LINC TAPE OPERATING SYSTEM FOR NOVA

PART I

Definition: The LINC Tape Operating System (LTOS) is a Nova computer program for managing files, by name, on LINC Tape.

Purpose: LTOS is designed primarily for assisting program development. To this end, it supports the editor, assembler, and object loaders, subject to the core requirements of each program. LTOS does not handle non-LINC Tape I/O operations for the system (aside from operator interchanges on TTY), it does not allocate memory, and it can simultaneously access only one input file and one output file on LINC Tape. LTOS can be used by any program, but because it has no absolute control over the computer (interrupts, particularly), the user program is entirely responsible for safe operation of the system. It is expected that the primary use of LTOS by a program will be for named file overlay and storage.

The advantages of such a limited operating system are as follows:

a. LTOS will occupy a very small amount of core, interfering as little as possible with the user program, while still providing basic functions. This space is approximately 1800_{10} words.

b. The absence of I/O control permits more flexible use of real-time devices, with only the LINC Tape hardware limitations common to all LTOS uses.

c. Sophisticated support is given the Extended Assembler, relocatable loader, and COI text editor, so that the programmer can concentrate on his application, rather than the details of operating such software.

PART II

LTOS Features

LTOS is basically a core-resident file handling system. The files are stored on one or more LINC Tapes, each with a directory specifying its files. The directory is automatically maintained by LTOS. There are three types of files. Two of these have special formats for use by the support systems programs; the third file type is unformatted, permitting external definition of its composition.

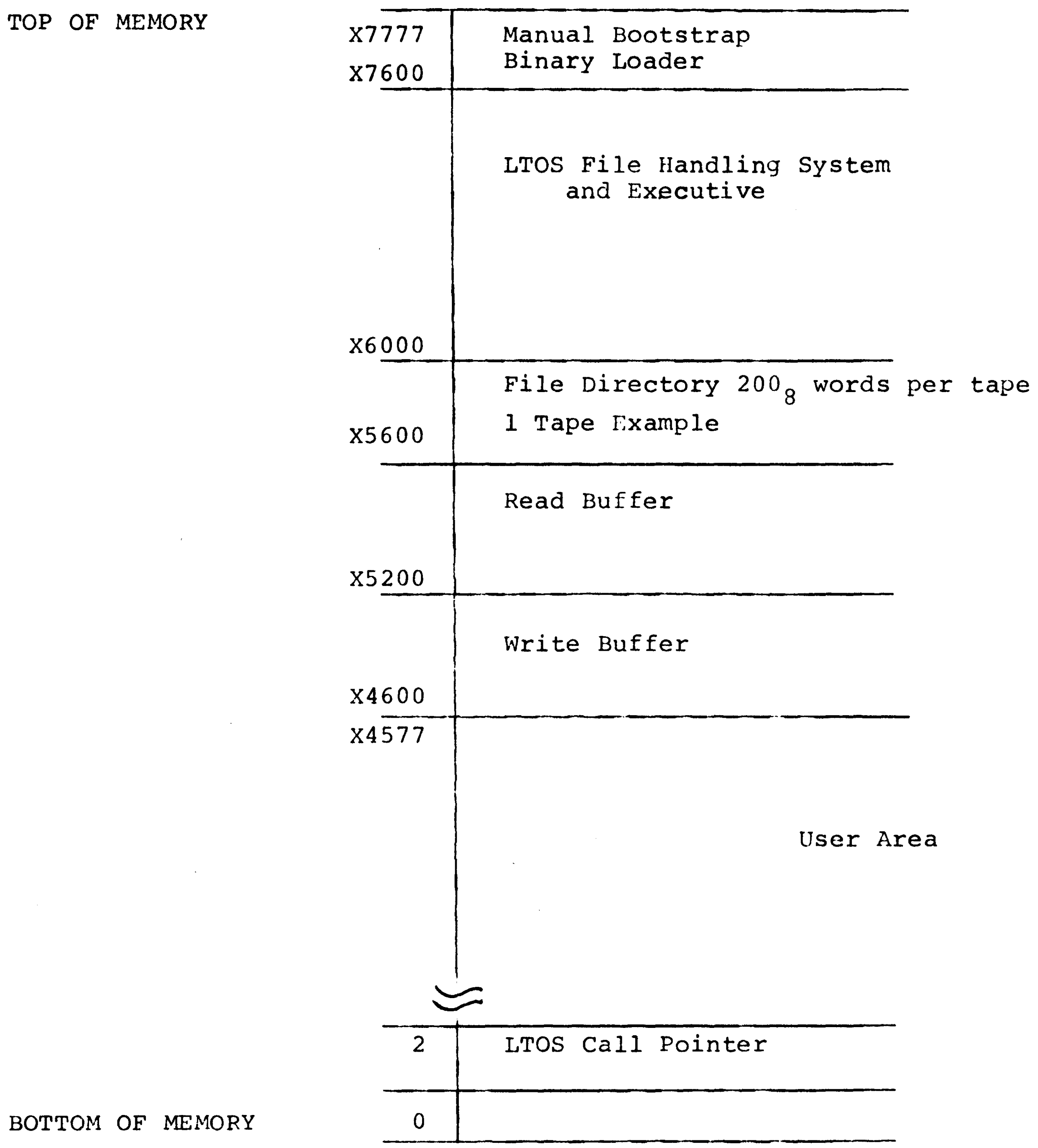
The file handling system can be used in one of two ways. The first is by keyboard executive, which gains control whenever LTOS is begun, or when the program "returns" to the system. The keyboard commands allow the operator to create, delete, and assign files by their names, and to load a program file for execution.

The second use of the file handling system is through calls from a program running elsewhere in core. In addition to the executive commands, a program call can open and sequentially read or write to any named file on the system. There are two "channels"; two files may be open at one, one to read, one to write. Calls are made by linkage to LTOS through a pointer in location 2. Command codes and arguments are stored in core following the call (a JSR instruction), and results of the command are returned in the AC's.

Memory Allocations

LTOS is normally located near the top of core, leaving 200_8 locations at the top for the manual bootstrap and small programs. The low end of LTOS is occupied by two LINC Tape buffers (512 words total) which can be utilized by some programs. See map on next page.

