ALPHA LSI AUTOLOAD

INTRODUCTION

The NAKED MINI LSI/ALPHA LSI Autoload program provides a fast, convenient method of program loading without the need of a resident bootstrap. Autoload is implemented in a Read-Only-Memory (ROM) and supports Teletype and High Speed Paper Tape, 9-track open reel Magnetic Tape, Digital Cassette and Moving Head Disk. Autoload is available as part of the NAKED MINI LSI/ALPHA LSI Autoload option.

ENVIRONMENT

The Autoload program does not occupy any memory, but does require the usage of base page locations : 30 through : 3B for scratch when loading from paper tape devices.

Autoload requires one or more of the following input devices:

ASR-33 or ASR-35 Teletype Reader
High Speed Paper Tape Reader
9-Track Open Reel Magnetic Tape
CAI Digital Cassette System
CAI Moving Head Disk System

If multiple magnetic tape, cassette or disk drives are available, Autoload will load from the device designated as unit 0.

FEATURES

- Selectively loads from any of the five devices.
- Loads most binary tapes as produced by the BETA assemblers.
- Loads all binary tapes produced by the Binary Punch (BDP) Program.
- Loads absolute or relative (at program specified origin or at operator specified origin).
- Features load-and-halt or load-and-execute modes of operation.
- Performs checksum and format verification on paper tape input.

LIMITATIONS

- Locations : 30 through : 3B must be available (not loaded into) for Autoload scratch use.

PROGRAM DESCRIPTION

The Autoload program provides two modes of operation dependent upon the device being utilized; "binary" format program tapes for teletype and high speed reader, and single record "bootstraps" for magnetic tape, cassette and disk.

Teletype and High Speed Reader:

The Autoload program will load (and optionally relocate) binary tapes produced by the Binary Dump Program (BDP). In addition, object programs as produced by the BETA assembler may be loaded directly if they do not contain any literals, external references or external definitions.

Because the Autoloader program uses locations : 30 through : 3B for scratch, these locations cannot be loaded and any prior contents of these locations will be destroyed by Autoload. If it is desired to load locations 0 to : 2F from a tape produced by BDP, these locations should be punched first followed by a second punch operation beginning at location : 3C or higher. Loading over locations : 30 through : 3B will cause erroneous operation of Autoload. An object program using locations 0 to : 2F should have the following form to load correctly:

```
ABS     0
 '              instructions for locations 0 to : 2F
 '
 '
ABS     : 3C
 '              remainder of program
 '
```

Magnetic Tape, Cassette and Disk:

For these devices, the Autoload program will load the first record (sector 0 for disc) beginning at location : 0 (or any other location selected) of memory. If the load and execute option was selected (SENSE switch set), Autoload will then enter the record loaded at its relative zero location.

This mode of operation provides a convenient method of "bootstrapping" into memory the first record of the user's system, program, etc. This record can then bring in any other segments, as required.

Becauser the Autoloader program uses locations : 30 through : 3B for scratch, these locations cannot be loaded and any prior contents of these locations will be destroyed by Autoload.

PROGRAM OPERATION

Prior to operation of the Autoload program, the user must select the device to be used and, if relocation is desired, the starting load location.

Program Loading:

Device selection requires that the value corresponding to the device and load mode be entered in the Console Sense Register. The possible values are:

Mode \ Device	TTY	HSPT	MT	Cassette	Disc
Load Abs.	: 0	: 1	: 2	: 3	: 4
Load Rel.	: 8	: 9	: A	: B	: C

If relocation was selected, the starting load location must be entered in the X register. The SENSE switch must be set if the load-and-execute mode is desired.

Depression of the AUTO switch will now initiate the Autoload program.

End Action:

The action performed at the end of the Autoload operation is dependent upon the device used.

Teletype and High Speed Paper Tape Reader:

After a successful load (a zero length record encountered without errors) control is transferred to the start location of the loaded program if:

1) A valid start address was on the tape, and
2) The SENSE switch was set.

If the SENSE switch was reset, or if no valid start address was found, Autoload will halt with : 0800 in the I register; the next location available for loading in the X register; and the start address in the A register will be negative (: FFFF) if no valid start address found.

To load another program depress the AUTO switch. Loading will continue at the next available location (X register) if relocatable mode was selected.

Magnetic Tape, Cassette and Disk:

After a successful load, control will be transferred to location zero of the program loaded if the SENSE switch was set. If the SENSE was reset, Autoload will halt with : 0800 in the I register.

Error Handling/Recovery:

The action performed, and recovery available, is dependent upon the device used.

Teletype and High Speed Paper Tape Reader:

If an invalid checksum or tape format error is detected, Autoload will halt with : 0801 in the I register. Format errors can be caused by attempting to load an object program that contains literals, external references or definitions.

In the event of an error, it is possible to backup the paper tape one record and depress AUTO to continue. This is not recommended, however, since it is possible that a record read in error could overlay other portions of the program. It is recommended that loads exhibiting errors be completely repeated.

Magnetic Tape, Cassette and Disk:

If an error occurs while attempting to load from one of these devices, the Autoload program will halt with : 0801 in the I register. To retry, the user should depress AUTO.

APPENDIX A

Autoload Operation Summary

The procedure to load programs into memory with the Autoload program is as follows:

1. Enable the Console by moving the Console Enable slide switch (located in the recess on the right side of the Console) to the enable position (up). The ENABLE indicator on the front panel will turn on.

2. If the STOP indicator is off, press the STOP switch to halt the computer. The indicator will turn on.

3. Press the RESET switch to clear the computer.

4. If relocation is desired, enter the beginning load location in the Index (X) register:

   a. If the SENSE/DATA indicator is on, press the SENSE/DATA switch to enable the Console Data register. The indicator will turn off.

   b. If the WRITE/READ indicator is off, press the WRITE/READ switch to enable register alteration. The indicator will turn on.

   c. Enter the beginning load location into the Console Data register via the hex keyboard.

   d. Press the X switch to transfer the address to the X register.

5. Enter into the Console Sense register the value (see Table A1) corresponding to the device and load mode desired:

   a. If the SENSE/DATA indicator is off, press the SENSE/DATA switch to enable the Console Sense register. The indicator will turn on.

   b. Enter the appropriate hex value from Table A1, below, into the Console Sense register via the hex keyboard.

   c. Press the SENSE/DATA switch to disable the Console Sense register. The indicator will turn off.

6. Set the SENSE switch (turn the SENSE indicator on) if load-and-execute mode desired. Otherwise reset the switch (turn indicator off).

7. Ready the load device.

8. Press the STOP switch to enable RUN mode (the STOP indicator will turn off). This enables RUN mode, but does not cause the computer to enter the RUN mode.

9. Press the RESET switch to clear the computer.

10. Press the AUTO switch to execute the Autoload program. The AUTO indicator will remain on until the load is completed.

Mode \ Device	TTY	HSPT	MT	Cassette	Disk
Load Abs.	: 0	: 1	: 2	: 3	: 4
Load Rel.	: 8	: 9	: A	: B	: C

Table A1

Load Mode Selection

BINARY LOADER
(BLD)

INTRODUCTION

The Binary Loader (BLD) is a short, "basic" loader available to load the Object Loader (LAMBDA) or other binary programs, as produced by the Binary Dump program (BDP), into memory. BLD is a "binary relocatable" program and may reside anywhere in memory. Typically, BLD is loaded into upper memory (at location : 0FB0 in 4K, : 1FB0 in 8K, etc.) using the Bootstrap Loader (BOOT).

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. Bootstrap Loader (BOOT)

PROGRAM DESCRIPTION

The Binary Loader will load all tapes produced by the Binary Dump program, and all "binary" programs produced by the BETA assemblers. A Binary program is one which contains only ORG's (REL or ABS, but no mode mixing), END or DATA statements and contains no out of range instructions, literals or external references or definitions. Additionally the RES directive may be used only in its single parameter form (i.e., no fill value appended).

Binary Tape Format

A binary tape, as produced by BDP or the BETA assembler, contains loader type codes and data or instructions grouped into records. Each record begins with a record marker (:FF) and a record length (in bytes) 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 Loader Type Codes

The binary loader recognizes 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.
DATA relocatable (code = : 07).	The type code is followed by a two byte count of the number of data words to load. This is followed by the relocatable data, packed two bytes per word.

