LTBOSS
INTERDATA BOSS FOR LINC TAPE
USER MANUAL

Computer Operations, Inc.
10774 Tucker Street
Beltsville, Maryland 20705

Doc. No. 126
November, 1973

TABLE OF CONTENTS

Page

INTRODUCTION 1

BOSS OPERATOR COMMANDS 2

DIRECT REFERENCES TO LINC TAPE 2

DIRECTORY REFERENCES TO LINC TAPE 4

File Directories 4
File Directory Commands 6
Loading and Dumping Core 8

DEVICE-INDEPENDENT REFERENCES TO LINC TAPE 9

Creating Buffers 9
Assigning Linc Tape Files 11
SVC Calls for Linc Tape 12
Rewind and Write EOF Commands 13
Miscellaneous I/O Features 13

LOADING PROGRAMS 15

COPY COMMANDS 16

MEMORY EXAMINATION 17

THE SYSTEM TAPE 18

OPERATING INSTRUCTIONS 20

EXAMPLE 21

LTBOSS - BOSS for Linc Tape
User Manual

INTRODUCTION

LTBOSS is a revision and extension of Interdata BOSS/4B, which is described in Basic Operating System (BOSS) Program Manual (Interdata Publication B29-216) or in Section 11.2 of the Interdata Model 70 User's Manual. This document describes the extensions and changes to BOSS/4B, and assumes a knowledge of one of the Interdata documents.

LTBOSS allows use of Linc Tape in two ways - direct reference and device-independent reference. For direct reference, the user must specify which block(s) are to be read or written, and must manually keep track of which blocks are used and for what purposes. Both operator commands and SVC calls are provided for direct reference.

Device-independent Linc Tape references are an extension of the device-independent logical I/O features of BOSS. Linc Tape files are referenced by name, and a directory stored on each tape records the names and their block assignments. Operator commands permit file creation and directory maintenance, while standard SVC calls are used for character-oriented I/O. Since Linc Tape is physically block-oriented, buffers must be used; so there are operator commands for assigning buffers.

All features presented in this manual are not present in all versions of LTBOSS. Assembly options allow tailoring a version of LTBOSS with only the features desired, and thereby smaller in size than a version with all the features. The LTBOSS Program Maintenance Manual describes these options.

In the description and examples which follow, operator commands typed by the user are underlined to distinguish them from responses typed by BOSS.

BOSS OPERATOR COMMANDS

The operator command structure has been extended to allow up to four arguments rather than just the one permitted in BOSS/4B. The format is:

OPERATOR xxxx xxxx xxxx xxxx

where each xxxx is a hexadecimal number of from one to four digits. Any arguments which are not given are assumed to be zero. Many commands do not require (or use) all four arguments. Use only one space between arguments.

Any OPERATOR may be abbreviated OP as only the first two letters are actually used by the BOSS command lookup.

DIRECT REFERENCE TO LINC TAPE

Two operator commands are provided:

```
RLT  xxxx  blok  nblk  tape
WLT  xxxx  blok  nblk  tape
```

RLT reads Linc Tape and WLT writes Linc Tape where:

xxxx is the starting core location;
blok is the starting block number;
nblk is the number of blocks to be transferred;
tape is the desired tape unit.

If there is an error, the message

```
I/O ERR  xxxx
```

will be printed, where

xxxx is the error code from the Linc Tape Utility routines.

Two corresponding SVC commands are provided for direct reference Linc Tape I/O from user programs. Their form is:

```
SVC  2,A(X2)
```

where the parameter block is:

code
xxxx
blok
nblk
tape

Code is 8 to read tape and 9 to write tape. The other arguments are the same as for the operator commands. Upon return to the instruction following the SVC, Register 0 will contain the Linc Tape status, which will be zero if the transfer was performed without error, or will be the error code defined above if there was an error. The condition code will reflect the state of Register 0.

DIRECTORY REFERENCES TO LINC TAPE

File Directories

To avoid the necessity of remembering block assignments, LTBOSS provides the capability to create and maintain a directory on each tape. The directory occupies block zero and consists of 0-31 file entries. As a backup measure, the directory is always written in both block -1 and block 0, but only block 0 is used when reading the directory.

For each file, the directory stores two six-character names, a starting block number, and a length in 16-bit words. The blocks occupied by a file are always contiguous. The file:

TEST FILE 027 0213

starts at block X'27' and is X'213' words long. The length in blocks is the first two digits of the length, plus one if the last two digits are non-zero. In this example, TEST FILE is 2+1=3 blocks long, and occupies blocks X'27', X'28', and X'29'.

The Linc Tape Directory consists of 1-32 entries, each 16 bytes long. The first entry may contain tape number, user name, or any other labelling information. It is ignored by all commands.

Each of the remaining entries consists of two six-character names (file first and second names), and two 16-bit numbers giving the starting block number and length in words (one-half the length in bytes) for the file. If the first halfword (two bytes) of an entry is zero, the entry is unused, and the rest of its contents are ignored.

A sample entry has this format (halfwords shown):

␣	N	all zero for "unused"
A	M	
E	1	
␣	N	
A	M	
E	2	
start block		
file length		

To initialize a new tape, block 0 should be written all zeroes to indicate an empty directory. This can be done with the command:

```
WLT xxxx 0 1 tape
```

where xxxx is the starting address of a block of 256 sequential zero bytes (an address above available core works well). If tape labelling information is desired, it should be entered into the first 16 bytes.

File Directory Commands

The following commands create and maintain directories. Underlined information is typed by the user. For an illustration of the use of these commands, see the example appended to this document.

1) To list all the files on a tape,

```
LISTF   tape
```

where: tape is the desired tape unit.

2) To create a new file entry in the directory,

```
FENTER   tape   blok   nwds
FILE NAME
name1    name2
```

where: tape is the desired tape unit
blok is the desired starting block
nwds is the desired length (or maximum length) in words
name1 is the first name of the file (maximum six letters)
name2 is the second name of the file (maximum six letters)

If blok is zero, LTBOSS will find an unused area on tape and assign it to the file. If nwds is zero, a length of one block (X'200') will be used.