Typical LTOS Core Map (all adresses in octal)



X = 0 for 4K
X = 1 for 8K
.....etc.

PART III

Files

A file is treated as a named, contiguous string of bytes, beginning at a directory-specified block on one of the system tapes. At some time, every file in the system must have a name assigned to it by the user. Duplications in file names are not allowed, unless the files are identical. A file must never occupy more than one LINC Tape, and if it won't fit on any tape while being built, it is lost and must be rebuilt. After a file is built, it is closed, and the LINC Tape space it occupies is remembered by the file handling system.

File Types

a. TEXT file

A string of ASCII characters, terminated by an ASCII End of Transmission character (Control-D). This file is packed left-right on tape, starting at a block boundary, and zero-filled to the end of the last block boundary. It is used by the COI text editor and language processors.

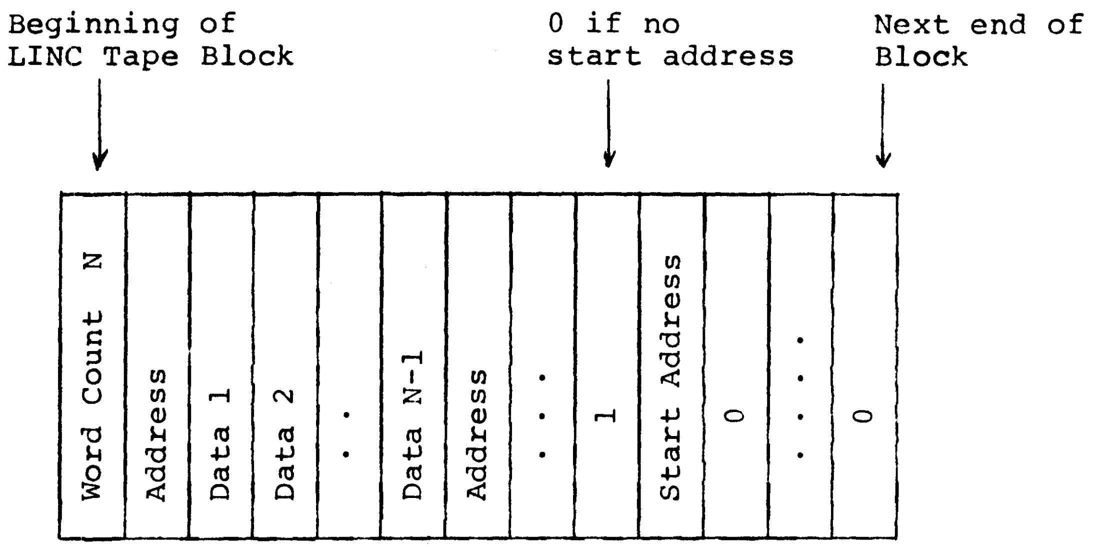
b. SAVE file

A binary image of core, including addresses. This type file is created by the LTOS command "SAVE".

c. UNFORMATTED file

A string of bytes, beginning at a block boundary, bytes packed left-right on tape. Bytes may be 0--377 octal. There is no "end of file"; something buried in the file must indicate end to the user. For example, a relocatable assembly will produce this type file. "End" is signalled by the assembler-relocatable loader convention of a data structure called a START BLOCK.

Word Format of SAVE File (2 bytes packed left-right form a word)



If "address" is negative, it is added to the size of core, and thereby, specifies a position relative to the top of memory.

PART IV

LTOS Executive Mode

The LTOS executive is a special file-handling program which is entered whenever the computer operator performs a PROGRAM LOAD or bootstrap with an LTOS system tape, or when control is passed to LTOS from a running program.

The executive permits the operator to control LTOS through the Teletype. The executive indicates its readiness to accept commands by typing a ready message, after which it waits for a command, executes the command, and again types the ready message. A command is actually the file name of the command-performing program, stored on tape as a SAVE file, or it may be the name of a file resident within the LTOS system.

Once a command program is entered, it will probably engage in additional dialogue with the operator, asking for file names, typing out responses. Until it returns to the executive, it behaves like any other user program; it can make calls to LTOS; it is totally responsible for operation of the computer. When finished, it may pass control back to the LTOS executive, ready for another command.

Initially, we shall supply the following commands as core resident LTOS programs. Command arguments are noted where applicable. Tape resident LTOS programs will be supplied as tape files and are described separately.

N.B. ↵ symbol represents carriage return, capitals computer response.

a.
```
save↵
NEW FILE, TAPE?  name,t↵
ADDRESSES?  first, last, start↵
```

Function - create a savefile from the area of core "first" thru "last" inclusive, on drive "t". Call the file "name". A start address may be defined as shown, or if not desired, type carriage return after "last". All numbers in octal, modulo 200000. Refer to specifications for savefiles.

b.
```
rubout↵
VICTIM?  name↵
```

Function - delete file "name" by substituting rubouts for the first occurrence of "name" in the directory. Search starts with tape 0. Tape space is not changed. Tape directory is not changed.

PART V

<u>Program Calls to LTOS</u>

Calls to LTOS are accomplished by a JSR@2, assembled into the user's program. Location 2 contains a pointer to the LTOS interpreter loaded by the LTOS executive. The 3 words following the call are a command code, and 2 arguments. LTOS normally returns to the core location 5 beyond the call; abnormal returns are to the location 4 beyond the call. A call might look like this in memory:

```
        JSR@2     ;Call instruction
        15        ;Command code
        ARG1      ;Arguments
        ARG2
ABN:    JMP PUNT  ;Error return, error code in AC1
OK:               ;Normal completion of call
```

Appendix A contains details on program calls to LTOS.

LTOS OPERATING PROCEDURE

Minimum hardware: 4K, Teletype, LINC Tape, Nova Series CPU

To manually enter the LTOS keyboard executive:

(1) System non-resident or destroyed

Boot the system. You must have an LTOS tape loaded on each drive, and the tape on drive 0 must be one specifically generated for your configuration. Follow LINC Tape boot procedure. The manual LTOS start address, see (2) below, will be printed, followed by the -EXEC-? ready message.