Checksum

A rotating, Exclusive OR byte checksum is computed 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
```

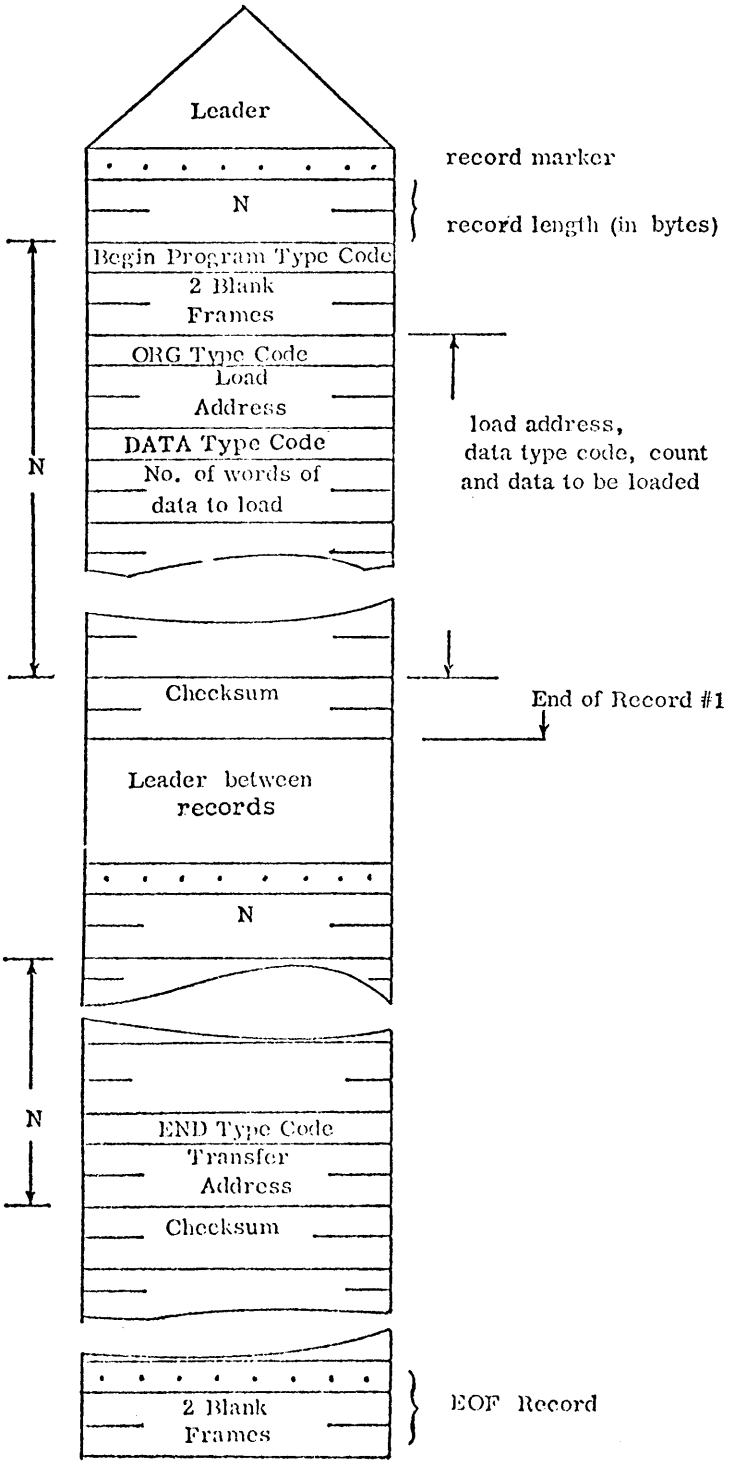

Figure 1. Binary Tape Format

PROGRAM OPERATION

Program Loading

BLD is loaded into available memory using the Bootstrap Loader (BOOT), typically directly below the Bootstrap.

Operation

Loading is initiated by entering BLD at relative location zero (first location of BLD). If the program to be loaded is to be relocated (loaded at a location other than the location punched by the Binary Punch program, BDP), the load bias desired (difference between assembled load address and load address desired) must be entered in the X register and the value : 8 entered in the console prior to execution of BLD.

BLD will accept input from either the TTY paper tape reader or from the high speed paper tape reader; whichever is ready.

End Action

Upon encountering an end-of-file, BLD will halt with : 0800 in the I register and the X register will contain the next location available for loading. The A register will be set to zero indicating a valid checksum comparison was made.

Pressing RUN will cause BLD to be re-entered at relative location zero and the next binary tape to be loaded. Refer to the operation section above.

Error Handling/Recovery

The binary loader detects invalid type code and checksum errors and halts with the error code (: 0800) in the I register.

A checksum error will cause the tape to halt on the last frame of the record with the difference between the computed checksum and that read from the tape displayed in the A register.

A type code error will cause the tape to halt within the record, two frames past the frame in question. The user should verify the tape being loaded is of a form loadable by BLD.

Appendix A

BLD OPERATION SUMMARY FOR ALPHA LSI COMPUTER

The procedure for loading programs into an ALPHA LSI computer using the Binary Loader is as follows:

1. Load BLD into memory using the Bootstrap procedure (location :0FB0 for 4K, :1FB0 for 8K, etc.).

2. If the STOP indicator is off, press the STOP switch to halt the computer. The indicator will turn on.

3. Press the Console RESET to clear the computer.

4. If the SENSE/DATA indicator is on, press the SENSE/DATA switch to enable the Console Data register. The indicator will turn off.

5. Enter into the Program Counter (P) register the starting location of BLD (:nFB0).

   a. If the WRITE/READ indicator is off, press the WRITE/READ switch to enable register alteration. The indicator will turn on.

   b. Enter the address into the Console Data register via the hex keyboard.

   c. Depress the P switch to transfer the address to the P register.

6. If the program is not to be relocated, proceed to step 9.

7. Enter the relocation bias into the Index (X) register.

   a. If the WRITE/READ indicator is off, press the WRITE/READ switch to enable register alteration. The indicator will turn on.

   b. Enter the address into the Console Data register via the hex keyboard.

   c. Depress the X switch to transfer the address to the X register.

8. Enter the relocation indicator (: 8) into the Console Sense register.

   a. Press the SENSE/DATA switch to enable the Console Sense register. The indicator will go on.

   b. Enter : 8 into the Console Sense register via the hex keyboard.

   c. Press the SENSE/DATA switch to disable the Console Sense register. The indicator will go off.

9. Place the binary tape in the selected reader (HSR or TTY) with the leader over the read station.

10. 'Ready' the selected reader. Be sure to check that both the high speed reader and the teletype reader are not both 'ready' (one or the other but not both).

11. Press the STOP switch to enable RUN mode. The STOP indicator will turn off. This enables the RUN mode, but does not cause the computer to enter RUN mode.

12. Press the RESET switch to clear the computer.

13. Depress the RUN switch. The program will be loaded into memory and the processor will halt with : 0800 in the I register indicating a successful load (: 0880 if an illegal type code was encounted or a checksum error occured). To display the I register:

   a. Press the STOP switch to disable RUN mode. The STOP indicator will turn on.

   b. If the WRITE/READ indicator is on, press the WRITE/READ switch to enable register display. The indicator will turn off.

   c. Depress the I switch to display the contents of the I register in the Console Data register.

14. The X register will contain the next sequential memory location available for loading. To load another binary tape at this location, repeat this procedure from step 8. Depress the X switch to display the contents of the X register in the Console Data register.

15. To load another binary tape at some location other than the next one in sequence, repeat this procedure from step 6.

Appendix B

BLD OPERATION SUMMARY FOR ALPHA 16 COMPUTER

The procedure for loading programs into an ALPHA 16 computer using the Binary Loader is as follows:

1. Load BLD into memory using the Bootstrap procedure (location :0FB0 for 4K, :1FB0 for 8K, etc.).

2. Depress the STOP switch. Check that the MANual EXecute switch is not depressed.

3. Depress the Console RESET to clear the computer.

4. Enter into the Program Counter (P) register the starting location of BLD (:nFB0).

   a. Raise all Register Select switches.

   b. Depress the P Register Select switch.

   c. Enter the starting address of BLD on the Data Entry switches.

   d. Depress the ENTRY switch.

5. If the program is not to be relocated, proceed to step 8.

6. Enter the relocation bias into the Index (X) register.

   a. Depress the X Register Select switch.

   b. Enter the load address on the Data Entry switches

   c. Depress the ENTRY switch.

7. Enter the relocation indicator (Data Switch 3) into the Data Switches.

   a. Raise all Data Switches.

   b. Depress Data Switch 3.

8. Raise the STOP switch.

9. Place the binary tape in the selected reader (HSR or TTY) with the leader over the read station.

10. 'Ready' the selected reader. Be sure to check that both the high speed reader and the teletype reader are not both 'ready' (one or the other but not both).

11. Depress the RUN switch. The program will be loaded into memory and the processor will halt with : 0800 in the I register indicating a successful load (: 0880 if an illegal type code was encountered or a checksum error occured). To display the I register:

    a. Raise all Register Select switches.

    b. Depress the I Register Select switch.

12. The X register will contain the next sequential memory location available for loading. To load another binary tape at this location, repeat this procedure from step 7. To display the X register, depress the X register select switch.

13. To load another binary tape at some location other than the next one in sequence, repeat this procedure from step 6.

```
0002                    *
0003                    * ALPHA-16/ALPHA-LSI BINARY LOADER (BLD) 96002
0004                    * COPYRIGHT 1973 BY COMPUTER AUTOMATION, INC.
0005                    * THIS PROGRAM IS BINARY RELOCATABLE
0006                    * STANDARD LOAD LOCATION = :NFB0
0007                    *
0008          0006      HSRDA   EQU   6
0009          0007      TTYDA   EQU   7
0010                    *
0011  0000                      REL   0
0012          0000      BLD     EQU   $
0013  0000  EA43                STX   SLOC      SAVE LOAD LOCATION
0014  0001  5801                ISA             CHECK FOR RELOCATION.
0015  0002  3101                JAN   $+2       . REL
0016  0003  9A40                STA   SLOC      . ABS, RESET LOAD LOC TO 0
0017  0004  FA28      BLD1      JST   INPB      GET NEXT CHAR (1 FRAME)
0018  0005  AA3D                XOR   HFF       TEST FOR MARKER.
0019  0006  3142                JAN   BLD1      . NO, KEEP CHECKING.
0020  0007  9A3E                STA   CKSUM     ZERO CHECKSUM
0021  0008  FA1F                JST   INPW      GET RECORD LENGTH (2 FRAMES)
0022  0009  3105                JAN   TYPE
0023  000A  E23A                LDX   PC        . 0 = NEXT AVAILABLE ADDR
0024  000B  0800                STOP  0         DISPLAY I=:0800, X=NEXT LOC
0025  000C  F60C                JMP   BLD       LOAD SOME MORE
0026                    *
0027          000D      ORG     EQU   $         PROCESS ORG
0028  000D  8A36                ADD   SLOC      ADD OFFSET
0029  000E  9A36      A         STA   PC        MODIFY STORAGE ADDRESS
0030                    *
0031          000F      TYPE    EQU   $         DECODE TYPE CODE
0032  000F  B236                LDA   CKSUM     SAVE CURRENT CHECKSUM
0033  0010  9A36                STA   TEMP
0034  0011  FA1B                JST   INPB      GET TYPE CODE
0035  0012  212C                JAZ   CKSM      . TC=0, CHECKSUM
0036  0013  0048                TAX             SAVE TC
0037  0014  FA13                JST   INPW      GET DATA WORD
0038  0015  C107                CXI   7
0039  0016  F207                JMP   DATA      . TC=7, REL DATA
0040  0017  C105                CXI   5
0041  0018  F60B                JMP   ORG
0042  0019  13A9                LRX   2
0043  001A  284B                JXZ   TYPE      . TC=1,2,3
0044  001B  00A8                DXR             . TC=4,5,6,7
0045  001C  3825                JXN   TCERR     . TC ERROR
0046  001D  324F                JOR   A         . TC=4
0047                    *
0048          001E      DATA    EQU   $         PROCESS ABS DATA
0049  001E  0310                NAR             . A= WORD COUNT
0050  001F  9A27                STA   CNT
0051  0020  FA07      WORD      JST   INPW      GET DATA WORD
0052  0021  C107                CXI   7
0053  0022  8A21                ADD   SLOC      ADD RELOCATION
```

