STA   CRLF
0426  0LF7  B2A7          LDA   TEST      GET DELIMITER
0427  0LF8  C0AB          CAI   '+'       PLUS?
0428  0LF9  F622          JMP   INP1      YES, CONTINUE
0429  0LFA  C0AD          CAI   '-'
0430  0LFB  F624          JMP   INP1      CONTINUE
0431  0LFC  F72B          RTN   INP       EXIT
0432                *
0433                *       CHK   CHECK FOR HEXADECIMAL AND CONVERT
0434                *
0435  0LFD  0B0A  CHK       ENT
0436  0LFE  92BD          SUB   HBA       TEST FOR DECIMAL DIGIT
0437  0LFF  3083          JAP   CHK2      NOT DECIMAL DIGIT
0438  0LF0  8ABD          ADD   HA        ADD 10
0439  0LF1  3086          JAP   CHK3      DECIMAL DIGIT
0440  0LF2  F664  CHK1    JMP   ERR       NOT HEXADECIMAL
0441  0LF3  92B9  CHK2    SUB   HD        -13
0442  0LF4  30C2          JAP   CHK1      NOT A THRU F
0443  0LF5  8AB9          ADD   H6        ADD 6
0444  0LF6  20C4          JAM   CHK1      NOT A THRU F
0445  0LF7  8AB6          ADD   HA        ADD 10 FOR VALUES 10 THRU 15
0446  0LF8  F70B  CHK3    JMP   *CHK      RETURN, AR = VALUE
0447                *
0448                *       RKB   READ KEYBOARD CHARACTER
0449                *
0450  0LF9  0B0B  RKB       ENT
0451  0LFA  4038          SEL   TDA,0     AUTO-ECHO ON
0452  0LFB  5939          RDA   TDA,1     INPUT CHARACTER
0453  0LFC  A2AF          IOR   H80       FORCE BIT 7 ON
0454  0LFD  403C          SEL   TDA,4     AUTO-ECHO OFF
0455  0LFE  F705          JMP   *RKB
0456                *
0457                *       HEX   TYPE AR AS 4 DIGIT HEXADECIMAL
0458                *
0459  0LFF  0B0B  HEX       ENT
0460  0F00  CE61          STX   PFLG      SAVE XR
0461  0F01  004B          TAX             VALU TO XREG
0462  0F02  1400          SOV             SET END FLAG
0463  0F03  0110  HEX1      ZAR             CLEAR AR
0464  0F04  1903          LRL   4         1 HEX DIGIT TO AR
0465  0F05  3804          JXN   HEX2      JMP IF NOT DONE
0466  0F06  C0A0          LAP   ' '       '      ELSE TYPE A BLANK
0467  0F07  FE5F          JST   TYP
0468  0F08  E069          LDX   PFLG      RESTORE XR
0469  0F09  F70A          JMP   *HEX      AND RETURN
0470  0F0A  92A3  HEX2    SUB   HA        CONVERT TO HEX
0471  0F0B  2082          JAM   $+3       JUMP IF 0 THRU 9
0472  0F0C  8AA2          ADD   H6        ELSE MAKE A THRU E
0473  0F0D  0150          IAR
0474  0F0E  8A9D          ADD   HBA       ADD ASCII OFFSET
0475  0F0F  FE66          JST   TYP       TYPE DIGIT
```