(2) System previously loaded

"RESET"
Set console data switches to manual start address (see (1) above).
"START"
-EXEC-? message should be typed

NOTE: (1) assumes nothing in core. Therefore, any temporary data (the current file directory, for example) is destroyed by the boot. The operating system code is loaded from drive 0, and the file directory is collected from each system tape.
(2) does not affect the file directory or LTOS code. If you aren't sure what to do after a program has run, or if you wish to abort a program, try (2) above. If that fails, try (1).

Whenever the LTOS keyboard executive is begun (as after a bootstrap), a ready message is typed:

```
-EXEC-?
```

The executive expects the operator to key in the name of a file which may be either a resident program file, or a save file. If the name is followed by a comma, the named file is loaded, control returns to the executive, and the ready message, -EXEC-?, is repeated. If the name is followed by a carriage return, the file is loaded and executed as a program. Control of the machine is then in the hands of the named program. If the program doesn't harm the resident part of LTOS, then control can be passed back to the executive by a program call, or by manual console operation. Most of the programs supplied in support of LTOS have a means of returning to the keyboard executive, just as most of them make calls to LTOS for system functions.

The details of operator response to -EXEC-? are as follows:

The response must be a file name of 6 printable letters or less.

An escape causes the -EXEC-? to repeat. Other control characters (except carriage return) and spaces are ignored. The operator can use line feeds and spaces to format his response.

A delete (rubout) cancels the response, and causes the ? mark to be repeated.

If more than 6 printable characters are typed, the computer cancels and repeats the ? mark.

The name must be terminated by comma or carriage return.

The name is then checked against the current file directory.
If it is not found, the message "NO GO" is typed and then the message is repeated.
If the name is found, it must be the name of either a "resident program" or a "save file". If not the NO GO message is typed.

An attempt to load the named program is made. Should the program contain data located within the area of the resident LTOS, or outside the limits of core, loading will be aborted, and the message NO GO typed.

If the file you named meets all the above requirements, it will load properly. In order to execute, you must have typed carriage return after the name and the program must have a start address specified within the file. If you typed comma, execution will not take place, regardless of a start address. This is useful if you want to have more than one program resident, assuming they don't overlap.

If the program loads, but does not execute, the LTOS ready message is typed.

If the program executes, what happens next depends entirely on it alone.

ERROR conditions

The only computer halts in LTOS and the stock utilities (FILES, TAPES, etc.) are for LINC Tape errors, or fatal operator errors. If the computer halts, (RUN lamp goes out) press console "CONTINUE".

NOTE: You NEVER need "RESET" after the halts discussed in this section.

If the halt is repeated, "EXAMINE" AC1.

The error code is displayed in the console "DATA" lamps.

The most frequently encountered error is trying to write on a protected device. Push the red light on the protected drive and press "CONTINUE".

Consult the following table for each code:

AC1 (Octal)	Description
01	Block Checksum Error AC0 - Bad Block. Try a few more times, then clean tape and head of drive which moves. If that fails, tape or drive is not good, and you must abort. Try a manual restart of LTOS.

AC1 (Octal)	Description
02	<u>Bad Tape Block</u> Same procedure as above.
04	<u>Illegal Block</u> A software error. Retry will do no good. Restart LTOS.
10	<u>Drive Status Error</u> Are the tapes in tension? Is the PROTECT lamp on when trying to write? If so, correct and retry.

If the AC1 code is none of the above, and you are attempting a boot of the system, chances are you have mounted a tape not intended for your computer. Start the boot procedure again. Should that not be the case, you have a system failure. "RESET" and start LTOS at the manual entry address.

There is one LINC Tape error condition which can be caused by power failure or operator error, and does not result in a HALT. If the operator presses the "REWIND" button on a moving tape drive, or if the tape runs off either reel, the program may become "hung", even if tape tension is restored. This condition is best detected by a Real-Time Clock timeout, but is also obvious to an operator. LINC Tapes will not regain tension after a power failure, and if one was in use at the time of power loss, the "hung" condition will result if the program is re-entered at the failure point. The simplest procedure in all "hung" situations is a manual restart of LTOS, with the job aborted. If recovery is essential, then the "hung" condition may be made to cause a DRIVE STATUS ERROR HALT (AC1 = 10) by restarting at location S+110 octal, S = manual restart address. This permits the unsuccessful LINC operation to be retried when the tapes have been properly loaded.

Automatic Operation of LTOS

Normally, all LTOS commands are entered at the Teletype keyboard. However, the Teletype paper tape reader may be used just like the keyboard, because LTOS "starts" the Teletype. If you make up a paper tape of the command characters you would use to drive an LTOS run, you can use this tape to automatically perform the entire job. This feature is obviously limited to those programs which don't use Teletype paper tape input; and you must place the reader switch in "STOP" or "FREE" if there is a tape in the reader which is _not_ a command tape.

The following sections describe the operation and use of COI-supplied LTOS programs. All of them run on any configuration of the operating system, and most have plentiful operator messages, so all that need be done to invoke one of these programs is to type its name, followed by a carriage return, in response to -EXEC-?. The set of programs may be added to or modified without notice.

TAPES Program

Memory location - top 200_8 words (except for manual boot)

To get "off" the system:

You should never just stop the computer and remove your tapes after a run. The file directory is core-resident. If any changes in the files have occurred between the time the tapes were loaded and now, you must update the tape-resident file directories. This is accomplished by executing TAPES. So, at the end of a session, return to the LTOS keyboard executive, and type TAPES. Each tape will have its own directory written (a halt will signal any tape status you must correct) and then the message "CHANGE TAPES, HIT ANY KEY"will be typed. At this point you may remove tapes and shut down the computer, or you may get back on the air by mounting the desired file tapes, and striking a key on the Teletype, as requested. Tape 0 need not be the configuration-dependent one required for a boot. The system is not changed by the new directories; just the core-resident file directory.

FILES Program

Memory location - lower core

FILES is a savefile supplied on most LTOS tapes. It lists the core-resident file directory on the Teletype printer. When started, FILES always lists the names of RESIDENT PROGRAMS under that heading. Resident programs are those resident within LTOS itself; they are not loaded from LINC Tape. Next, FILES asks for the tape whose directory you wish listed, with the message:

```
            TYPE (ESC) or TAPE # (CR)
```

You type ESC to return to the keyboard executive or you type an octal number representing the tape, then a carriage return.