3) To delete a file from a directory,

```
DELETE   tape
FILE NAME
name1   name2
```

where: tape is the desired tape unit
name1 is the file first name
name2 is the file second name

4) To change the name of a file,

```
RNAME  tape
FILE NAME
name1  name2
FILE NAME
name3  name4
```

where: tape is the desired tape unit
name1 is the present file first name
name2 is the present file second name
name3 is the new file first name
name4 is the new file second name

Note that the command is RN for "rename" to distinguish it from RE for "replace".

5) To change a file start block and length,

```
CHANGE  tape  blok  nwd
FILE NAME
name1  name2
```

where: tape is the desired tape unit
name1 is the file first name
name2 is the file second name

Changing a file starting block only changes the directory entry and does not move the file.

NOTES: All directory commands require reading the directory from the tape. The directory is always read into a buffer which starts the top of core minus X'200' bytes. Care should be exercised not to have other information there which would be destroyed.

Whenever LTBOSS asks for a FILE NAME, the command may be aborted at this point by entering a null line (CR only or # followed by CR).

Loading and Dumping Core

Two commands allow areas of core to be written and read as named Linc Tape files.

1) To read a file into core,

```
GETFILE  tape  xxxx
FILE NAME
name1  name2
```

where: tape is the desired tape unit
xxxx is the starting core location. If omitted, X'1000' is assumed. (all system files with second name "IMAGE" are loaded at X'1000'.)
name1 is the file first name
name2 is the file second name

The directory will be read, and if the file exists, it will be read into core starting at location xxxx. Since only complete blocks can be read, if the length includes a partial block, that whole block will be read. The user should be sure to allow enough core.

This command is especially useful for loading saved core images to be executed. GETFILE does not change BIAS as the BOSS Internal Loader does.

2) To write a file from core onto tape,

```
PUTFILE  tape  xxxx
FILE NAME
name1  name2
```

where: the arguments are the same as for GETFILE. The directory will be read, and if the file exists, it will be written from core starting at xxxx. Again, if the file length is not an integral number of blocks, core through the next complete block will be written out.

This command is especially useful for saving core images on tape after loading for running at a later time.

DEVICE-INDEPENDENT REFERENCES TO LINC TAPE

To read or write Linc Tape using device-independent I/O, a buffer must be created, and the file assigned to a logical unit (LU) number. Thereafter, the standard SVC1 read/write calls may be used.

When reading,the first read instruction to that LU fills the buffer, and subsequent reads take data from it. When the buffer is empty, more data is read from tape. An attempt to read beyond the end of the buffer results in the end of file (EOF) error status byte being returned. The rewind (RW) command (or SVC call) allows the same file to be re-read from the beginning.

When writing, a file must be created on tape (by FENTER) before assigning. This file should be of length greater than or equal to the expected output. Write instructions enter data into the buffer until it is full, then the buffer is written onto tape. After completing output, it is necessary to give the WRTFIL (WF) command (or SVC call) to assure that the last (partial) buffer is written out. WF also updates the directory to indicate the exact file length. The RW command may be used as in reading to reset BOSS to the beginning of the file.

Creating Buffers

When LTBOSS is started at X'D0', one block buffers are automatically created for LU1 at top of core minus X'200' and for LU2 at top of core minus X'400'. The BOSS system parameter CTOP (core top), the last useable word of memory,is set to physical top of core minus X'402', so that programs will not overwrite these buffers.

When core is available, larger buffers may be created for greater efficiency of operation, or buffers may be assigned to some other location in core. The command to create a buffer is:

BUFFER lu xxxx nbyt

where: lu is the logical unit number
(Only LU1-3 may have Linc Tape buffers in LTBOSS.)
xxxx is the starting core address of the buffer.
nbyt is the length of the buffer in bytes. This must be an integer multiple of X'200'.

A buffer must be created before assigning a logical unit to Linc Tape. However, once created, a buffer remains until changed by a new BUFFER command (or by restarting LTBOSS at X'D0); so it is not necessary to give the command explicitly before each ASSIGN.

Creating buffers does not automatically reset the value for the top of core, since buffers are not necessarily at the top of core. A new command was added to allow convenient setting of CTOP, the system parameter for the top of available core. The command is:

```
CTOP   xxxx
```

where: xxxx is the last free halfword. In general, it should be (as a hex number) xx FE.

This value for CTOP is used, for example, by the assembler to set the upper limit for the symbol table. Therefore, if buffers are reassigned, CTOP should be given to assure that the assembler does not overwrite the buffers with symbol table entries.

The true top of core has been added as a seventh parameter to the constant table accessed by the " Fetch" SVC.

Assigning Linc Tape Files

Before making assignment, the file must exist in the directory of the desired tape. This is usually done by FENTER if the file is to be written, or the file is the result of some other operation if it is to be read.

The assignment command for devices other than Linc Tape is the standard:

```
AS  lupa
```

If pa is the Linc Tape device number, the format is extended as follows:

```
ASSIGN  lupa  tape
FILE NAME
name1  name2
```

where:
lu is the logical unit being assigned
pa is the Linc Tape physical address
name1 is the file first name
name2 is the file second name

SVC Calls for Line Tape

After the assignment has been made, standard BOSS I/O calls may be made. They are of the form:

```
SVC   1,A(X2)
```

where, the parameter block is:

FCN	LU
STATUS	DEV.ADDR.
A (Start)	
A (End)	

Each parameter has the same meaning as in standard BOSS I/O calls in which the high order bit of FCN is "0". The valid FCN codes are:

X'20'	Write ASCII
X'30'	Write BIN
X'40'	Read ASCII
X'50'	Read BIN

Two command functions are available (high order bit of FCN is "1". They are:

FCN = X'C0', Rewind - The file assigned to the logical unit is reset to the beginning.

FCN = X'88' Write EOF - The file being written on the logical unit is completed and the directory entry updated for its actual length.

For example, the assembler issues a rewind command for LU1 between pass one and pass two of an assembly. It would be nice if the assembler did a write EOF command to LU2 to complete binary output, but it does not; therefore, the operation version (WF) must be used.

Rewind and Write EOF Commands

Operator commands are available for the rewind and write End-of-File functions. They are:

RW lu
WF lu

Their actions are identical to those described above for the SVC versions.