```
0476  0F10  F60D          JMP  HEX1      LOOP
0477  0F11  0000  SIZE    DATA 0         2ND
0478  0F12  0000  KEY     DATA 0         3RD
0479                      *
0480                      *      INSP  INSPECT FUNCTION
0481                      *
0482  0F13  FE5D  INSP    JST  CRLF      NEW LINE
0483  0F14  B772          LDA  *ADDR
0484  0F15  9E77          STA  VALU      CONTENTS OF EFFECTIVE ADDR
0485  0F16  B674          LDA  ADDR
0486  0F17  3084          JAP  INSP1     NOT INDIRECT
0487  0F18  8291          AND  ICON      DROP INDIRECT BIT
0488  0F19  9E77          STA  ADDR
0489  0F1A  B778          LDA  *ADDR     PICK UP EFFECTIVE ADDR
0490  0F1B  9E79          STA  ADDR      SAVE EFFECTIVE ADDR
0491  0F1C  FE1D  INSP1   JST  HEX       TYPE ADDR
0492  0F1D  B67F          LDA  VALU
0493  0F1E  FE1F          JST  HEX       TYPE CONTENTS
0494  0F1F  FE5E          JST  INP       GET CHANGE
0495  0F20  E681          LDX  PFLG
0496  0F21  2802          JXZ  $+3       NO CHANGE IF NO PARAM
0497  0F22  E684          LDX  VALU      ELSE GET VALUE
0498  0F23  EF81          STX  *ADDR     CHANGE
0499  0F24  E682          LDX  ADDR
0500  0F25  C0A0          CAI  ' '       '
0501  0F26  F203          JMP  INCR      INCREMENT ADDR IF BLANK
0502  0F27  C0AC          CAI  ','
0503  0F28  F202          JMP  DCRM      DECREMENT ADDR IF COMMA
0504  0F29  F6C7  INSP2   JMP  DBG       OR ELSE NEXT FUNCTION
0505  0F2A  C202  INCR    AXI  2
0506  0F2B  C301  DCRM    SXI  1
0507  0F2C  0030          TXA
0508  0F2D  20C4          JAM  INSP2     ERROR IF NEG ADDR
0509  0F2E  9E8C          STA  ADDR      SAVE ADDR AND
0510  0F2F  F61C          JMP  INSP      LOOK AGAIN
0511                      *
0512                      *      COMMA----ERROR IF COMMA
0513                      *       COMMA IS CALLED BY THE FOLLOWING ROUTINES
0514                      *       TO CHECK FOR A VALID DELIMITER: SRCH,COPY,
0515                      *       FILL, PRNT,GSZ,GET2,AND REG.
0516                      *
0517  0F30  0800  COMMA   ENT
0518  0F31  C0AC          CAI  ','
0519  0F32  F6A4  COMMA1  JMP  ERR
0520  0F33  F703          RTN  COMMA
0521                      *
0522                      *      MOD   MODIFY FUNCTION
0523                      *
0524  0F34  FE73  MOD     JST  INP       GET WORD
0525  0F35  FE05          JST  COMMA     ERROR IF COMMA
0526  0F36  EF94          STX  *ADDR     AND STORE
0527  0F37  DE96          IMS  ADDR      BUMP ADDR PTR
```

```
0528 0F33 C0A0          CAI  '          '
0529 0F39 F005          JMP  MOD        LOOP IF BLANK
0530 0F3A F011  MOD1    JMP  INSP2
0531                    *
0532                    *       PRNT  PRINT FUNCTION
0533                    *
0534 0F3B FE0B  LIST    JST  COMMA      ERROR IF COMMA
0535 0F3C 3110          ZAR
0536 0F3D 9E88          STA  LP
0537 0F3E FA64          JST  TOF
0538 0F3F FE0F  PRNT    JST  COMMA      ERROR IF COMMA
0539 0F40 FA42          JST  GSZ        GET SIZE
0540 0F41 FA16          JST  PERD       ERROR IF NOT PERIOD
0541 0F42 E631          LDX  SIZE       NO. OF WORDS
0542 0F43 C708  PRN1    LAM  8          SET FOR 8 WORDS/LINE
0543 0F44 9E32          STA  KEY
0544 0F45 FE8F          JST  CRLF       NEW LINE
0545 0F46 4039          SEL  TDA,1      SET INPUT MODE
0546 0F47 0150          IAR             TIME OUT
0547 0F48 3141          JAN  $-1        TIME OUT
0548 0F49 4839          SSN  TDA,1
0549 0F4A F24F          JMP  TEST1      STOP LISTING
0550 0F4B B049          LDA  ADDR
0551 0F4C FE4D          JST  HEX        TYPE ADDR
0552 0F4D B7A8  PRN2    LDA  *ADDR
0553 0F4E FE4F          JST  HEX        TYPE WORD
0554 0F4F FA3F          JST  TEST       NEXT FUNCTION IF DONE
0555 0F50 DE3E          IMS  KEY
0556 0F51 F004          JMP  PRN2       NEXT WORD IF NOT END OF LINE
0557 0F52 B09D          LDA  LP         SEE IF LINE PRINTER OUTPUT
0558 0F53 3150          JAN  PRN1
0559 0F54 DA45          IMS  GET2       BUMP LINE COUNT
0560 0F55 F012          JMP  PRN1       ELSE NEXT LINE
0561 0F56 FA4C          JST  TOF        OUTPUT TOF
0562 0F57 F014          JMP  PRN1
0563                    *
0564                    *DELIMETER TEST          IF NOT PERIOD GO TO ERROR
0565                    *
0566 0F58 0800  PERD    ENT
0567 0F59 C0AE          CAI  '.'
0568 0F5A F702          JMP  *PERD      ELSE RETURN
0569 0F5B F6CD          JMP  ERR        ERROR IF NOT PERIOD
0570                    *
0571                    *       FILL  FILL FUNCTION
0572                    *
0573 0F5C FE2C  FILL    JST  COMMA      ERROR IF COMMA
0574 0F5D FA3C          JST  GET2       GET 2 PARAMS
0575 0F5E FE26          JST  PERD       ERROR IF NOT PERIOD
0576 0F5F B64D          LDA  KEY        GET CONSTANT
0577 0F60 9FBE  FIL1    STA  *ADDR      STORE IN MEMORY
0578 0F61 FA2D          JST  TEST       NEXT FUNCTION IF DONE
0579 0F62 F0F2          JMP  FIL1       ELSE LOOP
```