An invalid response causes repetition of the message above. The number you type is interpreted modulo - 200000 octal, so that 7770000001 = 1. The file directory is listed as a columnized table. The first column is a 3-digit number representing the octal block number of the beginning of a file. The second column is a 2-letter file type code either "SF" for savefile, or an octal representation of the type code. Remember, only savefiles can be loaded by the LTOS executive. The third column is the file name.

The last entry in the list has no file name. This is the "end of tape" block, which marks the beginning of scratch area on the LINC Tape.

There is one special type of entry, the RUBOUT. A file which has been deleted but which still occupies space on tape is marked by the symbol (--) in the file name column, and its type code is always 00.

Deletion of Files

Two programs supplied with LTOS tapes are used to delete files under operator command. The RUBOUT program is used to "erase" a file from the directory. It asks for the victim file name, then acknowledges erasure with a message. If the file doesn't exist, or if it can't be deleted (resident) negative acknowledgement is given. In either case, return is made to the executive.

RUBOUT does nothing to the file on tape, and it doesn't "free" the space occupied by the victim file. It merely removes the file's name from the directory in <u>core</u>. RUBOUT is LTOS-resident.

The SQUASH program must be used to actually compact a tape containing "erased" files. SQUASH is most efficient when several files have been erased. The best time to run SQUASH is at the end of a session, as long as you don't run out of LINC Tape before then. SQUASH types the following message:

```
TYPE (ESC) or TAPE #(CR):
```

The operator should type the octal number of the tape he wishes SQUASHed followed by a carriage return. When finished, the message DONE! is typed, and the tape request repeated. You may SQUASH another tape or type ESC to return to keyboard executive. An invalid operator response causes repetition of the request.

The SQUASH process consists of four phases, all of which _must_ be complete:

1. Compact the file directory in core.
2. Relocate those files which must be shuffled down toward block zero on tape.
3. Update directory on tape from core directory.
4. Acknowledge completion with message.

DO NOT abort a SQUASH, once initiated, unless you care little for your files! SQUASH utilizes all of memory.

Addition of files

The ATTACH program can be used to add unformatted files to LTOS from an external source (paper tape or non-LTOS LINC tape). For paper tape input, ATTACH has a timeout to identify the end of file. For LINC input, the program requests the number of blocks, and does a straight transfer. All operations are prompted by messages on the Teletype. This program is expected to have application mostly for those users who already have binary tapes or non-LTOS LINC Tapes, and who wish to incorporate them into LTOS. Source-language tapes in ASCII should be converted to LTOS files using the EDITOR, not ATTACH. ATTACH utilizes all of memory outside of LTOS.

Dumping a File

If you wish to list an ASCII file, use the EDITOR, but if you want to create a paper tape image of a file (as the binary output of an assembly) use the DUMP program. Output devices can be specified by the operator (line printer, high speed punch, TTY punch, or TTY printer). A print function adds a linefeed character after each carriage return; a punch function adds leader and trailer nulls to an exact image of the file. DUMP occupies the first 500_8 words of core.

Keyboard Executive

KBEX is a program used to manipulate LINC Tape on a block basis. The operator types octal numbers and a function letter to read, write, or check tape. The program occupies the upper 200_8 words of core (except for the manual bootstrap).

KBEX types an asterisk whenever ready for a command. Operator response must be of the general form: (ESC)-escape to LTOS executive; or: aaa,bbb,ccc,dK (commas separate arguments)

where:	aaa	is an octal core address, modulo 200000. If negative, tape will start reverse, and the actual core address will be the 1's complement of aaa
	bbb	is the first LINC Tape block number, modulo 200000
	ccc	is the block count, octal
	d	is the drive number
	K	is the R, W, C for read, write, check respectively.

An empty numerical entry is interpreted as 0.

```
EXAMPLE:  *1000,,2,1R - read 2 blocks, starting at block 0,
                        from drive 1, into location 1000.
```

CAUTION: KBEX has no built-in protection against overwriting LTOS

You may freely use core from location 0 up to LTOS (usually approximately 4000_8 words from top of memory).

KBEX makes LTOS calls independently of location 2, thereby, permitting use of lower core as a buffer.

Application: KBEX is the fastest, simplest means of saving and loading core images of data, copying tapes, and checking tapes.

Errors: LINC Tape errors (except checksum) cause the message ? to be typed. These errors are:

Illegal block # requested
Drive not tensioned
Drive write-protected
Wrong number of words in block

Checksum errors cause the command to be tried 3 times before typing the error message.

<u>LTOS Assembler</u> (Requires at least 8K)

The LTOS Assembler, ASSEM, is based on the Data General Extended Assembler. Only the significant differences are described here.

1. ASSEM occupies memory from 0 to 7500_8, and uses core up to the base of LTOS for symbol storage. It calls upon LTOS for all LINC Tape I/O.

2. The supplied version has defined all the LTOS program call symbols, and the symbol "LINC" for device 74. All the symbols for DOS calls have been purged. All instruction definitions (including floating point) and pseudo-ops have been retained. In addition, a correction for definition of the absolute floating point interpreter call,"FINI", has been made.

3. All input is from an LTOS file. Characters are stripped to UASCII-7 without parity checking. In pass 1, the symbol definition pass, the operator specifies the input file in response to the question SOURCE FILE?.

4. If an .EOT is encountered on input, the assembler prints the .EOT line on the Teletype, then requests another SOURCE FILE?. There is no halt; assembly proceeds with the next file. ASSEM can store up to 20 source file names in a single pass.

5. Binary is always written on LINC Tape. In a binary pass, the assembler types the message:

   BIN FILE, TAPE?

   The operator must specify a new file name followed by a carriage return, or followed by a comma, a drive number, and a carriage return. Tape 0 is used if tape is not specified. If the assembly output is relocatable, the assembler next asks whether or not it should output local symbols with the question LOCAL SYMBOLS Y OR N?. The operator should type Y or N as desired. Binary output is an exact image of the normal paper tape binary produced by the DGC Extended Assembler.