Miscellaneous I/O Features

These commands allow special processing of ASCII input and output. Since they are controlled by assembly parameters, they do not exist in all versions of LTBOSS. They pertain to ASCII I/O for all devices, not just Linc Tapes.

It is sometimes desirable to accept input that includes TAB characters, but to convert the TAB character to the ASCII spare character (SP). The command:

SP xx

specifies a character xx (usually X'09' for TAB) which will be converted to SP (X'20') when encountered on input. Setting SP 00 effectively disables this function.

Two other features are available, but do not have associated operator commands. If the $LFEND assembly option is used, Line Feed is accepted the same as Carriage Return to end input lines, and Line Feed is used instead of Carriage Return to terminate output lines to Linc Tape. (Only a single character, either CR or LF, separates lines of text on Linc Tape.) If the $LWRCS option is used, lower case characters are converted to their upper case equivalents. This facilitates the use of input devices that produce the full ASCII character set.

After a program has been loaded (without offset), or read into core at the proper location via RLT or GET, it may be started by the command:

START xxxx

where: xxxx is the starting address

The PSW will be loaded with X'3400', and the program started. If xxxx is zero (or omitted), the program will be started at the transfer address (if the load process specified one), else at UBOT, the bottom of user core (and initial value of BIAS). (See Interdata writeup for a description of these variables.)

LOADING PROGRAMS

The LTBOSS loader has an extension which allows loading programs which would overwrite BOSS (e.g., a new version of BOSS itself). The new command is:

OFFSET xxxx

where: xxxx is an offset to be added to the load address of each word loaded by the LOAD command.

Absolute binary items are loaded at the specified address plus the offset. Relative binary items are loaded at the specified address plus the bias (set by the BIAS command) plus the offset. The load address should always be greater than the address of the top of LTBOSS to avoid overwriting it.

The message printed by the load command specifies the offset as well (if offset is non-zero) in the form:

BIAS bias + offset

If the offset is zero, only the bias value is printed.

Programs loaded with an offset are loaded to be run at their absolute or bias address. All relocatable references are affected only by BIAS. Therefore, one should not attempt to run a program as loaded with an offset. Instead, it should be written out on Linc Tape (via WLT or PUT), then read in (probably with the upper core keyboard executive) and run.

COPY COMMANDS

Two utility commands have been added to BOSS to facilitate moving ASCII and binary files between devices (particularly when one device is Linc Tape). These commands always copy from the device assigned to LU1 to the device assigned to LU2. The commands are:

	CA	(Copy ASCII)
and	CB	(Copy Binary)

They have no arguments.

Copying is terminated by a status error on the input device:

1) BREAK on the Teletype
2) Lifting the load switch on the High Speed Paper Tape Reader
3) EOF on Linc Tape

Output to Linc Tape must be completed by the write End-of- File command:

WF 2

which is described above.

The copy commands may be used to combine Linc Tape files into a single file by assigning an output file, repeatedly assigning an input file and giving the appropriate copy command, and finally giving the write End-of-File to LU2 after all the input files have been copied into the single output file.

MEMORY EXAMINATION

The memory examination commands have been extended. The set now includes:

OPEN xxxx	Open (and print the contents) of location xxxx.
+	Open the location 2 greater than the current open location (Plus followed by CR).
-	Open the location 2 less than the current open location (Minus followed by CR).
+ xxxx	Open the location xxxx greater than the current open location.
- xxxx	Open the location xxxx less than the current open location.
J	Open the location whose address is the current open location's contents ("J" followed by CR).
*	Print the address of the current open location (Asterisk followed by CR).
RE xxxx	Replace the contents of the current open location with xxxx.

THE SYSTEM TAPE

The System Tape supplied with this document contains a number of useful programs. The second name "IMAGE" indicates a program loaded at X'1000', normally obtained by the GET command. Images which do not load at X'1000' have a second name of the form "@xxxx" where xxxx is the address at which they are loaded. For example, GENLDR @3E00 should be loaded by "GET 0 3E00". Files of second name BIN are binary files. They may be loaded anywhere above the top of BOSS by setting BIAS, assigning and loading. Files of second name IDAL are source programs in Interdata assembly language.

The following are brief descriptions:

LTBOSS	@0	Full LTBOSS with directories and device-independent I/O. Loaded by upper core Keyboard Exec. Also TTY and High Speed Paper Tape I/O.
MINBOS	@0	Minimal version of LTBOSS. No directories. Only RLT and WLT commands. Only Teletype I/O.
OSTIDE	IMAGE	TIDE, the editor, to be loaded by "GET".
OSASM	IMAGE	The assembler .
OSLBLD	IMAGE	The library loader
LTCOPY	IMAGE	Copies Linc Tape on Unit 1 to Linc Tape on Unit 0.
GENLDR	@3A00	General loader. GET 0 3A00.
OSCLUB	BIN	The CLUB debugger in binary. Set BIAS and LOAD.
OSTIDE	BIN	TIDE binary
OSASM	BIN	Assembler binary
OSLBLD	BIN	Library Loader binary

Additional programs may be added to the system tape easily. Other core images can be created by loading (from paper tape) with the LOAD command or one of the loaders, a file created by FENTER, and the contents written out via PUT. Binary files may be added by FENTER, ASSIGN, and CB (Copy Binary) commands.

OPERATING INSTRUCTIONS

To load LTBOSS into a "dead" system, enter the Bootstrap Loader.

50	D500	78	80 lt 1D (lt=Linc Tape Device Addr)
52	047F		
54	4300	34	0000
56	0080	36	0050

Put the system tape on Unit 0 and tension it with the silver marker placed to the right of the head. Start at X'50'. The bootstrap will load the Linc Tape Keyboard Exec. at the top of core (entry X'nE00').

Read in LTBOSS from the system tape, Block 1 by:

Ø,1,~~8~~ A,ØR

Then start BOSS at X'D0' to cause the following initializing actions:

1) A top-of-core search is performed.
2) A buffer for LU1 is created at top - X'200'.
3) A buffer for LU2 is created at top - X'400'.
4) All logical units are assigned to the system device (TTY).