```
0580                    *
0581                    *       COPY    COPY FUNCTION
0582                    *
0583  0F63  FE33  COPY    JST   COMMA     ERROR IF COMMA
0584  0F64  FA35          JST   GET2      GET 2 PARAMS
0585  0F65  FE0D          JST   PERD      ERROR IF NOT PERIOD
0586  0F66  B6C4          LDA   ADDR      CHECK LEGALITY
0587  0F67  9655          SUB   KEY       OF PARAMETERS
0588  0F68  3082          JAP   CPY1      OK
0589  0F69  8E58          ADD   SIZE
0590  0F6A  31F8          JAG   COMMA1    ILLEGAL MOVE
0591  0F6B  B7C9  CPY1    LDA   *ADDR     GET WORD
0592  0F6C  9F5A          STA   *KEY      COPY
0593  0F6D  DE58          IMS   KEY       BUMP ADDR PTR
0594  0F6E  FA20          JST   TEST      NEXT FUNCTION IF DONE
0595  0F6F  F604          JMP   CPY1      ELSE LOOP
0596                    *
0597                    *       SRCH    SEARCH FUNCTION
0598                    *
0599  0F70  FE40  SRCH    JST   COMMA     ERROR IF COMMA
0600  0F71  FA28          JST   GET2      GET 2 PARAMS
0601  0F72  0008          XRM             SET MASK OFF
0602  0F73  C0A0          CAI   '         '
0603  0F74  FEB3          JST   INP       GET MASK IF COMMA
0604  0F75  EA2D          STX   MASK      AND SET MASK
0605  0F76  FE1E          JST   PERD      ERROR IF NOT PERIOD
0606  0F77  E666          LDX   SIZE      GET SIZE
0607  0F78  B7D6  SRC1    LDA   *ADDR     GET WORD
0608  0F79  AE67          XOR   KEY       COMPARE TO KEY
0609  0F7A  8228          AND   MASK      MASK
0610  0F7B  3105          JAN   SRC2      JUMP IF NOT EQUAL
0611  0F7C  FEC6          JST   CRLF      ELSE NEW LINE
0612  0F7D  B6DB          LDA   ADDR
0613  0F7E  FE7F          JST   HEX       TYPE ADDR
0614  0F7F  B7DD          LDA   *ADDR
0615  0F80  FE81          JST   HEX       TYPE CONTENTS
0616  0F81  FA0D  SRC2    JST   TEST      NEXT FUNCTION IF DONE
0617  0F82  F60A          JMP   SRC1      ELSE LOOP
0618                    *
0619                    *       GSZ     GET AND CHECK SIZE ^2ND PARAM<
0620                    *
0621  0F83  0800  GSZ     ENT
0622  0F84  C0AE          CAI   '.'
0623  0F85  F653  GSZ1    JMP   COMMA1    ERROR IF ONLY 1 PARAM
0624  0F86  FEC5          JST   INP       GET 2ND PARAM
0625  0F87  FE57          JST   COMMA     ERROR IF COMMA
0626  0F88  9E77          STA   SIZE      SAVE TERM CHAR
0627  0F89  0130          IXA             CALCULATE SIZE
0628  0F8A  96E8          SUB   ADDR
0629  0F8B  3181          JAG   $+2
0630  0F8C  F69A          JMP   CHK1      ERROR IF NEGATIVE
0631  0F8D  BE7C          EMA   SIZE      SET SIZE GET TERM CHAR
```