6. List output is specified at the time the assembler is started. It may be Teletype 33, Teletype 35 (37), or Data Products line printer. The difference between Teletypes is tab and form feed simulation. Error lines are always printed on the Teletype.

7. "Mode" is requested at the beginning of each pass, just as for other assemblers. The modes are different, however:

0	Escape to LTOS executive
1	Pass 1 symbol definition
2	Binary pass
3	List pass
4	Binary and List
5	Type errors only

8. The Nova interrupt is not used, since LINC Tape operation is non-simultaneous with other I/O devices.

9. In order to load binary output:
   a. Use BLDR for absolute binary out.
   b. Use RLDR for relocatable out.
   c. Use DUMP to punch paper tape of the binary for subsequent load by any of the standard Nova software.

LTOS Binary Loader

BLDR is a LINC Tape version of the normal absolute binary loader. It occupies the top 200_8 locations of core, excluding the bootstrap, and has the same start address as the paper tape loader (X7777).

BLDR first requests the name of the binary file to be loaded by typing

?

The operator should type the file name, then carriage return. If the file doesn't exist, or if a rubout is typed, the ? repeats. Loading begins once the file is properly opened; the file is treated as if it were a paper tape image. If there are any errors, the message

ERR

is typed, and return is made to LTOS executive. If there is no error, but no start address, return is made to LTOS. If there is a start address, the loaded program is started.

If desirable, a savefile can be created to permit rapid load, by name, of the new program. If the code is position-independent, move it to the desired area of core before you SAVE it. SAVE does not overwrite any of core outside of LTOS.

Restrictions:

1. You cannot load from a "resident" file, for example "SAVE". This will cause an error, not fatal.

2. You cannot store data in any part of LTOS, in any part of BLDR, or at any non-existent core address. This will cause termination of the load.

3. It is impossible to load into location 2, as the loader will protect the LTOS pointer there. Attempting to load location 2 will not cause an error, however.

LTOS Relocatable Loader

RLDR is a modification of DGC Extended Relocatable Loader. The operational details are similar, so only the differences are explained.

1. Corrections were made to permit DEBUG III to link to user symbols, through the User Status Table.

2. The only form of binary input is LINC Tape.

3. "SAFE" is normally set just below the LTOS input buffer, i.e., the LTOS output buffer is overwritten. The first thing RLDR does when loaded is type

   ```
                    EXTRA SAFE =
   ```

   The operator should type carriage return if he doesn't need space between the symbol table and LTOS, or the octal number of extra words saved, carriage return.

4. The loader then types the familiar * to indicate mode request. The modes are as follows:

	0	Escape to LTOS executive
	1	Load relocatable binary. This response will be followed by a ?, after which the operator should type the file name, then carriage return. Typing (ESC) or the name of a non-existent file causes another mode request. Typing (RUBOUT) or more than 6 letters causes repetition of the ? mark. Once opened properly, the load will proceed under all the normal procedures. The file is treated just like a paper tape image. All the normal loader messages apply. In addition, uncorrectable LINC errors cause the message "RE" and a return to mode request.
	2	Illegal - ? will be typed, then the mode request asterisk.
Same as DGC Loader	3	Set NMAX from console switches
	4	Complement symbols switch
	5	List memory symbols
	6	List global symbols
	7	Initialize
	8	Terminate

5. No interrupts are used.

6. Pseudo-zero is 2334_8, and SAFE is typically 3000_8 from top of core. MEM -(2334 + 3000), or approximately 12400_8 words in an 8K machine for loaded data and symbols.

7. As for the DGC Extended Relocatable Loader, NMAX begins at 1000_8, and locations 400-777 are reserved for run-time OS, if any. If not using a run-time operating system, the area 410-777 can be loaded without error.

8. After mode 8, it is appropriate to manually return to LTOS, and to SAVE the loaded program as a savefile, using the loader memory map as a guide.

9. If repetitively loading a large number of relocatable files, it may be helpful to prepare a paper tape of the keyboard commands, and to use the Teletype reader to automatically run the job. RLDR ignores nulls in such a tape, as does LTOS, and both "start" the reader. See the paragraph on "Automatic Operation" in the LTOS Operating Procedures.

DEBUG III

DEBUG III is supplied as a relocatable binary file name, DBUG·R. It is identical to the DGC DEBUG III for stand-alone operating system with following modifications:

1. The symbol "LINC" for device 74 is added.

2. The linkage of user symbols through the User Status Table (400-410) has been fixed. This DEBUG will link up to programs loaded using RLDR, if local symbols were included in the binary assembly, and if the "symbols switch" was set during load.

Caution: Absolute locations $10\text{-}17_8$ are overwritten with the breakpoint addresses whenever the $G or $P or $R are executed.

COPY program

Memory location - lower core

COPY is a dual purpose program. It can be used to:

1. Copy a named file to any LTOS tape
2. Move a named file to any tape.

The second function is the same as the first, but with the original file deleted. COPY is very useful. With it, you can arbitrarily rearrange the files on your tapes, in order to separate types, or to minimize access time. You can create a new tape by copying to it just those files you want.

Procedure - COPY is invoked by the LTOS executive. It contains several messages, and its use is largely self-explanatory. A copy is accomplished by responding N to the question DELETE SOURCE?(Y,N,). A move is accomplished by responding Y. The program will repeat or escape to LTOS executive depending on the response to the question CONTINUE?(Y,N).

BINLDR

BINLDR is a savefile equivalent to the Data General paper tape binary loader. It can be invoked in order to restore the binary loader when an LTOS run is complete, and you wish to revert to paper tape.

BINLDR does not load the area reserved for a manual bootstrap loader. It is loaded at the top of core, regardless of machine size.

RENAME program

Memory location - lower core

RENAME, as implied, is used to change the name of an LTOS file. This function is useful for assigning a commonly-used name to a new version of a file, as after debugging a newly edited source program.

RENAME, when executed, asks for the current (old) file name. When the operator has typed the file name followed by carriage return, RENAME lists the tape and block number of the first occurrence of the file. You can verify that this is the file you wish to rename by consulting a FILES listing. Resident files cannot be RENAMED.