To restart BOSS without this initialization, start at X'124'. If the floating point accumulators are not used, start at X'0' will enter BOSS at this point.

Bk 1· EDFE 1000

[illegible] 2 DDFE 1000

C-6P DDFC

scratch tape

OP 1AC RE 13

OP C98 RE 2302

ignore line feeds

BBE 7F -> A

EXAMPLE

The next few pages are an annotated example of many of the features of LTBOSS. In the example, the Linc Tape Device Code is X'14'.

```
LISTF 1                                 LIST DIRECTORY, UNIT 1
    TEST    FILE  027 0213              STARTS BLOCK 27, 3 BLOCKS LONG
     NEW    TEST  001 0800              STARTS BLOCK  1, 8 BLOCKS LONG

FENTER 1 A F00                          ENTER FILE, START BLOCK A,
FILE NAME                                 F BLOCKS LONG
LONGER FILNAM

LISTF 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
  LONGER  FILNAM  00A 0F00              NEW FILE ENTERED

FENTER 1 0 6A8                          LET LTBOSS FIND SPACE,
FILE NAME                                 7 BLOCKS LONG
FIND SPACE

LISTF 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
  LONGER  FILNAM  00A 0F00
    FIND   SPACE  02A 06A8              NEW FILE ENTERED

FENTER 1 0 100                          ANOTHER FILE, ONE BLOCK
FILE NAME
ONE BLOCK

LISTF 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
  LONGER  FILNAM  00A 0F00
    FIND   SPACE  02A 06A8
     ONE   BLOCK  009 0100              SPACE FOUND BETWEEN FILES

DELETE 1                                DELETE A FILE
FILE NAME
LONGER FILNAM

LISTF 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
    FIND   SPACE  02A 06A8
     ONE   BLOCK  009 0100

RNAME 1                                 RENAME A FILE
FILE NAME
ONE BLOCK
FILE NAME
NEW NAME

LISTF 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
    FIND   SPACE  02A 06A8
     NEW    NAME  009 0100              RENAMED FILE
```

```
CHFILE 1 9 1000                             CHANGE FILE PARAMETERS
FILE NAME                                     NEW LENGTH, X'10' BLOCKS
NEW NAME

LISTF 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
    FIND   SPACE  02A 06A8
     NEW    NAME  009 1000                  LENGTH CHANGED

GET 1 2000                                  READ IN FILE TO BYTES 2000-2427
FILE NAME                                     (2428-25FF ALSO LOADED)
TEST FILE

FE 1 0 200                                  MAKE ENTRY FOR NEW INPUT FILE
FILE NAME
TINY IDAL

AS 102                                      INPUT (LU1) FROM TTY

AS 214 1                                    OUTPUT (LU2) TO LINC TAPE 1
FILE NAME                                     ASSIGN OUTPUT FILE
TINY IDAL

CA                                          COPY ASCII FROM LU1 TO LU2
* TINY PROGRAM TO SAY 'HI'.                 THIS IS INPUT TO FILE
*                                           .
TOP SVC 2,HI                                .
 SVC 3,0 END OF JOB                         .
*                                           .
HI DC 7,12,C'HELLO THERE!'                  .
*                                           .
 END                                        .
I/O ERR A002                                TTY BREAK KEY HIT TO END INPUT

WF 2                                        WRITE EOF TO OUTPUT

AS 114 1                                    USE NEW FILE FOR INPUT (LU1)
FILE NAME
TINY IDAL

AS 202                                      OUTPUT TO TTY

CA                                          COPY ASCII TO VERIFY PROGRAM
* TINY PROGRAM TO SAY 'HI'.
*
TOP SVC 2,HI
 SVC 3,0 END OF JOB
*
HI DC 7,12,C'HELLO THERE!'
*
 END
I/O ERR 9814                                LINC TAPE EOF ENDS INPUT

GET                                         LOAD FILE TAPE 0 TO LOC 1000
FILE NAME                                     (SAME AS 'GET 0 0')
OSASM IMAGE                                   FILE IS OS ASSEMBLER
```

```
AS 114 1                                ASSIGN ASSEMBLER INPUT (LU1)
FILE NAME
TINY IDAL

FE 1 0 200                              CREATE FILE FOR BINARY
FILE NAME
TINY BIN

LI 1
    TEST    FILE 027 0213
     NEW    TEST 001 0800
    TINY    IDAL 031 0032               NOTE LGTH SET BY 'WF 2' ABOVE
    FIND   SPACE 02A 06A8
     NEW    NAME 009 1000
    TINY     BIN 032 0200               NEW FILE FOR OUTPUT

AS 214 1
FILE NAME
TINU BIN                                MISSPELLED NAME
NOT FND
  ?

AS 214 1                                ASSIGN FILE FOR BINARY (LU2)
FILE NAME
TINY BIN
START                                   START ASSEMBLER (AT 1000)
PASS 1
                                                                PAGE   1
  0018R                 END
  HI      0008R
  TOP     0000R
PASS 2
PAUSE

CONTINUE
                                                                PAGE   1
                * TINY PROGRAM TO SAY 'HI'.
                *
  0000R E120    TOP     SVC   2,HI
        0008R
  0004R E130            SVC   3,0               END OF JOB
        0000
                *
  0008R 0007    HI      DC    7,12,C'HELLO THERE!'
        000C
        4845
        4C4C
        4F20
        5448
        4552
        4521
                *
  0018R                 END
                                                                PAGE   2
NO ERRORS
  HI      0008R
  TOP     0000R
EOJ

WF 2                                    WRITE EOF FOR BINARY OUTPUT
```

```
LISTF 1
     TEST    FILE  027 0213
      NEW    TEST  001 0800
     TINY    IDAL  031 0032
     FIND   SPACE  02A 06A8
      NEW    NAME  009 1000
     TINY     BIN  032 0037                NOTE FILE LENGTH SET

AS 114 1                                   ASSIGN BINARY FILE
FILE NAME
TINY BIN

LOAD 1                                     AND LOAD IT
BIAS 1000
END 1018
EOJ

START                                      AND RUN IT
HELLO THERE!                                 WOW!
EOJ

GET                                        GET TIDE EDITOR
FILE NAME
OSTIDE IMAGE

AS 114 1                                   INPUT FILE (LU1)
FILE NAME
TINY IDAL

FE 1 0 200                                 CREATE FILE FOR OUTPUT
FILE NAME
NTINY IDAL

AS 214 1                                   AND ASSIGN IT FOR OUTPUT (LU2)
FILE NAME
NTINY IDAL
```

```
ST                                      START EDITOR
                                          AND EDIT FILE
TIDE
←
A 100
IOERR 9814                              (LINC TAPE EOF)
←
4
   4  SVC 3,0 END OF JOB
←
C
*
 SVC 2,PAUSE
*
 B TOP
*
   6 *                                  TTY BREAK KEY HIT (END INPUT)
←
8
.  8 *
←
I
*
PAUSE DC 1
*
   9 *                                  TTY BREAK AGAIN
←
T
←
P                                       PRINT EDITED FILE
* TINY PROGRAM TO SAY 'HI'.
*
TOP     SVC   2,HI
        SVC   2,PAUSE
        B     TOP
*
HI      DC    7,12,C'HELLO THERE!'
PAUSE   DC    1
*
.       END
←
0                                       OUTPUT IT
←
!                                       LEAVE TIDE
PAUSE

WF 2                                    EOF ON OUTPUT FILE

AS 114 1                                ASSIGN INPUT
FILE NAME
NTINY IDAL

CH 1 32 200                             FIX OLD BIN FOR REUSE
FILE NAME
TINY BIN

AS 214 1                                ASSIGN OUTPUT
FILE NAME
TINY BIN
```

```
GE                                       GET ASSEMBLER AGAIN
FILE NAME
OSASM IMAGE

AS 300                                   LISTING TO NULL DEVICE

ST                                       RUN ASSEMBLER
PASS 1
PASS 2
PAUSE
CO
EOJ

WF 2                                     WRITE EOF ON BINARY

RW 2                                     REWIND BINARY

BIAS 1000                                SET BIAS

LO 2                                     LOAD PROGRAM
BIAS 1000
END 101E
EOJ

FE 1 0 200                               CREATE A FILE TO SAVE IT
FILE NAME
TINY IMAGE

PUT 1 1000                               AND PUT IT OUT
FILE NAME
TINY IMAGE

ST                                       NOW RUN IT
HELLO THERE!
PAUSE
CO                                       AND AGAIN
HELLO THERE!
PAUSE

LISTF 1
    TEST     FILE  027 0213
     NEW     TEST  001 0800
    TINY     IDAL  031 0032
    FIND    SPACE  02A 06A8
     NEW     NAME  009 1000
    TINY      BIN  032 0037
   NTINY     IDAL  019 0037
    TINY    IMAGE  033 0200
```

```
FE 1 0 200                               COMBINE TWO ASCII FILES
FILE NAME
COMBIN IDAL

AS 214 1                                 ASSIGN OUTPUT
FILE NAME
COMBIN IDAL

AS 114 1                                 ASSIGN FIRST INPUT
FILE NAME
TINY IDAL

CA                                       COPY INPUT TO OUTPUT FILE
I/O ERR 9814

AS 114 1                                 MORE INPUT
FILE NAME
NTINY IDAL

CA                                       COPY IT
I/O ERR 9814

WF 2                                     NOW EOF OUTPUT

LI 1
    TEST    FILE  027 0213
     NEW    TEST  001 0800
    TINY    IDAL  031 0032
    FIND   SPACE  02A 06A8
     NEW    NAME  009 1000
    TINY     BIN  032 0037
   NTINY    IDAL  019 0037
    TINY   IMAGE  033 0200
  COMBIN     IDAL 034 0069                AND THERE IT IS
```