```
PAGE  0014  09/17/73  16:29:29  ** DBG (DEBUG) 96004-10C1 **

0032  0F8E  F700          JMP   *GSZ      RETURN
0033              *
0034              *       TEST  TEST FOR END OF PRINT/SRCH/COPY/FILL
0035              *
0036  0F8F  0800  TEST    ENT
0037  0F90  DEEE          IMS   ADDR      BUMP ADDR PTR
0038  0F91  00A8          DXR             COUNT WORDS
0039  0F92  2801          JXZ   $+2
0040  0F93  F704          JMP   *TEST     ELSE RETURN
0041  0F94  B6DF          LDA   LP        LINE PRINTER OUTPUT
0042  0F95  3101          JAN   $+2
0043  0F96  FEF0          JST   CRLF      PRINT LAST LINE
0044  0F97  0528  TEST1   XRP
0045  0F98  EEE3          STX   LP        RESET LINE PRINTER FLAG
0046  0F99  F65F          JMP   MOD1      NEXT FUNCTION IF DONE
0047              *
0048              *       GET2  INPUT 2ND AND 3RD PARAMETERS
0049              *
0050  0F9A  0800  GET2    ENT
0051  0F9B  FE1B          JST   GSZ       GET AND CHECK SIZE (PARAM 2)
0052  0F9C  C6AE          CAI   ','
0053  0F9D  F618          JMP   GSZ1      ERROR IF ONLY 2 PARAMS
0054  0F9E  FEDD          JST   INP       GET 3RD PARAM
0055  0F9F  FE6F          JST   COMMA     ERROR IF COMMA
0056  0FA0  EE8E          STX   KEY       AND SAVE
0057  0FA1  E690          LDX   SIZE      XR ? SIZE
0058  0FA2  F708          JMP   *GET2     RETURN
0059              *
0060              *       TOF   --  TOP-OF-FORM
0061              *
0062  0FA3  0800  TOF     ENT
0063        0FA3  MASK    EQU   TOF
0064  0FA4  C68C          LAP   :8C       OUTPUT TOF
0065  0FA5  6D21          WRA   :21
0066  0FA6  C730          LAM   60
0067  0FA7  9E90          STA   GET2      RESET LINE COUNT
0068  0FA8  F705          RTN   TOF
0069              *
0070  0FA9  125A  TCON    DATA  :125A
0071  0FAA  7FFF  ICON    DATA  :7FFF
0072  0FAB  0080  H80     DATA  :80       BIT 7
0073  0FAC  000A  HBA     DATA  :BA
0074  0FAD  000D  HD      DATA  :D        13
0075  0FAE  000A  HA      DATA  :A
0076  0FAF  0006  H6      DATA  :6
0077              *
0078              0DB0          END   DBUG
0000 ERRORS
```

```
0351  INP3       0402
0416  INP5       0398  0400  0404  0406  0408
0482  INSP       0309  0510
0491  INSP1      0486
0504  INSP2      0508  0530
0149  JMP        0307
0151  JMP1       0202
0478  KEY        0281* 0290  0543* 0555* 0576  0587  0592* 0593*
                 0608  0656*
0534  LIST       0298
0372  LP         0093* 0325* 0361* 0363* 0369* 0382  0536* 0557
                 0641  0645*
0663  MASK       0604* 0609
0335  MB         0189  0195* 0223  0229* 0274  0276*
0524  MOD        0311  0529
0530  MOD1       0646
0203  NOGO       0201
0385  NOP:ME
0254  OREG       0320
0109  OV         0153  0168  0216* 0246
0126  PARCTL     0183* 0221  0232  0234* 0327
0506  PERD       0303* 0540* 0568  0575* 0585* 0605*
0344  PFLG       0257  0266  0393* 0414* 0460* 0468  0495
0237  PRINT      0222
0542  PRN1       0558  0560  0562
0552  PRN2       0556
0538  PRNT       0300
0252  REG        0314
0268  REG1       0258
0340  RJMP       0136  0186
0450  RKB        0280* 0285* 0351* 0394* 0455
0112  RO         0355
0212  RTN        0120* 0196  0238
0223  RTN1       0236
0213  RTN2       0330
0215  RTN3       0121
0339  RTRN       0184* 0193
0477  SIZE       0252  0287* 0541  0589  0606  0626* 0631* 0657
0607  SRC1       0617
0616  SRC2       0610
0599  SRCH       0331
0670  TCON       0377
0080  TDA        0091  0166  0219  0364  0365  0366  0451  0452
                 0454  0545  0548
0636  TEST       0390* 0416* 0426  0554* 0578* 0594* 0616* 0640
0644  TEST1      0549
0128  TFND       0089* 0144* 0156
0662  TOF        0537* 0561* 0663  0668
0133  TRAP1      0305
0127  TRP1       0137*
0115  TRTN       0102  0118* 0119  0138
0360  TYP        0096* 0098* 0100* 0260* 0279* 0326* 0368  0371
```

```
                    0381* 0385* 0467* 0475*
    0369  TYP1      0362
    0343  VALU      0175  0225* 0230  0269  0392* 0421  0422* 0484*
                    0492  0497
    0108  XR        0152  0217* 0244
    0255  XREG      0318
0678 SOURCE LINES
```