RENAME then asks for the new file name. This must be unique and if accepted, the program types DONE! and returns to the LTOS executive.

BASIC (requires at least 8K)

A. Data General Single-User BASIC has been modified so it can be loaded as a savefile named BASIC. Since both LTOS and BASIC must be core resident in order to load, you need 8K. In addition, LTOS is protected when BASIC is running, so return to the LTOS executive can be made (manually) without bootstrapping. There is no LINC I/O in BASIC, so no run-time support is provided by LTOS. Refer to D.G.C. manual 093-000042 for information on BASIC programming.

B. Modifications to core image of binary tape D.G.C. 91-000018:

LOC	WAS	IS	
7414	ADCZL 3,3	SYSTM	;Replace core-sizing
7415	LDA 2 7452	.ADDR	;Routine with LTOS
7416	LDA 0,0,2	1000	;Call to Determine
7417	STA 3,0,2	0	;Address of LTOS
7420	LDA 1,0,2	0	
7421	STA 0,0,2	LDA 0,.-3	
7422	SUB# 1,3,SZR	ADD 0,2	;Don't save block buffers
7423	JMP 7430	JMP 7433	
7571	200	1	

SYSTEM GENERATION

In order to use LTOS on your particular machine configuration, it is necessary to perform a system generation. This process is performed only once for a given core size and number of LINC Tape drives. A system generation is performed in order to create an LTOS that will execute in upper memory of your particular machine. After the sysgen is performed, the LTOS will automatically load into upper memory during the bootstrap process. Each and every tape that has been "sysgen'ed" for the particular hardware configuration will boostrap properly, loading LTOS into upper memory and then executing it. LTOS tapes containing useful files may be utilized with LTOS irrespective of the sysgen that was performed on them. The only restriction is that when bootstrapping, the tape that is on drive 0 must be one that has been properly sysgen'ed. The LTOS file tapes on the other drive(s) can be any LTOS tapes. Since the sysgen process alters only blocks below the file directory and the active file area of the tape, the sysgen process will not destroy an existing LTOS file tape. The sysgen process, as described below, involves using a special SYSGEN LINC Tape to create (assemble) a new LTOS for your hardware configuration. The SYSGEN tape can be used to perform a sysgen on any size Nova, 8K or greater, with at least one LINC Tape drive. If you have a 4K Nova, COI will supply LTOS already prepared. A user with an 8K or larger Nova can create an LTOS for any machine configuration, including 4K. Only one LINC Tape drive is required to perform the sysgen. After the first tape has been sysgen'ed, the user has the option to sysgen as many tapes at once as he desires by mounting them on drive 0 and pressing Continue. These and other directions are printed out during the sysgen process.

The SYSGEN LINC Tape is to be considered archival in nature. It contains files necessary to perform the sysgen and nothing else. It should never be used as a "scratch" tape.

The sysgen process takes about three minutes. Then as many tapes as are available may be sysgen'ed. The only requirement is that tapes to be sysgen'ed must be MARKed and certified Nova LINC Tapes, COI part number CO-635-256-N.

Sysgen procedure:

1. Load manual bootstrap in machine not equipped with auto program load.
2. Mount SYSGEN LINC Tape on drive 0 with reflective marker to right of head. Tape must be tensioned.
3. Start bootstrap or auto program load, if available, with device 74_8 on the switches.
4. -EXEC-? will print on TTY.
5. TYPE: ASSEM(CR)
6. TYPE: 1 when listing device is requested.
7. TYPE: 1 when mode is requested.
8. TYPE: $$\underline{N}$K(CR) when source file is requested
   where N = core size of target machine
   for example: $16K(CR) for a 16K Nova LTOS System.
9. TYPE: $$\underline{M}$T(CR) when next source file is requested
   where M = number of LINC Tape drives on target machine
   for example: $2T(CR) for dual drive LINC Tape System.
10. TYPE: LTOS$$(CR) when next source file is requested.
11. TYPE: 2 when mode is requested.

12. TYPE: SYSTEM(CR) when Binary File, Tape? is requested. This is name of LTOS assembler output file. If machine halts during the binary output phase, you probably have the Write Protect switch on for drive 0. Extinguish the red light by pressing this switch and then press Continue.

13. TYPE: 0 when mode is requested.

14. TYPE: SYSGEN(CR) following -EXEC-?.

15. TYPE: SYSTEM(CR) when LTOS Binary is requested.

16. Remove SYSGEN LINC Tape for future use. Mount tape to receive LTOS on drive 0. Press Continue. Repeat for as many tapes as desirable. When finished, start next process from bootstrap.