LTBOSS

Interdata BOSS for Linc Tape

Program Maintenance Manual

November, 1973

TABLE OF CONTENTS

Page

INTRODUCTION 1

FEATURE ASSIGNMENT 2

ASSEMBLING LTBOSS 4

CHANGES TO BOSS/4B 6

Shrinking 6
Enhancements 7
Linc Tape Use 8

ARDS EXTENSIONS TO BOSS 15

LTBOSS - BOSS for Line Tape
Program Maintenance Manual

Introduction

The information in this document will assist in preparing different versions of LTBOSS and in adding or changing features.

There is a feature assignment section at the beginning of LTBOSS to select various options. The source code is heavily commented to assist in understanding and modification. It is divided into several small files for ease in editing.

Feature Assignment

The features assignment section is at the beginning of LTBOSS. If a feature is desired, the variable should be set to "1" (via EQU); if not desired, to "0".

Feature	Approx. Length (hex bits)	Description
$LOADR	244	The LOAD command.
$PROT	38	Protection features. Checks for legality of START, BIAS, CTOP, and BUFFER addresses.
$COPY	30	The copy commands CA and CB.
$TABSP	14	The SP command which allows tabs to be converted to spaces.
$LFEND	12	Line Feed ends a line as well as Carriage Return.
$LWRCS	A	Lower Case letters are converted to upper case.
$TTYR	64	Teletype Paper Tape Reader (Input on device X'FF').
$TTYP	3E	Teletype Paper Tape Punch (Output on device X'FF').
$ARDS	2A	The ARDS terminal is supported as device X'12'.
$HSPTP	5A	The High Speed Paper Tape Punch and Reader are supported as device X'13'.
$LTRW	164	Linc Tape Commands RLT and WLT are provided.

Feature	Approx. Length (hex bits)	Description
$LTBIO	4CE	Full Line Tape buffered I/O is provided (Length in addition to $LTRW).
CRDRDR	EC	The Card Reader is supported.
LNPRTR	1A	The Line Printer is supported.
MAGTAP	DC	The Magnetic Tape Unit is supported (Add 36 to length if $LTBIO not included).

Lengths are only approximate, as same code is shared by different options.

Assembling LTBOSS

LTBOSS consists of nine separate IDAL source files. The first is the feature assignment (LTBOSS and MINBOS are examples). The files BOSS1 to BOSS8 are parts of the source code. They contain:

BOSS1 - Initialization, Variables, and SVC 1 processor.

BOSS2 - SVC 2 and SVC 3 processors.