```
0054  0023  9B21          STA  *PC       STORE IN MEMORY
0055  0024  DA20          IMS  PC        BUMP MEMORY POINTER
0056  0025  DA21          IMS  CNT       BUMP COUNT
0057  0026  F006          JMP  WORD      . AGAIN
0058  0027  F018          JMP  TYPE      . DONE
0059                *
0060  0028  0800  INPW    ENT            INPUT WORD TO A REGISTER
0061  0029  FA03          JST  INPB      GET 1ST BYTE
0062  002A  1357          LLA  8         . SHIFT
0063  002B  FA01          JST  INPB      ADD 2ND BYTE
0064  002C  F704          RTN  INPW      EXIT
0065                *
0066  002D  0800  INPB    ENT            INPUT BYTE TO A REGISTER
0067  002E  4032  INPB1   SEL  HSRDA,2    STEP HSR
0068  002F  403A          SEL  TTYDA,2   STEP TTY
0069  0030  4931  HSR     SEN  HSRDA,1   HSR READY ?
0070  0031  F202          JMP  TTY       . NO
0071  0032  7831          IBA  HSRDA,1   . READ
0072  0033  F203          JMP  CK
0073  0034  4939  TTY     SEN  TTYDA,1   TTY READY ?
0074  0035  F605          JMP  HSR       . NO
0075  0036  7839  TTY1    IBA  TTYDA,1   . READ
0076  0037  BA0E  CK      EMA  CKSUM     COMPUTE CHECKSUM, SAVE BYTE
0077  0038  11D0          RRA  1
0078  0039  3201          JOR  $+2
0079  003A  8A07          ADD  H80       . TURN BIT 7 ON
0080  003B  AA0A          XOR  CKSUM
0081  003C  8206          AND  HFF       . STRIP TO 8 BITS
0082  003D  BA08          EMA  CKSUM     . SWAP BACK
0083  003E  F711          RTN  INPB      EXIT
0084                *
0085        003F  CKSM    EQU  $         VERIFY CHECKSUM
0086  003F  FE12          JST  INPB      GET FROM TAPE
0087  0040  9206          SUB  TEMP      COMPARE TO CALCULATED
0088  0041  217D          JAZ  BLD1      . OK
0089        0042  TCERR   EQU  $         TC OR CHECKSUM ERROR
0090  0042  0880  H80     STOP :80       . DISPLAY I=:0880
0091                *
0092  0043  00FF  HFF     DATA :FF
0093  0044  1800  SLOC    HLT            START LOAD LOC (BASE)
0094  0045  0800  PC      HLT            CURRENT LOC (WORKING)
0095  0046  0800  CKSUM   HLT            CHECKSUM STORAGE
0096        0047  CNT     EQU  $
0097  0047  0000  TEMP    DATA 0         CHECKSUM STORAGE
0098                *
0099        0000          END  BLD
0000 ERRORS
```

```
0029  A          0046
0012  BLD        0025  0099
0017  BLD1       0019  0088
0076  CK         0072
0085  CKSM       0035
0095  CKSUM      0020* 0032  0076* 0080  0082*
0096  CNT        0050* 0056*
0048  DATA       0039
0090  H80        0079
0092  HFF        0018  0081
0069  HSR        0074
0008  HSRDA      0067  0069  0071
0066  INPB       0017* 0034* 0061* 0063* 0083  0086*
0067  INPB1
0060  INPW       0021* 0037* 0051* 0064
0027  ORG        0041
0094  PC         0023  0029* 0054* 0055*
0093  SLOC       0013* 0016* 0028  0053
0089  TCERR      0045
0097  TEMP       0033* 0087
0073  TTY        0070
0075  TTY1
0009  TTYDA      0068  0073  0075
0031  TYPE       0022  0043  0058
0051  WORD       0057
0099 SOURCE LINES
```

REQUEST FOR PROGRAMMING ACTION (RPA)

PROGRAM NAME	PROGRAM ID #	ORIGINATOR	DATE	RPA#
ALPHA 16/LSI LAMBDA Object Loader	96003-00E0		11/8/73	70

DESCRIPTION OF PROBLEM

On ALPHA LSI Computers the console RESET button is intended for use as a master clear and system initialization. It should only be used to clear the system prior to operation or to abort and reinitialize a runaway program or peripheral. It is not proper to set up register contents and then press RESET just prior to RUN.

PROBLEM VERIFIED [X] YES [ ] NO BY:
COMMENTS DATE:

Documentary Only: Under "LAMBDA Operation Summary for ALPHA LSI Computers", page 3C-10, delete step 12.

TEMPORARY PATCH AVAILABLE [ ] YES [X] NO

LOCATION	OLD CONTENTS	NEW CONTENTS

UPDATED PROGRAM RELEASE

SCHEDULED DATE _____ NEW PROGRAM ID _____

APPROVED BY

MS

CAI FORM # _____

LAMBDA

OBJECT LOADER

INTRODUCTION

The LAMBDA loader loads into memory both the object program tapes produced by the BETA Assembly programs and the binary tapes produced by the Binary Dump program (BDP).

ENVIRONMENT

The LAMBDA loader is designed to reside in upper memory and will create an external name table, below the loader, expanding toward lower memory (see figure 1). The loader will accept input from either the ASR-33 paper tape reader or a high speed paper tape reader. The optional load map will be produced on the ASR-33 printer or line printer.

Hardware Required:

ALPHA LSI or ALPHA 16 series computers
ASR-33 or 35 Teletype or High Speed Paper Tape Reader

Software Required:

Binary Loader (BLD) or Autoload

FEATURES

1. Loads either absolute or relocatable object or binary format programs.

2. Maintains, in the base page, a common literal pool insuring that equal valued literals are stored only once.

3. Maintains a table of external reference names and links all of the references as they are encountered. Whenever an external definition is found, the external references are unthreaded to reference the load location of the external name.

4. Satisfies external linkages by generating link addresses in either the base page (see assembler EXTR directive) or in main memory (see assembler REF directive). If EXTR link is requested and if the external definition is already defined and in the base page, a direct link is made.

5. Selective loading from a library tape results in loading only those (unloaded) programs or subprograms referenced by a previously loaded program. User option to override selective loading and unconditionally load all programs read (forced load) is provided.

6. The LAMBDA loader is a 'binary relocatable' program and can reside any place in memory.

7. A load map may be optionally generated on the ASR-33 printer or line printer.

PROGRAM DESCRIPTION

The LAMBDA loader will load (and optionally relocate) object programs produced by the BETA assemblers or binary programs produced by the Binary Dump Program (BDP). The capability of using a pre-assembled subroutine library is included in a manner which guarantees that required routines will be loaded precisely once. The first program presented to the loader (usually the main program) will be unconditionally loaded. The loading will terminate when an end-of-file is detected on the object tape (see figure 2). At that time, a load map is optionally output and the computer is halted. If additional program modules are required, another object tape (or library tape) may be mounted in the reader and depressing RUN will cause loading to continue to the next end-of-file.

As the loader is designed to 'slough' through an input stream, selecting and loading only those (unloaded) subprograms referenced by a previously loaded program, close attention should be paid to physical subprogram ordering, particularly in the library tape. In the example below, A represents a main program while B, C, D, and E are subprograms. Note that there are four orderings which will result in a one-pass loading; any other ordering will require multiple load passes:

A	calls	C and E
B	calls	D
E	calls	B

correct ordering of library:

A	A	A	A
C	E	E	E
E	C	B	B
B	B	C	D
D	D	D	C

In the event that ordering problems occur, they may be simply resolved by duplication of the offending programs in the library.

LAMBDA also provides an unconditional load (force load) mode of operation should selective loading be undesirable. The user may also specify whether or not the defined and undefined symbols are to be listed on the teletype or line printer (see figure 4).

Object Tape Format

An object tape, as produced by the BETA assemblers, contains loader type codes and data or instructions grouped into records. Each record begins with a record marker (:FF) and a record length (in bytes) and is terminated with a checksum (figure 3).

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

Object Loader Type Codes

The object loader recognizes 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.
DATA relocatable (code = : 07)	The type code is followed by a two byte count of the number of data words to load. This is followed by the relocatable data, packed two bytes per word.
RES and store constant (code = : 08)	The type code is followed by a two byte count of the number of words to reserve and a two byte constant to be stored in these words.
BAC relocatable (code : 09)	The type code is followed by a two byte address constant.
MREF absolute literal (code = : 0A)	The type code is followed by a two byte literal and one byte containing the most significant 8 bits of the memory reference instruction.
MREF relocatable literal (code = : 0B)	The type code is followed by a two byte literal and one byte containing the most significant 8 bits of the memory reference instruction.
MREF external (code = : 0C)	The type code is followed by the external name (6 bytes) and one byte containing the most significant 8 bits of memory reference instruction.
External Definition (code = : 0D)	The type code is followed by the external name (6 bytes). The load location counter is used as the definition address.
REF (code = : 0E)	The type code is followed by the external name (6 bytes).

NAME (code = : 0F)	The type code is followed by the external name (6 bytes). When used, it is the first type code on the object tape after the record marker and Begin Program type code.
MREF Byte relocatable (code = : 10)	The type code is followed by a two byte address and one byte containing the most significant 8 bits of the memory reference instruction.

Checksum

A rotating, Exclusive OR byte checksum is computed 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 schecksum
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
```

SENSE switch set: No load; list undefined names.

SENSE switch reset:

Print Symbols / Load Mode	Defined and Undefined		Defined Only		Undefined Only		Neither Defined or Undefined
	TTY	LP*	TTY	LP*	TTY	LP*	
Library	: 0	: 1	: 2	: 3	: 4	: 5	: 6
Unconditional	: 8	: 9	: A	: B	: C	: D	: E

*Centronics Line Printer. Operation with Data Products Line Printer requires the word NOP: ME in LAMBDA to be NOP'ed prior to operation.

Figure 4. Load Options

Load Map

Each time the loader encounters an end-of-file in the input stream, a load map is output and the processor halts. The load map consists of three types of outputs:

1. Optional load map indicating names and load locations for external definitions (see figure 4).

2. Optional map of all unsatisfied external references (see figure 4).

3. Load summary line.

Sample load map:

```
START          0100
SI             0180
LOC            0222
FRAME          U
PUNCH          U
SI             M
E  00FA  0475  0100
```

External entry points START, SI, and LOC were loaded at the addressed shown. External names FRAME and PUNCH were referenced but not loaded yet.

External name SI appeared in 2 program modules. The loader flags the second appearance as multiply defined (M). The first module containing SI was loaded and linked as necessary. The second module was not loaded.

: 00FA is the next base page location that will be assigned by the loader (: FB - : FD have already been assigned).

: 0475 is the highest main memory location loaded +1. In the absence of operator intervention, this will be the next location loaded by LAMBDA.

: 0100 is the transfer address that appeared on the END statement in the last program loaded. If a transfer address did not appear on the END statement, nothing will be shown here (the absence of a transfer address on the summary line indicates no transfer address).

PROGRAM OPERATION

Program Loading

LAMBDA is loaded into available memory using the Binary Loader (BLD). The loader will load behind itself; however, no check is made for loading into non-existing memory. For this reason, the loader should reside in high memory.

Operation

Loading is initiated by entering LAMBDA at relative location zero (first location of LAMBDA). If the program to be loaded is to be relocated from its assembled origin address, the relocation bias (offset from assembled origin address) must be preset in the A register prior to entering the loader.

The loader will normally start assigning base page locations, as needed, starting at location :FD and progressing towards location zero. If :FD is not to be used as the initial base page location, the user may preset any other base page location in the X register prior to entering the loader (X preset to 0 will select the default value of :FD).

The loader will accept input from either the TTY paper tape reader or the high speed paper tape reader (whichever device is loaded and ready).

Reset the SENSE switch and enter the value corresponding to the options selected (see figure 4) into the Console Sense register (low order Data switches on ALPHA-16).

Depress RUN to initiate loading.

At end-of-load (EOF read) the names still undefined may be output by depressing the SENSE switch and RUN switch.

End Action

Upon encountering an end-of-file, the loader will optionally output the load map and halt. The following actions will be taken if the operator presses RUN:

1. If forced loading option is exercised, the next program or subprogram will be loaded.

2. If the SENSE switch is pressed after loading is completed, a list of undefined (unsatisfied) references will be printed on the selected device.