APPENDIX A

PROGRAM CALLS TO LTOS

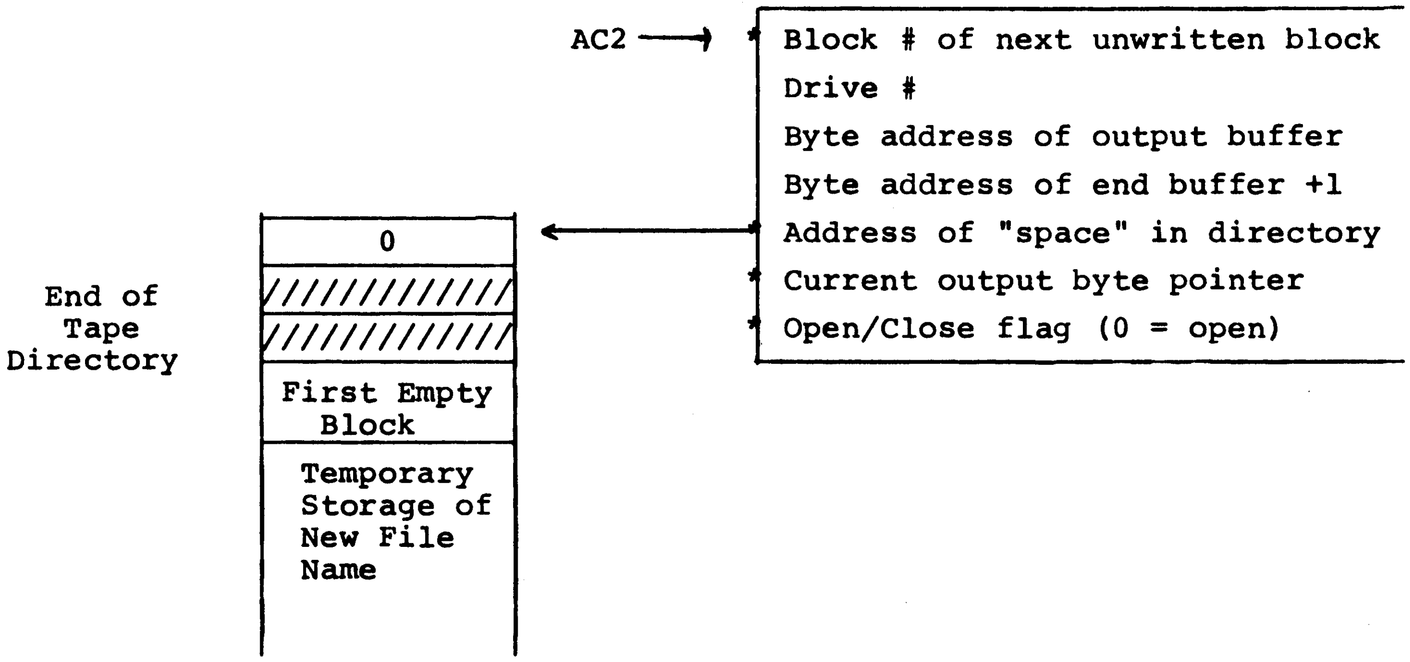
SYSTM (JSR@2)
(CODE)
(ARG1)
(ARG2)
ABN: - (error code returned in AC1)
NORM: - (arguments, if any, returned in AC's)
(Codes and the JSR@2 instruction are defined as assembler "user" symbols in the LTOS assembler supplied by COI)

COMMAND CODE (OCTAL)	SYMBOL	FUNCTION

0	.LOAD	Load a savefile. ARG1 is a byte pointer to the name. ARG2 is an execute/no execute indicator. 0 $\Rightarrow$ execute. If no start address in the file, or if no execute, do a normal return.
		AC0 = start address (-1 if none)
		AC1 = address of file entry in directory
		AC2 = AC2 before call
		AC3 = "ABN"

1	.SAVE	Write a logical record to an OPEN savefile. Do not close. ARG1 = WORD COUNT, ARG 2 = FIRST ADDRESS. If WORD COUNT = 0, then there will be no start address. If word COUNT = 1, then ARG2 = START ADDRESS. If WORD COUNT $>$ 1, then store words from memory, one less than word count.

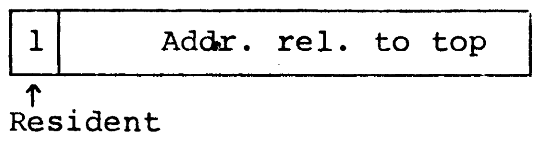
COMMAND CODE (OCTAL)	SYMBOL	FUNCTION
2	.WOPN	Create and open a new file. ARG1 = byte pointer to new name ARG2 = tape number Checks user's name for legal characters, makes sure there is no file of the same name already on the system, checks tape number, checks if room in directory, and opens write channel. Returns address of WRITE CONTROL Block in AC2.



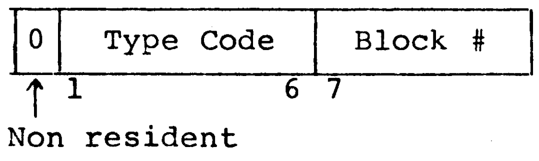
* Block # of next unwritten block
Drive #
Byte address of output buffer
Byte address of end buffer +1
* Address of "space" in directory
* Current output byte pointer
* Open/Close flag (0 = open)

COMMAND CODE (OCTAL)	SYMBOL	FUNCTION
3	.NEWF	No arguments. LTOS requests a new file name from TTI. If followed by comma instead of carriage return, request is made for a tape number. Proceed as for ".WOPN". (User should type message for operator before call.)
4	.PUT	ARG1 = byte in right half word left half doesn't matter. Put byte into write channel, writing tape block if necessary. Abnormal return if channel not open.
5	.CLOS	ARG1 = type code 0 $\Longrightarrow$ unformatted bit 1 (1) $\Longrightarrow$ savefile Close write channel, officially entering file into directory. Type code identifies the file for subsequent reading. Only bits 1-6 are used for type code.
6	.OLDF	User should type message to cue operator before call. No arguments. LTOS requests the name of an existing file from TTI, terminated by a carriage return. Proceed as for ".ROPN" below. (Returns same arguments.)

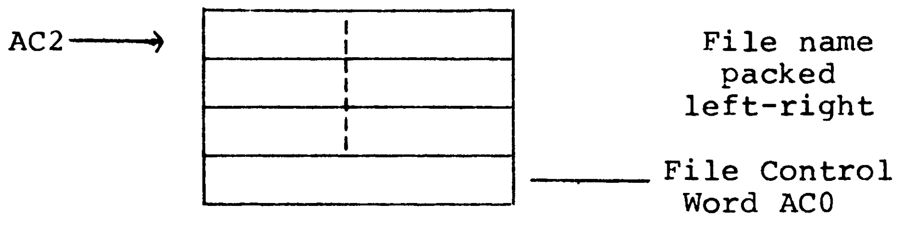
COMMAND CODE (OCTAL)	SYMBOL	FUNCTION
7	.ROPN	ARG1 = byte pointer to existing file name. Find file in directory and open read channel. Return file control word in AC0:



or



AC2 = address of directory entry



AC1 = address of READ CONTROL Block

AC1 →	Block #
+1	Drive #
+2	Byte addr. of input buffer
+3	Byte addr. of end of buffer +1
	____________
	____________
+6	Addr. of current input buffer ptr.