BOSS3 - Operator Command processor, Memory examination, and loader.

BOSS4 - ASSIGN, copy, and Linc Tape operator commands.

BOSS5 - Common output and driver routines, TTY and HSPTR support.

BOSS6 - Linc Tape Drivers.

BOSS7 - CRDRDR, LNPRTR, MAGTAP device drivers.

BOSS8 - Linc Tape Utility Routines

Not all files are required. For instance, BOSS7 is needed only if one or more of its devices is requested.

The simplest approach is to create a large file and to combine into it all the required files, then to assemble. For example, with the system tap on Unit 0, a tape with NEWBOS IDAL, the option assignments, on Unit 1, use these commands.

```
FE  1  0  6000
FILE NAME
BOSS  IDAL
```

```
AS  280  1
FILE NAME
BOSS  IDAL

AS  180  1
FILE NAME
NEWBOS  IDAL

CA
I/O   ERR  9880

AS  180  0
FILE NAME                  Repeat for BOSS2 to BOSS8,
BOSS1  IDAL                omitting any not required.

CA
I/O   ERR  9880

WF  2
```

Continue by GETting the assembler, assigning input and output files, and list device. Large versions of BOSS will require 24K bytes of core to assemble. If there is enough core, assign larger buffers especially for input on LU1 for efficiency. Be sure to set CTOP to limit the assembler symbol table. If there is not enough room allowed for the assembler symbol table, there will be "S" (Symbol Table Overflow) errors on pass 1. Assign smaller buffers and increase CTOP (or buy more core if necessary) to correct this.

Changes to BOSS/4B

BOSS for Linc Tape is modified from BOSS/4B supplied by Interdata. This document describes only the modifications.

BOSS has been modified in four ways:

1. Shrinking
2. Enhancements
3. Linc Tape Use
4. Ards Use

Shrinking

In order to add features to BOSS and still keep it under X'1000' bytes, it was necessary to substantially reduce the size of the original BOSS. A good deal of this was provided for by substituting short branches wherever possible. Certain other omissions were made, and they are each noted in the listing. A number of features were made optional, controlled by assembly parameters.

Enhancements

Enhancements to BOSS itself were of two types: to get around certain restrictions and to add new features.

One of the old restrictions which is circumvented is loading the PSW of a program started from BOSS in such a manner that privileged instructions do not trap. Restrictions are also removed on where the bias is set and where programs are loaded.

The most substantial enhancement is to the cell examination area. P, L, J, "," , "+" and "-" have been added or changed for more powerful debugging tools, but the philosophy of operation has not changed.

Linc Tape Use

This section is divided into two parts: a general description of Linc Tape I/O, and a detailed description of some of the more subtle coding of Linc Tape features.

A. Linc Tape I/O

Linc Tape I/O is different from I/O to other devices supported by BOSS in these ways:

1) There are multiple drives on the same device address.

2) There are multiple files on the tape on each drive.

3) Actual I/O is block oriented, rather than character oriented.

Each of these has implications on the extensions to BOSS.

The multiple drives are provided for by an (additional) argument in commands specifying the drive number. This is added to the ASSIGN command, and included in the many new Linc Tape commands.

The multiple files on each tape are provided for by directories written in block 0 of each tape (also in block -1 for compatibility with the Varian monitor). The ASSIGN command requests a file name (as well as requiring the drive number). It reads the directory and then sets up parameters for I/O by SVC's to the file. The commands GETFILE and PUTFILE, which are intended for reading/writing a whole file at once (unbuffered) request a file name and read the directory. Other commands specifically list or modify the directory only — LISTF, RENAME, DELETE, CHANGE, and FENTER.

It is possible to do I/O without using the directories (user must keep track of unit and block assignments) with the RLT and WLT commands.

Since Linc Tape is read/written blocks at a time while SVC I/O is character (or group of characters) oriented, buffering must be performed. Buffers at the top of core are assigned to LU's 1 and 2 when BOSS is initialized. The user may reassign these or assign others with the BUFFER command.

For output, a file should be created (by FENTER) larger than the expected output. This length will be set up as the maximum output length. The Write End of File SVC call or operator command (WF) outputs a remaining partially filled buffer, and enters the actual file length in the directory. Attempts to write more than the file length will cause the EOF error report.

For input, the actual file length from the directory is used. Attempts to read beyond the end result in an EOF. The Rewind SVC call or operator command (RW) resets an input file to its beginning. This is useful between passes of an assembly (SVC call from the assembler), for restarting aborted programs from the beginning (rather than re-doing ASSIGN), etc.

The information stored on Linc Tape should be exactly the stream of characters that would be punched on paper tape with the following exceptions:

1) The output drivers do not write leader (NUL characters) on Linc Tape.

2) The ASCII output drivers store only a single CR at the end of a line (alternately, a single LF if the $LFEND feature is active).

B. Program Notes for Linc Tape Extensions

These are divided into several sections.

1. Varian Linc Tape compatibility extensions

Three compatibility features are included:

1) File directories are written in block -1 as well as block 0.

2) A single LF character (rather than CR or CR-LF) separates ASCII output lines. (Activated by $LFEND feature.)

3) TAB character can be converted to space for the Interdata assembler. The command SPACECHAR specifies what character will be converted to space. Before assembling, the command "SP 9" should be used. At other times, especially when using TIDE to edit, "SP 0" should be in effect to avoid having TAB characters converted. (Activated by $TABSP feature.)

2. Parameters for Buffered LINC Tape I/O

There are nine parameters stored for each logical unit which can be used for LINC Tape I/O. At present, only LU's 1, 2, and 3 may be so used. Others could be added by increasing the number of entries in the "LT Buffer Storage" section of Operator Commands.

1) LTBUFF
2) LTBUFL

Specify the start and length in bytes of the I/O buffers. Initializing BOSS assigns one block buffers at the top of core to LU1 and LU2, and none to LU3.

3) LTUNIT
4) LTSBLK
5) LTNWRD

Are the physical LINC Tape units assigned to the LU, the starting block number on the Tape, and the number of 16-bit words in the file (maximum number for output). These three are set by the ASSIGN command (by looking up the file in the directory of LTUNIT).