3. If all external references have been satisfied and a transfer address appeared on an END directive, control will transfer to the transfer address.

4. If a transfer address did not appear on an END directive, the summary line will be output again.

5. If selective loading has been selected and there are still external references to be satisfied, the loader will read the next program(s) from the selected reader.

Error Handling/Recovery

Detection of an error condition by the loader will cause output of an appropriate error message followed by a computer halt. The error message will contain a one-character error code followed by the last location loaded.

The following errors are detected by the LAMBDA loader:

C — Checksum error. Loading may be continued by rereading the offending record. It is not advisable to do so.

T — Type code error. An illegal type code has been encountered on the input tape.

L — Load location overflow. Program storage requirements have exceeded available memory.

S — Page zero overflow. Program linkage requirements have exceeded base page capacity.

Appendix A

LAMBDA OPERATION SUMMARY FOR ALPHA LSI COMPUTER

The procedure for loading programs into an ALPHA LSI computer using the Object Loader is as follows:

1. Load LAMBDA into high memory using the Binary Loader (BLD) or Autoload.

2. If the STOP indicator is off, press the STOP switch to halt the computer. The indicator will turn on.

3. Press the Console RESET to clear the computer.

4. If the SENSE/DATA indicator is on, press the SENSE/DATA switch to enable the Console Data register. The indicator will turn off.

5. Enter the relocation bias (if relocation desired) into the A register (otherwise set A to zero).

   a. If the WRITE/READ indicator is off, press the WRITE/READ switch to enable register alteration. The indicator will turn on.

   b. Enter the address (or zero) into the Console Data register via the hex keyboard.

   c. Press the A switch to transfer the address to the A register.

6. Enter the initial base page location (if :FD is not desired) into the Index (X) register (otherwise set X to zero).

   a. If the WRITE/READ indicator is off, press the WRITE/READ switch to enable register alteration. The indicator will turn on.

   b. Enter the address (or zero) into the Console Data register via the hex keyboard.

   c. Press the X switch to transfer the address to the X register.

7. Enter into the Program Counter (P) register the starting location of LAMBDA.

   a. If the WRITE/READ indicator is off, press the WRITE/READ switch to enable register alteration. The indicator will turn on.

   b. Enter the address into the Console Data register via the hex keyboard.

   c. Press the P switch to transfer the address to the P register.

8. Enter the value corresponding to the mode of loading and output device desired (see figure 4) into the Console Sense register.

   a. Press the SENSE/DATA switch to enable the Console Sense register. The indicator will go on.

   b. Enter the applicable value into the Console Sense register via the hex keyboard.

   c. Press the SENSE/DATA switch to disable the Console Sense register. The indicator will go off.

9. Place the object tape in the selected reader (HSR or TTY) with the leader over the read station.

10. 'Ready' the selected loader. Be sure to check that both the high speed reader and the teletype reader are not both 'ready' (one or the other but not both).

11. Press the STOP switch to enable RUN mode. The STOP indicator will turn off. This enables the RUN mode, but does not cause the computer to enter RUN mode.

12. Press the RESET switch to clear the computer.

13. Press the RUN switch to initiate loading. The program will be loaded into memory and the processor will halt with : 0800 in the I register indicating a successful load (: 0801 if an error occurred). To display the I register:

   a. Press the STOP switch to disable RUN mode. The STOP indicator will turn on.

b. If the WRITE/READ indicator is on, press the WRITE/READ switch to enable register display. The indicator will turn off.

c. Press the I switch to display the contents of the I register in the Console Data register.

14. At end of load examine the load map. If unsatisfied references have been listed, mount appropriate library tape(s) and depress RUN.

15. Repeat step 14 until all unsatisfied references have been loaded.

16. If a transfer address appeared on the END statement and the SENSE Switch and Register Display Indicator 3 is reset, program execution will commence when RUN is depressed. To display the Register Display Indicator:

a. Press the SENSE/DATA switch to enable the Console Sense Register. The indicator will go on.

b. Examine Register Display Indicator 3.

Appendix B

LAMBDA OPERATION SUMMARY FOR ALPHA 16 COMPUTER

The procedure for loading programs into an ALPHA 16 computer using the Object Loader is as follows:

1. Load LAMBDA into memory using the Binary Loader (BLD) or Autoload.

2. Press the STOP switch.

3. Press the Console RESET to clear the computer.

4. Enter the relocation bias (if relocation desired) into the A register (otherwise set A to zero).

   a. Press the A Register Select switch.

   b. Enter the load address on the Data Entry switches.

   c. Press the ENTRY switch.

5. Enter the initial base page location (if :FD is not desired) into the Index (X) register (otherwise set X to zero).

   a. Raise all Register Select switches.

   b. Press the X Register Select switch.

   c. Enter the load address on the Data Entry switches.

   d. Press the ENTRY switch.

6. Enter into the Program Counter (P) register the starting location of LAMBDA.

   a. Raise all Register Select switches.

   b. Press the P Register Select switch.

   c. Enter the starting addres of LAMBDA on the Data Entry switches.

   d. Press the ENTRY switch.

7. Enter the value corresponding to the mode of loading and output device desired (see figure 4) into the low order 4 Data Switches.

   a. Raise all Data Switches.

   b. Press the appropriate Data Switches.

8. Raise the STOP switch.

9. Place the object tape in the selected reader (HSR or TTY) with the leader over the read station.

10. 'Ready' the selected reader. Be sure to check that both the high speed reader and the teletype reader are not both 'ready' (one or the other but not both).

11. Press the RUN switch to initiate loading. The program will be loaded into memory and the processor will halt with : 0800 in the I register indicating a successful load (: 0801 if an error occured). To display the I register:

    a. Raise all Register Select switches.

    b. Press the I Register Select switch.

12. At end of load examine the load map. If unsatisfied references have been listed, mount appropriate library tape(s) and press RUN.