COMMAND CODE (OCTAL)	SYMBOL	FUNCTION
10	.GET	Return next byte from READ CHANNEL in AC0, right half. Left half = 0. Abnormal return if attempt to GET byte from resident file, or if uncorrectable LINC read error.
11	.ADDR	Return address of LINC UTILITIES AC0 , address of directory base AC1. First address of LTOS area ≡ address of WBUF in AC2.
12	.EXIT	Jump to LTOS command executive
13	.TYPE	ARG1 = byte pointer to packed message, term. by null. Type message. Left byte of each word first, up to but not including null.
14	.NUMB	Prompt with space and input octal number, returning it in AC1. Octal string must end in comma, AC0 ≠ 0 or carriage return AC0 = 0. If not, abnormal return with AC0 = character. All input is echoed.
15	.NAME	No arguments. Type ? and input and echo a name from the Teletype. The name is packed into a buffer in LTOS. A byte pointer to it is returned in AC0. Name may be printable characters except comma (41⟶137), terminated by return AC1 = 0 or comma AC1 ≠ 0. (ESC) character causes abnormal return. Character codes 0⟶ 40 ignored; codes 140⟶177 cause repetition of the ? prompt.

SUMMARY OF EDITOR KEYBOARD COMMANDS

A	append text to lines in core, from keyboard.
C m(,n)	change; delete line(s), then insert text from keyboard.
D m(,n)	delete line(s) from core.
E	close LINC read file, simulate CTRL-D.
G $string$ G m,$string$	get, list on TTY, all lines containing string. as above, but begin search at line *m*.
H H m(,n)	list all lines on printer. list line(s) on printer.
I m	insert text *before* line *m*, from keyboard.
K	kill all lines in core, close LINC read file, destroy unclosed LINC write file.
L	list; same as H above, but for TTY printer.
M m(,n) oldstring newstring	modify line(s), substituting new string for *first* occurrence of old string in each subject line. List modified lines on TTY. Either string may be empty, i.e. just carriage return.
P m(,n)	punch line(s), waiting for keystroke before and after punching.
R (x)	read (x) lines from LINC read file, or until END OF FILE character, CTRL-D.
T (x)	punch either 10 or (x) inches of leader/trailer; wait for keystroke before and after punching.
W m(,n)	write line(s) to LINC write file, then *delete* lines written.
X	exit to LTOS executive, unless lines remain in core, or LINC files open.
Y (x)	yank (x) lines from paper tape device, or until timeout.
Z	write END OF FILE character to LINC write file, close file, enter name in LTOS file directory.

EDITOR USAGE NOTES

1. EDITOR types * as command prompt. Errors, aborted commands cause ?? message, followed by prompt.

2. Any command will be aborted by typing ESC (CTRL-SHIFT K).

3. All command lines must be followed by carriage return.

4. Command parameters in parentheses are optional. Multiple parameters must be separated by comma.

5. Parameters _m_ and _n_ represent line numbers. Line numbers are sequentially assigned to the text in core, and are adjusted after each command.

   _m_ and _n_ may be specified by any of the following:

   a) decimal literal
   b) $string$, evaluated to line in which string first occurs. Search begins at line 1 for _m_, at line _m_ for _n_.
   c) / meaning the last line
   d) . the current line. For parameter _m_, the last line processed, or last+1 in case of previous delete (D,W). For parameter _n_, the value of _m_.
   e) arithmetic combination of any of above (+,-).

   ```
   Examples:  L56,59    list lines 56 through 59 inclusive
              L/-4,/    list last 5 lines
              L$LABEL:$,.+2    list 3 lines, starting at the
                               first line containing LABEL:
   ```

6. Parameter _x_ should be a literal decimal count.

7. Commands G H L M P W (output) may be "soft" halted by striking any key. The halt always occurs at the end of a line. The current line pointer (.) will locate the last line processed, last+1 for W.

8. EDITOR may be restarted at any time by NOVA console "RESET" and "START" at location 3. Text and program status will be unchanged, but you do this at the risk of incomplete I/O.

9. Any text command which might increase core space beyond the high limit (bottom of LTOS) will be stopped, and the message FULL will be typed.
   -N.B.- a CHANGE resulting in FULL means _all_ the specified lines were deleted, but only those typed were stored.

10. TEXT: any kind of text input is subject to the same set of rules, whether from keyboard, paper tape, or LINC file. These rules also apply to the strings in command specifications.

Permissible graphics - upper case letters, numerics, and punctuation, plus space. ASCII codes 40-137 octal.

Horizontal TAB (11) and FORM (14). FORM is not simulated on input, just on output.

Truncation of a line occurs after 72 characters; the message !!TR!! will be printed.

CTRL-A echoes ←, deletes the previous character in the line, unless empty or truncated, and can be repeated.

CTRL-X cancels and restarts the whole line.

ESC (CTRL-SHIFT K) cancels the current line, and stops the command in process.

CARRIAGE RETURN completes the current line.

END OF FILE (CTRL-D) on LINC read only, closes file, acts as ESC above.

ALL OTHER CHARACTERS ARE IGNORED.

PARITY IS NOT CHECKED ON INPUT. We rely on LINC Tape error detection for LINC files.

11. OUTPUT: text output from core is transmitted character-for-character, except for 3 cases. First, a carriage return is always followed by a line feed. Second, tabs are either followed by a null or simulated by substitution of spaces, modulo 8. Third, form feeds are either followed by 12 nulls or simulated by carriage return/line feed pairs, modulo 66 (the number of lines on 11 inch paper). The conditions of simulation are as follows:

TTY print - simulate tabs and forms if ASR33, neither if ASR/KSR 35, as defined in initialization.

Line printer - simulate tabs only.

Punch and LINC write file - as defined in the initialization phase. Normally, no simulation is desirable, in order to minimize storage space. However, if you wished to list a paper tape off-line on a machine without tab or forms hardware, then you might specify simulation for punch and LINC output. Another example is the listing of a file on the line printer without line numbers. In this case, you would simulate tabs, and use the LTOS program DUMP to list the EDITOR output.

12. Output parity. The state of bit 8 on all characters output by the EDITOR is defined in the initialization phase. It may be even parity, always 1, or always 0 (7-level). The line printer and TTY printer ignore bit 8. As previously stated, the EDITOR does not check input parity, so this convention is for the convenience of other programs or equipment.