6) LTCWRD

The current 16-bit word in the file. Counts from 0 to LTNWRD (input) or actual length (output). Updated only after read or write, not character by character.

7) LTPNT

8) LTPNTN

The current byte pointer in the buffer (counts from 0 to LTPNTN), and the end of the useable information in the buffer (may assume value from 0 to LTBUFL). The buffer is empty if LTPNT=0. LTPNTN will be less than LTBUFL only when reading the last partial buffer, or writing into a file whose remaining length is less than the buffer size.

9) LTDIRP

Pointer to the directory entry for this file. Used by Write End-of-File (WF) which replaces the old (maximum) length with the final value of LTCWRD.

3. LINC Tape Drivers

The following are brief comments on sections of code in the LINC Tape Device Driver section, keyed to statement labels.

LTDVR

Check that this LU is legal for LT, get its buffer location, and use the proper standard read/write subroutine based on the command.

LTWRCH

Subroutine to write a character (equivalent to WRTCHR for character oriented devices). If buffer is empty, sets up parameters (See LTSET), enters character, and checks for buffer full.

LTWRBF

Zero the rest of an LT block, set up args (LTARGS), and write out buffer. Update word counter.

LTRDCH

Subroutine to read a character (equivalent to RDCHAR). If buffer empty, read in bufferful or rest of file and update count. Get character, and if at end of buffer, reset pointer for buffer empty.

LTCMD

For commands recognize only WEOF, BKSP, REWIND. BKSP or REWIND reset LTPNT and LTCWRD to zero.

LTCWF

Write end of file. Write out current partial buffer, read directory, enter LTCWRD as file length, save current SVC return so can use WDIR which does SVC call, restore and return.

LTARGS

Convert LTCWRD to current block and set up args for utilities.

LTSED

Get minimum of buffer size and bytes left in file (watching for EOF) to set LTPNTN limit. Convert to I/O length in blocks.

4. Operator Commands

Most of the operator commands are sufficiently documented in program listings. In general, they call RFILNM to read the directory and a file name and find the file. It returns to either a " found" or "not-found" return location, with the appropriate arguments in the specified places. Some processing is then done on this information. The coding for RLT and WLT is used by the other routines after setting parameters for reading or writing.

The one operator command which requires a bit of additional explanation is FENTER, in the case where the second argument is zero and the routine must find space of a specified length. In this case, four pointers are kept. The approach is to find a space between the end of one file and the start of the next, which is big enough to accommodate the file to be entered. ACØ points to the entry in the directory whose <u>end</u> is being considered; RØ contains the value of that end of file.AC1 points to the directory entry whose start is being considered, and AC2 contains the value of that start address. The FINDSPACE coding goes through the valid entries in the directory and finds their file end address, each time searching the directory whose start is the earliest value after the current <u>end</u>. When it finds this start value for a particular end value, it compares the space between the two to the space requirement specified in the FENTER command. If the space is big enough, the end of the next file in the directory becomes the current end and the start search is reported. If no space big enough is found, an error is reported.

ARDS Extensions to BOSS

The Teletype ASCII drivers are used for ARDS. On ASCII output, ARDS device code is recognized, and if an Erase character (FF) is to be sent, the routine waits for a character (anything) to be typed in before erasing the screen.

There are optional assembly parameters to BOSS to allow conversion of lower case to upper case letters, conversion of tabs to spaces, and to accept Line Feed as well as Carriage Return as a line terminator.

INTERDATA / LINC-TAPE CONTROLLER MOD II

PROGRAMMING INFORMATION

I. DEVICE ADDRESS X'80' JUMPER SELECTABLE - SEE LOGIC SCHEMATIC

II. INT (INITIALIZE SWITCH ; POWER UP SEQUENCE)

STOPS ALL DRIVES, SELECTS DRIVE φ
DISABLES INTERRUPT, CLEARS EX & ERR

III. COMMANDS :

BITS ~	8	9	10	11	12	13	14	15
* MOVE FORWARD	Ø	Ø	φ	1	φ	1	φ	1
* MOVE REVERSE	φ	φ	φ	1	φ	1	1	φ
* STOP	φ	φ	φ	1	φ	φ	φ	φ
* BOOT	φ	φ	φ	1	1	1	φ	1
MODE READ	x	φ	1	φ	x	x	φ	φ
WRITE	x	φ	1	φ	x	x	1	φ
CHECK	x	φ	1	φ	x	x	φ	1
DRIVE SELECT nnnn = drive φ → F	φ	1	φ	φ	n	n	n	n
INTERRUPT ENABLE	1	φ	x	φ	1	x	x	x
DISABLE	1	φ	x	φ	φ	x	x	x

* ALL MOTION CONTROL COMMANDS CLEAR QUEUED INTERRUP
CLEAR STATUS BITS ERR & EX (next page),
AND SET MODE READ

TITLE INTERDATA/LINC MOD II PROGRAMMING

dm

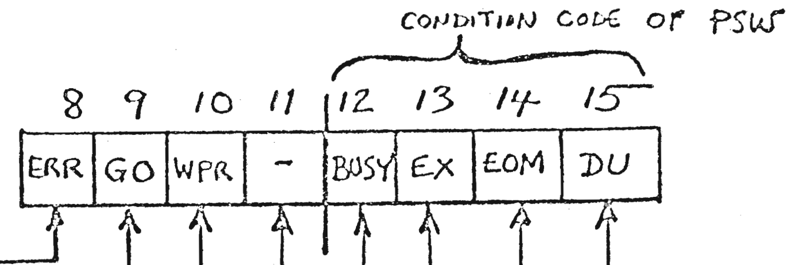
CHK

DWG A-10370

SHT 1 OF 6

COMPUTER OPERATIONS, INC.
BELTSVILLE, MARYLAND

IV. STATUS BYTE :



CHECKSUM ERROR
cleared by INT
and MOTION command

TARGET BLOCK FOUND
IN FORWARD DIRECTION
DATA PHASE ACTIVE.

SELECTED
DRIVE IS WRITE-PROTECTED

NOT USED (Ø)

SELECTED
DRIVE NOT
IN TENSION,
OR LINCTAPE
POWER SUPPLY
DETECTED
POWER FAIL,
OR ATTEMPT
TO WRITE ON
PROTECTED DRIVE.
ALL FORCE
TAPE STOP.