13. Repeat step 12 until all unsatisfied references have been loaded.

14. If a transfer address appeared on the END statement and the SENSE Switch and Data Switch 3 are reset, program execution will commence when RUN is pressed.

```
PAGE  0001  10/29/73  14:04:31  LAMBDA OBJECT LOADER 96003-10E2

0002                  * ALPHA LSI/ALPHA 16 OBJECT LOADER (LAMBDA) 96003
0003                  * COPYRIGHT 1973 BY COMPUTER AUTOMATION, INC.
0004                  * THIS PROGRAM IS BINARY RELOCATABLE
0005                  *
0006          0004    LPDA    EQU  4          STANDARD LP DEVICE ADDRESS
0007          0007    TDA     EQU  7          STANDARD TTY DEVICE ADDRESS
0008                  *
0009                  * THE LAMBDA OBJECT LOADER LOADS OBJECT
0010                  * PROGRAMS INTO MEMORY, SATISFIES LINKAGES AND
0011                  * ASSIGNS LITERALS. THE SYMBOL DEFINITION TABLE
0012                  * STARTS AT THE BEGINNING OF THE LOADER AND
0013                  * WORKS DOWNWARDS. THE LITERALS ARE ASSIGNED
0014                  * STORAGE AT :FD AND DOWNWARDS.
0015                  *
0016                  *
0017                  * PROGRAM ENTRY POINT
0018  0AE0                  RFL  :AE0       4K LOAD LOCATION
0019  0AE0  FAB9    BEGG    JST  EXT        SETUP REF/DEF TABLE ADDR
0020                  * TYPE BRANCH
0021  0AE1  0011            DATA T1-BEGG    START OF RECORD
0022  0AE2  0029            DATA T2-BEGG    END, ABSOLUTE
0023  0AE3  0027            DATA T3-BEGG    END, RELATIVE
0024  0AE4  0056            DATA T4-BEGG    ORG, ABSOLUTE
0025  0AE5  0055            DATA T5-BEGG    ORG, RELOCATABLE
0026  0AE6  0059            DATA T6-BEGG    DATA, ABSULUTE
0027  0AE7  0058            DATA T7-BEGG    DATA, RELOCATABLE
0028  0AE8  0036            DATA T8-BEGG    RES
0029  0AE9  0064            DATA T9-BEGG    BAC, RELOCATABLE
0030  0AEA  006A            DATA TA-BEGG    MREF, ABSOLUTE
0031  0AEB  0069            DATA TB-BEGG    MREF, RELOCATABLE
0032  0AEC  0071            DATA TC-BEGG    MREF, EXTERNAL
0033  0AED  0088            DATA TD-BEGG    EXT
0034  0AEE  008C            DATA TE-BEGG    REF
0035  0AEF  0094            DATA TF-BEGG    NAME
0036  0AF0  0068            DATA T10-BEGG MREF, BYTE RELOCATABLE
0037                  * BEGINNING OF RECORD
0038  0AF1  3121    T1      JAN  ERRT       TYPE ERROR
0039  0AF2  F2C3            JMP  START
0040                  * THIS ROUTINE STORES A AT THE NEXT BASE PAGE
0041                  * LOCATION AND RETURNS THE BASE PAGE LOCATION
0042                  * IN A THE BASE PAGE LOCATION IS INCREMENTED
0043                  * AND TESTED FOR OVERFLOW
0044  0AF3  0800    SSLOC   ENT
0045  0AF4  EADF            STX  TEMP3
0046  0AF5  E2DA            LDX  SLOC
0047  0AF6  281E            JXZ  ERRS1
0048  0AF7  0128    SSLOCA  IXR
0049  0AF8  C100    BPT     CXI  0          THIS INSTRUCTION IS BUILT.
0050  0AF9  F206            JMP  SSLOCB
0051  0AFA  2105            JAZ  SSLOCB     ASSIGN VALUE
0052  0AFB  D400            CMS  *0         VALUE ALREADY THERE?
0053  0AFC  0000            NOP
```

```
0054  0AFD  F606          JMP  SSLOCA   NO
0055  0AFE  003W          TXA           YES
0056  0AFF  F205          JMP  SSLOCC
0057  0B00  9BCF  SSLOCB  STA  *SLOC    STORE IN SCRATCH PAD
0058  0B01  B2CE          LDA  SLOC
0059  0B02  00D0          DAR           NEXT STORE LOCATION
0060  0B03  9ACC          STA  SLOC
0061  0B04  0150          IAR           RETURN WITH A=WHERE STORED
0062  0B05  E2CE  SSLOCC  LDX  TEMP3
0063  0B06  F713          RTN  SSLOC
0064                *  END, RELOCATABLE
0065  0B07  2083  T3      JAM  T21      CONTINUE IF OVERFLOW
0066  0B08  8AC5          ADD  RBASE    OTHERWISE RELOCATE
0067                *  END, ABSOLUTE
0068  0B09  2081  T2      JAM  T21      CONTINUE IF OVERFLOW
0069  0B0A  9AC6          STA  ELOC     SAVE ADDR IN EXECUTE LOC.
0070  0B0B  B2BF  T21     LDA  HIGH     UPDATE LOAD LOCATION
0071  0B0C  D2C2          CMS  LLOC     SAVE HIGHEST STORE LOCATION
0072  0B0D  B2C1          LDA  LLOC
0073  0B0E  0000          NOP
0074  0B0F  9ABA          STA  HIGH
0075  0B10  9ABD          STA  RBASE
0076  0B11  9ABD          STA  LLOC
0077  0B12  F2A3  ISTART  JMP  START
0078  0B13  C6D4  ERRT    LAP  'T'      TYPE ERROR
0079  0B14  F2CF          JMP  ERR
0080  0B15  F2CD  ERRS1   JMP  ERRS
0081                *  RES
0082  0B16  0210          CAR           SAVE WORD COUNT
0083  0B17  9ABD          STA  TRY
0084  0B18  FAAA          JST  GNW      GET CONSTANT
0085  0B19  DABB  T8A     IMS  TRY      TEST FOR DONE
0086  0B1A  F201          JMP  $+2
0087  0B1B  F29A          JMP  START
0088  0B1C  FAED          JST  SLLOC    AND STORE IN LOAD LOCATION
0089  0B1D  003W          TXA
0090  0B1E  F605          JMP  T8A      REPEAT IF NOT DONE
0091                *  END OF FILE
0092  0B1F  9AAC  T0      STA  FLOAD    NO FORCE LOADING
0093  0B20  C602          LAP  2        LIST UNDEFINES?
0094  0B21  5C01          INAM 1        READ DATA SWITCHES
0095  0B22  3101          JAN  $+2
0096  0B23  FAF6  T0B     JST  PUND     UNDEFINE SYMBOL PRINT
0097  0B24  FAF7  T0A     JST  PEXEC    PRINT EXECUTE INFORMATION
0098  0B25  001W          ARM
0099  0B26  9AB4          STA  NRD
0100  0B27  B2A2          LDA  HIGH     GET NEXT LOAD LOCATION
0101  0B28  0800  HALT    STOP 0        LOADER HALT LOCATION
0102  0B29  9AA4          STA  RBASE    SET LOAD AND RELATIVE BIAS
0103  0B2A  9AA4          STA  LLOC
0104  0B2B  3448          JSS  T0B      LIST UNDEFINES?
0105  0B2C  5801          INA  1        READ FROM DATA SWITCHES
```

```
0106  0B2D  13D3          LRA  4          FORCE LOAD?
0107  0B2E  3201          JOR  $+2
0108  0B2F  F282          JMP  FL         YES
0109  0B30  B2A9          LDA  UND        TEST NO. OF UNDEF SYMBOLS
0110  0B31  315F          JAN  ISTART     IF A NOT ZERO, CONTINUE
0111  0B32  B29F          LDA  ELOC       LOADING OTHERWISE TEST
0112  0B33  20CF          JAM  10A        EXECUTION ADDRESS. IF NONE
0113  0B34  F39C          JMP  *ELOC      THEN HALT OTHERWISE EXECUTE
0114                  * ORG, RELOCATABLE
0115  0B35  8A98  T5      ADD  RBASE      RELOCATE ADDRESS
0116                  * ORG, ABSOLUTE
0117  0B36  9A98  T4      STA  LLOC       PUT ADDRESS IN LOAD LOC.
0118  0B37  F27E          JMP  START
0119                  * DATA, RELOCATABLE
0120  0B38  0108  T7      ZXR             SET RELOCATABLE FLAG
0121                  * DATA, ABSOLUTE
0122  0B39  0310  T6      NAR             SETUP DATA REPEAT COUNT
0123  0B3A  EA9B          STX  SPELL      SAVE ADD BIAS FLAG
0124  0B3B  9A99          STA  TRY
0125  0B3C  FA86  T6A     JST  GNW        GET NEXT DATA WORD
0126  0B3D  E298          LDX  SPELL      RESET FLAG
0127  0B3E  3801          JXN  $+2        IF NOT RELOC.,CONTINUE
0128  0B3F  8A8F          ADD  RBASE      OTHERWISE RELOCATE DATA
0129  0B40  FAC9          JST  SLLOC      AND PUT IN LOAD LOCATION
0130  0B41  DA93          IMS  TRY        DONE?
0131  0B42  F606          JMP  T6A        NO - REPEAT
0132  0B43  F272          JMP  START      YES
0133                  * BAC
0134  0B44  8A89  T9      ADD  RBASE      RELOCATE THE BYTE ADDRESS
0135  0B45  8A88          ADD  RBASE      AND RELOCATE IT
0136  0B46  FAC3          JST  SLLOC      BAC TO LOAD LOCATION
0137  0B47  F26F          JMP  START
0138                  * MREF, BYTE RELOCATABLE
0139  0B48  8A85  T10     ADD  RBASE      RELOCATION BIAS FOR BYTE
0140                  * MREF, RELOCATABLE
0141  0B49  8A84  TB      ADD  RBASE      RELOCATION BIAS
0142                  * MREF, ABSOLUTE
0143  0B4A  FE57  TA      JST  SSLOC      ADDR OF THE NEXT BASE PAGE
0144  0B4B  9A86  TAA     STA  TEMP       LOCATION. SAVE BASE PAGE
0145  0B4C  FA9A          JST  GNB        ADDRESS AND GET INST. BYTE
0146  0B4D  1357  TAAA    LLA  8          SHIFT INSTRUCTION PART
0147  0B4E  A283          IOR  TEMP
0148  0B4F  FABA          JST  SLLOC      PUT INST AND BASE PAGE ADDR
0149  0B50  F265          JMP  START      IN NEXT LOAD LOCATION
0150                  * MREF, EXTERNAL
0151  0B51  FAF9  TC      JST  RDS        READ SYMBOL AND FIND
0152  0B52  FAE2          JST  FSYM       IT IN EXTERNAL DEF TABLE
0153  0B53  B401          LDA  #1
0154  0B54  8287          AND  H7FFF
0155  0B55  3111          JAN  TCA        CONTINUE IF BASE PAGE
0156  0B56  B400          LDA  #0         GET MEMORY LOCATION FLAG
0157  0B57  208A          JAM  TCC        DEFINED?
```

```
0158  0B58  9A79          STA   TEMP      SAVE LOCATION
0159  0B59  13D7          LRA   8         0-:FF?
0160  0B5A  3184          JAG   TCD
0161  0B5B  FA8B          JST   GNB       GET INSTRUCTION PART
0162  0B5C  13D0          LRA   1         DROP INDIRRECT BIT
0163  0B5D  1350          LLA   1
0164  0B5E  F611          JMP   TAAA      PROCESS
0165  0B5F  B400  TCD     LDA   #0        CONTINUE WITH BASE PAGE
0166  0B60  FE6D          JST   SSLOC     YES
0167  0B61  F203          JMP   TCB       CONTINUE
0168  0B62  0110  TCC     ZAR             SET STORE FLAG
0169  0B63  FE70          JST   SSLOC     OTHERWISE, ASSIGN IT
0170  0B64  A278          IOR   H8000
0171  0B65  9C01  TCB     STA   #1        SET SYMBOL BASE PAGE VALUE
0172  0B66  8275          AND   H7FFF
0173  0B67  F61C  TCA     JMP   TAA       CONTINUE WITH THE BASE
0174                *                     PAGE ADDRESS IN THE A REG.
0175                * EXT
0176  0B68  FAE2  TD      JST   RDS       GET SYMBOL AND ASSIGN
0177  0B69  B265          LDA   LLOC      LOAD LOCATION TO SYMBOL
0178  0B6A  FAFC          JST   DSYM      IN EXTERNAL DEF TABLE
0179  0B6B  F24A          JMP   START
0180                * REF
0181  0B6C  FADF  TE      JST   RDS       READ SYMBOL
0182  0B6D  FAC7          JST   FSYM      FIND IN EXTERNAL DEF TABLE
0183  0B6E  3083          JAP   TEA       CONTINUE IF SYMBOL THERE
0184  0B6F  B25F          LDA   LLOC      OTHERWISE THREAD IT
0185  0B70  A26C          IOR   H8000
0186  0B71  BC00          EMA   #0
0187  0B72  FA97  TEA     JST   SLLOC     SAVE THREAD OR SYMBOL DEF
0188  0B73  F242          JMP   START
0189                * NAME
0190  0B74  FAD6  TF      JST   RDS       GET SYMBOL SPELLING
0191  0B75  FADC          JST   SEXT      AND FIND NAME IN TABLE
0192  0B76  F21D          JMP   TF7       IT'S THERE.
0193  0B77  FA6F  TF5     JST   GNB       GET NEXT TYPE CODE
0194  0B78  C005          CAI   5         TEST FOR REL ORG
0195  0B79  F205          JMP   TF1       IT IS
0196  0B7A  D263          CMS   HF        AND TEST IT
0197  0B7B  0000          NOP
0198  0B7C  F209          JMP   TF4       NOT NAM TEST FOR LOAD
0199  0B7D  FA45  TF3     JST   GNW       GET FIRST 2 LETTERS OF NAM
0200  0B7E  F63A          JMP   TF
0201  0B7F  FA43  TF1     JST   GNW       GET LOAD VALUE
0202  0B80  2102          JAZ   $+3       IGNORE IF 0
0203  0B81  FA06          JST   TF2       NOT 0, TEST FOR LOAD
0204  0B82  F64D          JMP   T5        LOAD, GO TO ORG PROCESSOR
0205  0B83  FA63          JST   GNB       GET NEXT TYPE CODE
0206  0B84  C00F          CAI   :F        NAME TYPE CODE?
0207  0B85  F608          JMP   TF3       YES
0208  0B86  FA01  TF4     JST   TF2       NO, TEST FOR LOAD
0209  0B87  F22F          JMP   STARTA    LOAD, GO
```

```
0210  0B88  0800  TF2     ENT
0211  0B89  DA42          IMS   FLOAD     OTHERWISE TEST FOR 1ST LOAD
0212  0B8A  F201          JMP   $+2
0213  0B8B  F703          RTN   TF2       LOAD THIS PROGRAM
0214  0B8C  5A01          INX   1         READ DATA SWITCHES
0215  0B8D  13AB          LRX   4         FORCE LOAD?
0216  0B8E  3201          JOR   $+2
0217  0B8F  F707          RTN   TF2       YES
0218  0B90  E23A          LDX   NAMEC     NOT 1ST LOAD,TEST FOR NAMES
0219  0B91  3842          JXN   $-2       NAMES, CONTINUE
0220  0B92  DAF3          IMS   SKIPF     NO, SET SKIP FLAG
0221  0B93  F222          JMP   START
0222  0B94  2083  TF7     JAM   TF8       DEFINED?
0223  0B95  C6CD          LAP   'M'       YES  ERROR
0224  0B96  FAFB          JST   PACHAR
0225  0B97  F620          JMP   TF5       CONTINUE
0226  0B98  DA32  TF8     IMS   NAMEC     BUMP NEED FLAG
0227  0B99  F622          JMP   TF5
0228                    * START,INITIALIZE PROGRAM
0229  0B9A  0000  EXT     DATA  0         REF/DEF TABLE START LOC.
0230  0B9B  9A32          STA   RBASE     SET RELOCATE BIAS
0231  0B9C  9A32          STA   LLOC
0232  0B9D  9A2C          STA   HIGH
0233  0B9E  3801          JXN   $+2       SKIP IF NOT ZERO
0234  0B9F  C4FD          LXP   :FD       OTHERWISE SET TO HEX FD
0235  0BA0  EA2F          STX   SBASE
0236  0BA1  0130          IXA             SET UP COMPARE
0237  0BA2  A2E7          IOR   CXI
0238  0BA3  9EAB          STA   BPT
0239  0BA4  13AF          LRX   8         FLUSH BITS 0-7
0240  0BA5  2801          JXZ   $+2
0241  0BA6  F23C          JMP   ERRS
0242  0BA7  0010          ARM
0243  0BA8  9A28          STA   ELOC      INITIALIZE START VECTOR
0244  0BA9  8E0F          ADD   EXT
0245  0BAA  9E10          STA   EXT       ADJUST REF/DEF LOCATION
0246  0BAB  9ADC          STA   LLIM      SET LOWER LIMIT
0247  0BAC  8ADE          ADD   END
0248  0BAD  9AD9          STA   ULIM      COMPUTE UPPER LIMIT
0249  0BAE  0110          ZAR
0250  0BAF  9A29          STA   NSY       NUMBER OF SYMBOLS IN TABLE
0251  0BB0  9A29          STA   UND       NUMBER OF UNDEFINED SYMBOLS
0252  0BB1  FAF9          JST   CRLF      INITIALIZE PRINTER
0253  0BB2  0010  FL      ARM
0254  0BB3  9A27          STA   NRD       FORCE A BLOCK RECORD READ
0255  0BB4  9A17          STA   FLOAD     FORCE A LOAD
0256  0BB5  9A17          STA   FREAD     FORCE A READ
0257                    * START AND RECYCLE
0258  0BB6  FA30  START   JST   GNB       GET NEXT TYPE BYTE
0259  0BB7  D2D1  STARTA  CMS   H10       TEST TYPE CODE LIMIT
0260  0BB8  F201          JMP   $+2
0261  0BB9  F6A6          JMP   ERRT      TYPE ERROR
```

```
0262  0BBA  3101          JAN   $+2       END OF FILE?
0263  0BBB  F69C          JMP   T0        YES, PROCESS
0264  0BBC  8E22          ADD   EXT
0265  0BBD  0048          TAX             AND SAVE
0266  0BBE  B400          LDA   #0
0267  0BBF  8E25          ADD   EXT
0268  0BC0  0048          TAX
0269  0BC1  FA01          JST   GNW       READ FIRST WORD
0270  0BC2  F400          JMP   #0
0271                *  THIS ROUTINE RETURNS THE NEXT INPUT BUFFER
0272                *                     BYTE IN THE A REGISTER.
0273  0BC3  0800  GNW     ENT
0274  0BC4  FA22          JST   GNB       GET FIRST OF TWO BYTES
0275  0BC5  1357          LLA   8         OVER HALF WORD
0276  0BC6  9A0C          STA   TEMP1
0277  0BC7  FA1F          JST   GNB       GET SECOND ONE
0278  0BC8  A20A          IOR   TEMP1
0279  0BC9  F706          RTN   GNW       EXIT
0280                *  CONSTANT AND STORAGE LOCATIONS
0281  0BCA  0000  HIGH    DATA  0
0282  0BCB  0000  NAMEC   DATA  0         COUNT OF NAME DEFINITIONS
0283  0BCC  FFFF  FLOAD   DATA  :FFFF     FORCE LOAD FIRST TIME
0284  0BCD  FFFF  FREAD   DATA  :FFFF     FORCE READ 1ST TIME
0285  0BCE  0000  RBASE   DATA  0         PROGRAM RELATIVE BIAS
0286  0BCF  0000  LLOC    DATA  0         CURRENT LOAD LOCATION
0287        0BD0  SBASE   EQU   $         SPECIAL (PAGE 0)
0288  0BD0  0000  SLOC    DATA  0         LOC CTR FOR SPECIAL AREA
0289  0BD1  FFFF  ELOC    DATA  -1        EXECUTE LOCATION
0290  0BD2  0000  TEMP    DATA  0         TEMPORARY STORAGE
0291  0BD3  0001  TEMP1   DATA  1         TEMPORARY STORAGE
0292  0BD4  0000  TEMP3   DATA  0         TEMPROARY STORAGE
0293  0BD5  0000  TRY     DATA  0         TEMPROARY COUNT
0294  0BD6  0000  SPELL   DATA  0         LAST SPELLING PROCESSED
0295  0BD7  0000          DATA  0
0296  0BD8  0000          DATA  0
0297  0BD9  0000  NSY     DATA  0         NO OF SYMBOLS IN ENT/EXT TA
0298  0BDA  0000  UND     DATA  0         NUMBER OF UNDEFINED SYMBOLS
0299  0BDB  FFFF  NRD     DATA  -1        BLOCK RECORD SIZE
0300  0BDC  7FFF  H7FFF   DATA  :7FFF     CONSTANT
0301  0BDD  8000  H8000   DATA  :8000     CONSTANT
0302  0BDE  000F  HF      DATA  :F        CONSTANT
0303                *  THIS ROTUINE PROCESSES ERRORS
0304  0BDF  C6C3  ERRC    LAP   'C'       CHECKSUM ERROR
0305  0BE0  F203          JMP   ERR
0306  0BE1  C6CC  ERRL    LAP   'L'       LOAD LOCATION OVERFLOW
0307  0BE2  F201          JMP   ERR
0308  0BE3  C6D3  ERRS    LAP   'S'       PAGE ZERO LOCATION OVERFLOW
0309  0BE4  FAA7  ERR     JST   PAHEX     PRNT CHAR IN A AND LOAD LOC
0310  0BE5  0801          STOP  1         ERROR HALT
0311  0BE6  F601          JMP   $-1       MUST START ALL OVER ! ! !
0312                *  THIS ROUTINE RETURNS THE NEXT BYTE IN THE
0313                *                     A REGISTER
```

```
0314                    * THE ROUTINE TESTS CHECKSUM, SKIPS RECORDS, AND
0315                    * PROCESSES THE BLOCK RECORD MARK AND WORD COUNT
0316  0BE7  0800  GNB    ENT
0317  0BE8  DE0D  GNBA   IMS   NRD       START NEW RECORD
0318  0BE9  F217         JMP   GNB2      NO, CONTINUE
0319  0BEA  DE1D         IMS   FREAD     FIRST READ
0320  0BEB  F201         JMP   $+2       NO CONTINUE
0321  0BEC  F205         JMP   GNB1      YES, DON'T DO CHECKSUM
0322  0BED  B2D1         LDA   CKSUM
0323  0BEE  9E1C         STA   TEMP      SAVE CKECKSUM
0324  0BEF  FAD2         JST   RDW       READ CHECKSUM
0325  0BF0  961F         SUB   TEMP      AND TEST FOR EQUAL
0326  0BF1  3152         JAN   ERRC      CHECKSUM ERROR
0327  0BF2  FAD6  GNB1   JST   RDB       GET NEXT BYTE
0328  0BF3  92CA         SUB   HFF       TEST FOR RECORD MARK
0329  0BF4  3142         JAN   GNB1      LOOP UNTIL FIND :F (EOF)
0330  0BF5  9AC9         STA   CKSUM
0331  0BF6  FACB         JST   RDW       OTHERWISE GET WORD COUNT
0332  0BF7  2110         JAZ   GNB4      EOF?
0333  0BF8  0310         NAR             NO
0334  0BF9  9E1F         STA   NRD       -(WORD COUNT)
0335  0BFA  FACE         JST   RDB       GET NEXT BYTE
0336  0BFB  00D0         DAR             TYPE CODE 1?
0337  0BFC  3102         JAN   $+3
0338  0BFD  9E32         STA   NAMEC     YES
0339  0BFE  9A87         STA   SKIPF     RESET SKIP FLAG
0340  0BFF  0150         IAR
0341  0C00  F201         JMP   $+2       TEST FOR LOADING
0342  0C01  FAC7  GNB2   JST   RDB
0343  0C02  BA83         EMA   SKIPF     TEST FOR SKIP
0344  0C03  2102         JAZ   GNB3      NO, CONTINUE
0345  0C04  BA81         EMA   SKIPF     YES, GET NEXT BYTE
0346  0C05  F61D         JMP   GNBA
0347  0C06  BA7F  GNB3   EMA   SKIPF
0348  0C07  F720         RTN   GNB
0349  0C08  9A7D  GNB4   STA   SKIPF     EOF
0350  0C09  F722         RTN   GNB