BUSY
WAIT FOR THIS
TO DROP BEFORE
PROCEEDING.

EITHER BIT8 (ERR)
OR LATE STATUS = 1.
LATE MEANS THERE WAS NO
I/O TRANSFER QUICKLY ENOUGH.
LATE FORCES DRIVE STOP.

EOM ⇒ TARGET NOT FOUND
BLOCK SEARCH PHASE

V. INTERRUPT ~

ATTENTION SIGNAL GENERATED IF INTERRUPT CONDITION QUEUED AND INTERRUPT ENABLED.

ATTENTION SIGNAL REMOVED WHEN "INTERRUPT ACKNOWLEDGE" GENERATGD BY COMPUTER, WHEN INTERRUPT IS DISABLED (QUEUED CONDITION IS REMEMBERED), OR WHEN A MOTION COMMAND IS ISSUED.

THE CONDITIONS WHICH QUEUE INTERRUPTS ARE –

1. BUSY STATUS CHANGING FROM 1 TO Ø.
2. MOTION OF TAPE STOPPING.

NOTE THAT IF MOTION STOPS BECAUSE OF MOTION STOP COMMAND OR INITIALIZE, THEN INTGRRUPT IS NOT QUEUED.

VI. OPERATING SEQUENCE:

A. INITIATION OF LINCTAPE MOTION

1. STOP COMMAND X'1Ø'
2. SELECT DRIVE X'4n'
3. MOVE FWD or REV X'15' X'16'
4. SET MODE READ X'2Ø'
   WRITE X'22'
   CHECK X'21'
5. WRITE TARGET BLOCK # (2 BYTES OR HALFWORD)
6. CHECK STATUS. MUST = xxxx,1000 binary
7. ENABLE interrupt X'88', start selector channel,
   set up automatic I/O, or enter STATUS loop

B. SERVICE (selector channel interrupt, LINC interrupt, STATUS = $\overline{\text{BUSY}}$)

1. DISABLE interrupt (only if PSW bit 1 = 1; multilevel priority)
2. IF STATUS = xxxx,xxx1 ⇒ DEVICE UNAVAILABLE ERROR. Restart.
3. IF STATUS = Øxxx,x1xx ⇒ FINISHED, IF ALL DATA ALREADY DONE
   LATE ERROR, IF NOT ALREADY DONE.
   = 1xxx,x1xx ⇒ CHECKSUM ERROR. Restart
4. IF STATUS = xxxx,xx1x ⇒ EOM - Block search required.
   WITHIN 60 milliseconds, you must:
   ⓐ READ block # (2 bytes/1 halfword)
   ⓑ Compare to TARGET. If more than 1 block below target and moving REV, then MOVE FORWARD. If above target and moving FWD, MOVE REVERSE. Do not issue MOTION command if change of direction is unnecessary. (cont'd)

TITLE
INTERDATA/LINC MOD II PROGRAMMING

dm
CHK'D

DWG A-1037Ø

SHT 6 OF 6

COMPUTER OPERATIONS, INC.
BELTSVILLE, MARYLAND

ⓒ If MOTION command was issued, re-establish MODE [READ/WRITE/CHECK] It doesn't hurt to SET MODE anyway

ⓓ go to step A.6 above.

5. IF STATUS = xxxx, xxØx then GO=1, and you are in the data phase. The TARGET block has been found. BUSY WILL DROP once for every halfword to be transferred.

For READ & WRITE modes, this will occur 256 times per block, with a required response time of about 240 microseconds.

For CHECK, BUSY drops once per block, at the beginning of a block; you must Read the block # (2 bytes) within 60 milliseconds. Status bit EX is valid for checksum computation on previous block.

ⓐ Do the I/O read or write (always 2 bytes) unless all data required has already been transferred, in which case, do nothing. The latter will cause a LATE (see B.3)

ⓑ go to step A.6

VII. AUTOLOAD OPERATION :

WITH A PROPERLY WRITTEN LINCTAPE, THE STANDARD "50 SEQUENCE" CAN BE USED TO BOOTSTRAP ANY DESIRED PROGRAM.

THE FOLLOWING REQUIREMENTS HOLD:

1. TAPE ON DRIVE Ø MUST BE IN TENSION AND PARKED SO THAT THE HEAD IS BETWEEN TAPE BEGINNING AND FOIL MARKER IE, BELOW FIRST BLOCK.

2. LOCATION X'52' SHOULD BE SET TO THE HIGHEST ADDRESS LOADED BY BOOT. IN THE CASE OF <u>INTERDATA T1</u> TAPES, THAT'S X'47F'.

3. LOCATION X'78' MUST CONTAIN LINC DEVICE ADDRESS AND BOOT COMMAND BYTES = X'8Ø1D'.

4. <u>INT</u> SWITCH MUST BE PRESSED, SO THAT DRIVE Ø IS SELECTED.

* ±

```
UNPAK   LH    ADR,0(RTN1)
        LHI   FBA,3(ADR)
FPTZ    LIS   CHAR,15
        NHR   CHAR,RØ
        SRLS  RØ,4
        SIS   CHAR,A
        BMS
        AIS   CHAR,7
        AHI   CHAR,X'3A'
        STB   CHAR,0(FBA)
        SIS   FBA,1
        CLHR  FBA,ADR
        BNLS  FPTZ
        B     2(RTN1)
```

```
ONCRLF  LHI   0,X'4160'
OCSTO   STH   0,NOCRLF
        BR    RTN1
OFCRLF  LHI   0,X'303'
        BS    OCSTO
        ORG
        DC    ONCRLF
        DC    OFCRLF
```

```
SVC4    STH   0 RSAVE
        LIS   0,B
        STB   0,DOCRLF-1
        LH    0,RSAVE
        SVC1
```

```
SVCRTN  LIS   0,1
        STB   0,DOCRLF-1
        LH    0,RSAVE
        LPSW  X'96'
```

~~VIBN # 4~~

in TTYDVR

```
        BS
        BS    DOCRLF
DOCRLF  BAL
```