0351                    * THIS ROUTINE STORES A AT THE LOAD LOCATION
0352                    * AND INCREMENTS THE LOAD LOCATION. THE LOAD
0353                    * LOCATION IS TESTED FOR OVERFLOW
0354                    * HIGH IS UPDATED TO THE HIGHEST LOCATION+1 USED
0355  0C0A  0800  SLLOC  ENT
0356  0C0B  0048         TAX             SAVE STORAGE DATA
0357  0C0C  B63D         LDA   LLOC      LOAD LOCATION
0358  0C0D  20EC         JAM   ERRL      TEST LOAD LOC. FOR OVERFLOW
0359  0C0E  D279         CMS   LLIM      OR WITHIN LOADER
0360  0C0F  F203         JMP   SLLOCA
0361  0C10  D276         CMS   ULIM
0362  0C11  F630         JMP   ERRL
0363  0C12  0000         NOP
0364  0C13  EF44  SLLOCA STX   *LLOC     STORE DATA
0365  0C14  DE45         IMS   LLOC      INCREMENT LOAD LOCATION
```

```
0366  0C15  B646          LDA  LLOC     GET LOAD LOCATION
0367  0C16  D64C          CMS  HIGH     COMPARE TO HIGHEST
0368  0C17  F70D          RTN  SLLOC    LESS THEN HIGH,
0369  0C18  9L4E          STA  HIGH     GREATER THEN HIGH,
0370  0C19  F70F          RTN  SLLOC    EQUAL TO HIGH
0371                    *
0372  0C1A  0800    PUND  ENT
0373  0C1B  F20C          JMP  PUND4    GOTO ROUTINE
0374                    *
0375                    * THIS ROUTINE PRINTS AN E FOLLOWED BY THE BASE
0376                    * PAGE, LOAD, AND EXECUTE LOCATIONS.
0377  0C1C  0800    PEXEC ENT
0378  0C1D  C6C5          LAP  'E'
0379  0C1E  FAC2          JST  PTTY
0380  0C1F  B64F          LDA  SLOC     BASE PAGE LOCATION
0381  0C20  FA7A          JST  PHEX
0382  0C21  B657          LDA  HIGH     HIGHEST LOAD LOCATION
0383  0C22  FA78          JST  PHEX
0384  0C23  B652          LDA  ELOC     EXECUTE LOCATION
0385  0C24  2081          JAM  $+2      IF MINUS DON'T LIST
0386  0C25  FA75          JST  PHEX
0387  0C26  FA84          JST  CRLF
0388  0C27  F70B          RTN  PEXEC
0389                    * THIS ROUTINE SEARCHES THE SYMBOL TABLE FOR
0390                    * UNDEFINED EXTERNAL SYMBOLS AND LISTS THEM.
0391  0C28  E68E    PUND4 LDX  EXT      GET SYMBOL TABLE START
0392  0C29  B650          LDA  NSY      NUMBER OF SYMBOLS
0393  0C2A  0310          NAR
0394  0C2B  2108          JAZ  PUND3    IF ZERO, EXIT
0395  0C2C  9E57          STA  TRY
0396  0C2D  C305    PUND1 SXI  5
0397  0C2E  B400          LDA  #0       GET NEXT DEFINITION
0398  0C2F  3082          JAP  PUND2    IF DEFINED, CONTINUE
0399  0C30  C6D5          LAP  'U'      OTHERWISE PRINT U AND
0400  0C31  FA60          JST  PACHAR   UNDEFINED SYMBOL
0401  0C32  DE5D    PUND2 IMS  TRY      TEST FOR DONE
0402  0C33  F606          JMP  PUND1    REPEAT IF NOT DONE
0403  0C34  F71A    PUND3 RTN  PUND
0404                    * THIS ROUTINE WILL FIND A SYMBOL IN THE
0405                    * EXTERNAL SYMBOL TABLE OR WILL PUT THE SYMBOL
0406                    * IN THE NEXT AVAILABLE EXTERNAL SYMBOL TABLE
0407                    * LOCATION. THE EXTERNAL SYMBOL TABLE ADDRESS
0408                    * IS RETURNED IN THE A REGISTER AND THE LOCATION
0409                    * IN X. THE TABLE IS TESTED FOR OVERLFOW
0410  0C35  0800    FSYM  ENT
0411  0C36  FA1B          JST  SEXT     SEARCH FOR SYMBOL
0412  0C37  F702          RTN  FSYM     SYMBOL IN TABLE
0413  0C38  C305          SXI  5
0414  0C39  0030          TXA
0415  0C3A  D283          CMS  HFF      TEST FOR IN BASE PAGE
0416  0C3B  F658          JMP  ERRS     OVERFLOW
0417  0C3C  0000          NOP
```

```
0418  0C3D  B667          LDA  SPELL     GET CHARACTERS
0419  0C3E  9C04          STA  @4
0420  0C3F  B668          LDA  SPELL+1
0421  0C40  9C03          STA  @3
0422  0C41  B669          LDA  SPELL+2
0423  0C42  9C02          STA  @2
0424  0C43  DE6A          IMS  NSY       ADD TO COUNT OF SYMBOLS
0425  0C44  B667          LDA  H8000     UNDEFINED SYMBOL INDICATOR
0426  0C45  9C01          STA  @1        UNDEFINED PAGE 0 REF FLAG
0427  0C46  9C00          STA  @0        UNDEFINED REFERENCE FLAG
0428  0C47  EA40          STX  LLIM
0429  0C48  DE6E          IMS  UND       COUNT UNDEFINED SYMBOLS
0430  0C49  0000          NOP
0431  0C4A  F715          RTN  FSYM
0432                * STRING 6 SYMBOL CHARACTERS IN 3 WORD LOCATION
0433                * SPELL
0434  0C4B  0800    RDS   ENT
0435  0C4C  9E76          STA  SPELL     STRING 6 SYMBOL CHARS
0436  0C4D  FE8A          JST  GNW
0437  0C4E  9E77          STA  SPELL+1
0438  0C4F  FE8C          JST  GNW
0439  0C50  9E78          STA  SPELL+2
0440  0C51  F706          RTN  RDS       EXIT
0441                * THIS ROUTINE SEARCHES THE EXTERNAL DEFINITION
0442                * TABLE FOR THE SYMBOL IN LOCATION-STRING SPELL.
0443                * IF FOUND, THE EXTERNAL DEFINITION ADDRESS IS
0444                * RETURNED IN X AND THE DEFINITION ADDRESS IS
0445                * RETURNED IN A, AS FOLLOWS:
0446                *       JST  SEXT
0447                *        +1  SYMBOL IN TABLE
0448                *        +2  SYMBOL NOT IN TABLE
0449  0C52  0800    SEXT  ENT
0450  0C53  E6B9          LDX  EXT       POINT AT TABLE
0451  0C54  B67B          LDA  NSY       NO OF SYMBOLS
0452  0C55  210E          JAZ  SEXT2A    FIRST SYMBOL, EXIT
0453  0C56  0310          NAR            NEGATIVE COUNT TO
0454  0C57  9E82          STA  TRY       TRIAL COUNTER
0455  0C58  C305    SEXT1 SXI  5         BACK DOWN 5 WORDS
0456  0C59  B683          LDA  SPELL
0457  0C5A  AC04          XOR  @4
0458  0C5B  3106          JAN  SEXT2     NO MATCH
0459  0C5C  B685          LDA  SPELL+1
0460  0C5D  AC03          XOR  @3
0461  0C5E  3103          JAN  SEXT2     NO MATCH
0462  0C5F  B687          LDA  SPELL+2
0463  0C60  AC02          XOR  @2
0464  0C61  2103          JAZ  SEXT3     FOUND SYMBOL
0465  0C62  DE8D    SEXT2 IMS  TRY       TEST FOR ALL DONE
0466  0C63  F608          JMP  SEXT1     NO, TRY AGAIN
0467  0C64  DE12    SEXT2A IMS SEXT
0468  0C65  B400    SEXT3 LDA  @0        DEFINITION
0469  0C66  F714          RTN  SEXT
```

```
0470                    * THIS ROUTINE DEFINES A SYMBOL. THE EXTERNAL
0471                    * SYMBOL TABLE IS TESTED FOR OVERFLOW.
0472                    * ALL UNDEFINED REF AND EXTR REFERENCES ARE DEFINED.
0473  0C67  0800M DSYM   ENT
0474  0C68  9E96         STA  TEMP      SAVE DEFINITION
0475  0C69  FE34         JST  FSYM
0476  0C6A  309A         JAP  DSYM3     ERROR IF SYMBOL DEFINED
0477  0C6B  B691         LDA  UND       DEC NO. OF UNDEF SYMBOLS
0478  0C6C  00D0         DAR
0479  0C6D  9E93         STA  UND
0480  0C6E  EE9B         STX  TEMP1
0481  0C6F  B401         LDA  *1        PAGE ZERO REFERENCE
0482  0C70  AE93         XOR  H8000
0483  0C71  2104         JAZ  DSYM4     NO UNDEF REF YET,CONTINUE
0484  0C72  9C01         STA  *1        ELSE DEFINE IT IN PAGE ZERO
0485  0C73  0048         TAX
0486  0C74  B6A2         LDA  TEMP
0487  0C75  9C00         STA  *0
0488  0C76  E6A3 DSYM4   LDX  TEMP1
0489  0C77  B6A5 DSYM1   LDA  TEMP      PICK UP DEFINITION
0490  0C78  BC0M         EMA  *0        SWAP DEFINITION AND POINTER
0491  0C79  AE9C         XOR  H8000     GET ACTUAL ADDR OF THREAD
0492  0C7A  2102         JAZ  DSYM2
0493  0C7B  0048         TAX
0494  0C7C  F605         JMP  DSYM1     CONTINUE
0495  0C7D  C604 DSYM2   LAP  4         LIST DEF?
0496  0C7E  5C01         INAM 1         READ DATA SWITCHES
0497  0C7F  3105         JAN  DSYM3
0498  0C80  E6AD         LDX  TEMP1
0499  0C81  FA2F         JST  PCHAR     SYMBOL
0500  0C82  B6B0         LDA  TEMP
0501  0C83  FA17         JST  PHEX      DEFINITION
0502  0C84  FA26         JST  CRLF
0503  0C85  F71E DSYM3   RTN  DSYM
0504  0C86  000M SKIPF   DATA 0         SKIP FLAG
0505  0C87  000M ULIM    DATA 0         LOADER UPPER BOUND
0506  0C88  000M LLIM    DATA 0         LOADER LOWER BOUND
0507  0C89  001M H10     DATA :10       CONSTANT
0508  0C8A  C10M CXI     CXI  0
0509  0C8B  0211 END     DATA ENDL-BEGG+1   LOADER SIZE
0510                    * THIS ROUTINE PRINTS THE CHARACTER IN A
0511                    * FOLLOWED BY THE LOAD LOCATION.
0512  0C8C  080M PAHEX   ENT
0513  0C8D  FA53         JST  PTTY      PRINT CHAR IN A
0514  0C8E  B6BF         LDA  LLOC
0515  0C8F  FA0B         JST  PHEX      PRINT LOAD LOCATION
0516  0C90  FA1A         JST  CRLF      CR LF
0517  0C91  F705         RTN  PAHEX
0518                    * THIS ROUTINE PRINTS THE SYMBOL FOLLOWED BY
0519                    * A BLANK, FOLLOWED BY THE CHARACTER IN A.
0520  0C92  080M PACHAR  ENT
0521  0C93  9EC0         STA  TEMP1
```

```
0522  0C94  FA1C          JST  PCHAR     FOLLOWED BY THE SYMBOL
0523  0C95  C6A0          LAP  '         '
0524  0C96  FA4A          JST  PTTY      FOLLOWED BY A BLANK
0525  0C97  B6C4          LDA  TEMP1
0526  0C98  FA48          JST  PTTY      PRINT CHAR IN A
0527  0C99  FA11          JST  CRLF      CR LF
0528  0C9A  F708          RTN  PACHAR
0529                  * THIS ROUTINE PRINTS BLANK FOLLOWED BY THE
0530                  * FOUR DIGIT HEX NUMBER IN A.
0531  0C9B  0800  PHEX    ENT
0532  0C9C  0048          IAX            4 DIGITS
0533  0C9D  C6A0          LAP  '         '
0534  0C9E  FA42          JST  PTTY
0535  0C9F  C704          LAM  4
0536  0CA0  9A20          STA  TEMP2     SAVE COUNT
0537  0CA1  1803  PHEX1   LLL  4         NEXT DIGIT
0538  0CA2  86C4          AND  HF
0539  0CA3  D23A          CMS  HA
0540  0CA4  8A3A          ADD  M7
0541  0CA5  0000          NOP
0542  0CA6  8A39          ADD  HB7
0543  0CA7  FA39          JST  PTTY      PRINT IT
0544  0CA8  DA18          IMS  TEMP2     TEST COUNT FOR DONE
0545  0CA9  F6D8          JMP  PHEX1     NO, REPEAT
0546  0CAA  F70F          RTN  PHEX
0547                  * THIS ROUTINE PRINTS A CARR. RETURN, LINE FEED.
0548  0CAB  0800  CRLF    ENT
0549  0CAC  C68D          LAP  :8D
0550  0CAD  FA33          JST  PTTY
0551  0CAE  C68A          LAP  :8A
0552  0CAF  FA31          JST  PTTY
0553  0CB0  F705          RTN  CRLF
0554                  * THIS ROUTINE TYPES 6 ASCII CHARACTERS FROM
0555                  * THE LOCATION INDICATED BY X.
0556  0CB1  0800  PCHAR   ENT
0557  0CB2  C203          AXI  3
0558  0CB3  C723          LAM  3
0559  0CB4  9A0C          STA  TEMP2     SAVE COUNT
0560  0CB5  B401  PCHAR1  LDA  @1        NEXT CHAR, LEFT BYTE
0561  0CB6  13D7          LRA  8
0562  0CB7  FA29          JST  PTTY
0563  0CB8  B401          LDA  @1        NEXT CHAR, RIGHT BYTE
0564  0CB9  FA27          JST  PTTY
0565  0CBA  00AB          DXR
0566  0CBB  DA05          IMS  TEMP2     TEST COUNT FOR DONE
0567  0CBC  F6D7          JMP  PCHAR1    NO, REPEAT
0568  0CBD  F70C          RTN  PCHAR     YES, DONE
0569  0CBE  00FF  HFF     DATA :FF       CONSTANT
0570  0CBF  0000  CKSUM   DATA 0         CHECKSUM
0571  0CC0  0080  H80     DATA :80       CONSTANT
0572  0CC1  0000  TEMP2   DATA 0         TEMPORARY STORAGE
0573                  * THIS ROUTINE READS A WORD AND RETURNS THE
```

```
0574                    * WORD IN THE A REGISTER.
0575            0CC2    RDW     EQU   $
0576    0CC2    0800    RPTW    ENT
0577    0CC3    FA05            JST   RPTB      GET HI BYTE
0578    0CC4    1357            LLA   8         SAVE
0579    0CC5    9E04            STA   TEMP2
0580    0CC6    FA02            JST   RPTB      GET LO BYTE
0581    0CC7    A606            IOR   TEMP2
0582    0CC8    F706            RTN   RPTW      EXIT
0583                    * THIS ROUTINE READS A BYTE AND RETURNS THE
0584                    * BYTE IN THE A REGISTER.
0585            0CC9    RDB     EQU   $
0586    0CC9    0800    RPTB    ENT               READ PAPER TAPE BYTE
0587    0CCA    0110            ZAR               SETUP TIMES OUT FOR TTY
0588    0CCB    4032            SEL   6,2         STEP  HSR
0589    0CCC    403A            SEL   7,2         STEP  TTY
0590    0CCD    0150            IAR               TIME TTY INPUT
0591    0CCE    2142            JAZ   $-2         TIME OUT?
0592    0CCF    4831            SSN   6,1         HSR READY?
0593    0CD0    F204            JMP   RHSR        YES
0594    0CD1    4939            SEN   7,1         TTY READY?
0595    0CD2    F605            JMP   $-5         NO, LOOP AGAIN.
0596    0CD3    5838            INA   7,0         READ FROM TTY
0597    0CD4    F201            JMP   $+2
0598    0CD5    5830    RHSR    INA   6,0         READ FROM HSR
0599    0CD6    BE17            EMA   CKSUM       GET CHECKSUM
0600    0CD7    11D0            RRA   1           ROTATE RIGHT 1 BYTE
0601    0CD8    3201            JOR   $+2
0602    0CD9    8E19            ADD   H80
0603    0CDA    AE1B            XOR   CKSUM       EXCLUSIVE OR
0604    0CDB    861D            AND   HFF         CLEAR HIGH BYTE
0605    0CDC    BE1D            EMA   CKSUM       SAVE CHECKSUM
0606    0CDD    F714            RTN   RPTB        RETURN WITH BYTE
0607    0CDE    000A    HA      DATA  :A          CONSTANT
0608    0CDF    FFF9    M7      DATA  -7          CONSTANT
0609    0CE0    00B7    HB7     DATA  :B7         CONSTNANT
0610                    *
0611                    * THIS ROUTINE OUTPUTS A CHARACTER
0612                    *
0613            008A    LF      EQU   :8A         LINE PRINTER LINE FEED CHARACTER
0614    0CE1    0800    PTTY    ENT
0615    0CE2    9E19            STA   RPTB        SAVE CHARACTER
0616    0CE3    5801            ISA               READ DATA SWITCHES
0617    0CE4    13D0            LRA   1
0618    0CE5    B61C            LDA   RPTB
0619    0CE6    2205            JOS   HSPRNT      LIST ON LINE PRINTER?
0620    0CE7    403C            SEL   TDA,4       INITIALIZE TTY FOR OUTPUT
0621    0CE8    6C38            OTA   TDA,0       DOIT
0622    0CE9    4939            SEN   TDA,1       WAIT FOR DONE
0623    0CEA    F601            JMP   $-1
0624    0CEB    F70A            RTN   PTTY        EXIT
0625    0CEC    C08A    HSPRNT  CAI   LF          IGNORE LINE FEED?
```

```
0626  0CED  F70C  NOP:ME RTN  PTTY
0627  0CEE  6D21        WRA  LPDA,1    OUTPUT CHARACTER
0628  0CEF  F70E        RTN  PTTY      EXIT
0629              ENDL   END
0000 ERRORS
```

```
0019  BEGG      0021  0022  0023  0024  0025  0026  0027  0028
                0029  0030  0031  0032  0033  0034  0035  0036
                0509
0049  BPT       0238*
0570  CKSUM     0322  0330* 0599* 0603  0605*
0548  CRLF      0252* 0387* 0502* 0516* 0527* 0553
0508  CXI       0237
0473  DSYM      0178* 0503
0489  DSYM1     0494
0495  DSYM2     0492
0503  DSYM3     0476  0497
0488  DSYM4     0483
0289  ELOC      0069* 0111  0113  0243* 0384
0509  END       0247
0629  ENDL      0509
0309  ERR       0079  0305  0307
0304  ERRC      0326
0306  ERRL      0358  0362
0308  ERRS      0080  0241  0416
0080  ERRS1     0047
0078  ERRT      0038  0261
0229  EXT       0019* 0244  0245* 0264  0267  0391  0450
0253  FL        0108
0283  FLOAD     0092* 0211* 0255*
0284  FREAD     0256* 0319*
0410  FSYM      0152* 0182* 0412  0431  0475*
0316  GNB       0145* 0161* 0193* 0205* 0258* 0274* 0277* 0348
                0350
0327  GNB1      0321  0329
0342  GNB2      0318
0347  GNB3      0344
0349  GNB4      0332
0317  GNBA      0346
0273  GNW       0084* 0125* 0199* 0201* 0269* 0279  0436* 0438*
0507  H10       0259
0300  H7FFF     0154  0172
0571  H80       0602
0301  H8000     0170  0185  0425  0482  0491
0607  HA        0539
0101  HALT
0609  HB7       0542
0302  HF        0196  0538
0569  HFF       0328  0415  0604
0281  HIGH      0070  0074* 0100  0232* 0367  0369* 0382
0625  HSPRNT    0619
0077  ISTART    0110
0613  LF        0625
0506  LLIM      0246* 0359  0428*
0286  LLOC      0071  0072  0076* 0103* 0117* 0177  0184  0231*
                0357  0364* 0365* 0366  0514
0006  LPDA      0627
0608  M7        0540
```

```
0282  NAMEC     0218  0226* 0338*
0626  NOP:ME
0299  NRD       0099* 0254* 0317* 0334*
0297  NSY       0250* 0392  0424* 0451
0520  PACHAR    0224* 0400* 0528
0512  PAHEX     0309* 0517
0556  PCHAR     0499* 0522* 0568
0560  PCHAR1    0567
0377  PEXEC     0097* 0388
0531  PHEX      0381* 0383* 0386* 0501* 0515* 0546
0537  PHEX1     0545
0614  PTTY      0379* 0513* 0524* 0526* 0534* 0543* 0550* 0552*
                0502* 0564* 0624  0626  0628
0372  PUND      0096* 0403
0396  PUND1     0402
0401  PUND2     0398
0403  PUND3     0394
0391  PUND4     0373
0285  RBASE     0066  0075* 0102* 0115  0128  0134  0135  0139
                0141  0230*
0585  RDB       0327* 0335* 0342*
0434  RDS       0151* 0176* 0181* 0190* 0440
0575  RDW       0324* 0331*
0598  RHSR      0593
0586  RPTB      0577* 0580* 0606  0615* 0618
0576  RPTW      0582
0287  SBASE     0235*
0449  SEXT      0191* 0411* 0467* 0469
0455  SEXT1     0466
0465  SEXT2     0458  0461
0467  SEXT2A    0452
0468  SEXT3     0464
0504  SKIPF     0220* 0339* 0343* 0345* 0347* 0349*
0355  SLLOC     0088* 0129* 0136* 0148* 0187* 0368  0370
0364  SLLOCA    0360
0288  SLOC      0046  0057* 0058  0060* 0380
0294  SPELL     0123* 0126  0418  0420  0422  0435* 0437* 0439*
                0456  0459  0462
0344  SSLOC     0363  0143* 0166* 0169*
0348  SSLOCA    0054
0357  SSLOCB    0050  0051
0362  SSLOCC    0056
0258  START     0039  0077  0087  0118  0132  0137  0149  0179
                0188  0221
0259  STARTA    0209
0392  T0        0263
0397  T0A       0112
0238  T1        0021
0139  T10       0036
0268  T2        0022
0273  T21       0065  0068
0265  T3        0023
```

```
    0117  T4          0024
    0115  T5          0025  0204
    0122  T6          0026
    0125  T6A         0131
    0120  T7          0027
    0082  T8          0028
    0085  T8A         0090
    0134  T9          0029
    0143  TA          0030
    0144  TAA         0173
    0146  TAAA        0164
    0141  TB          0031
    0151  TC          0032
    0173  TCA         0155
    0171  TCB         0167
    0168  TCC         0157
    0165  TCD         0160
    0176  TD          0033
    0007  TDA         0620  0621  0622
    0181  TE          0034
    0187  TEA         0183
    0290  TEMP        0144* 0147  0158* 0323* 0325  0474* 0486  0489
                      0500
    0291  TEMP1       0276* 0278  0480* 0488  0498  0521* 0525
    0572  TEMP2       0536* 0544* 0559* 0566* 0579* 0581
    0292  TEMP3       0045* 0062
    0190  TF          0035  0200
    0201  TF1         0195
    0210  TF2         0203* 0208* 0213  0217
    0199  TF3         0207
    0208  TF4         0198
    0193  TF5         0225  0227
    0222  TF7         0192
    0226  TF8         0222
    0096  TOB         0104
    0293  TRY         0083* 0085* 0124* 0130* 0395* 0401* 0454* 0465*
    0505  ULIM        0248* 0361
    0298  UND         0109  0251* 0429* 0477  0479*
0629 SOURCE LINES
```