COMPUTATION CENTER
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS 02139

TO: ALL CTSS USERS

FROM: COMPUTATION CENTER PROGRAMMING STAFF

SUBJECT: INTRODUCTION TO CTSS USAGE

DATE: JULY, 1967

The attached is minimal information to enable the new CTSS user to compile, load, and execute his FORTRAN and MAD programs.

The CTSS Programmer's Guide is the basic source for CTSS information; section references in the text are to this Guide. The "Primer on Use of Time-Sharing," a learner's guide, from the M.I.T. Department of Aeronautics and Astronautics and "Using FORTRAN on Time-Sharing" by Donel McKenzie from Northeastern University have been quoted liberally with permission of the authors.

PRELIMINARY REMARKS

Introduction

The time-sharing system, CTSS, gives the user potential access to the computer in a sequence of short bursts of time, with other users employing the intervals in between or his time if it is not required by him during that burst. From the user's point of view it is as if he has sole use of the machine with all its capabilities, except that the real time (the clock-on-the-wall time) taken to execute a program may seem lengthened.

Rapid response to the user, which this system makes possible, is of great use in modifying or correcting programs, exploratory calculations, and many other applications in which repeated back and forth interaction between the user and computer is desirable. If a program requires a large amount of computer time, or if it produces a large amount of output, the system is not usually used.

At the present time, CTSS at the Computation Center is in operation from 10 a.m. to midnight on weekdays; and from 9 a.m. to midnight on weekends. CTSS users can be locked out during these periods for forced background operations (batch processing). Currently scheduled lock-out periods are daily from 9 a.m. to 10 a.m., noon to 1 p.m., 5:30 p.m. to 6:30 p.m., and midnight to 1 a.m. 24-hour operation will begin as soon as the 7094 emulator is in operation on the 360 - 65/40 A.S.P. system.

Some familiarity with a programming language (FORTRAN or MAD) is required to use the computer, whether by card input (batch processing) or by console (time-sharing).

Use of this Primer

The comments which follow are intended to be a brief explanation of the use of the time-sharing system. By no means are all the available commands discussed, nor are those mentioned here necessarily completely discussed. The information given here is minimal but should be enough for one to begin using the time-sharing system and to reference the CTSS Programmer's Guide.

Using CTSS

Commands

Commands are messages sent from the user's console to the computer, specifying the next action it is to take; they are really inputs to a program called the "supervisor" which controls and implements the CTSS system. Issuing a command involves giving its name, followed perhaps by one or more other names or numbers which serve as parameters required to completely specify the action to be taken. The command name and the various parameters are separated by blanks. Every individual CTSS command and editing request must be terminated with a carriage return.

Typing from the Console

When using an IBM 1050 or 2741 console to communicate with the computer, instructions and data should be typed in one or more lines on the typewriter; the machine takes cognizance of and, if appropriate, action on each line after and only after each carriage return.

Alphabetic characters should be typed only in lower case. The shift should be used only as required for certain symbols. The computer responses always will be in upper case when printing alphabetic characters.

If an error is made in typing, use either of the following:

a. Type either ", the quotation mark, or # , the number sign, to delete the preceding character. To delete the preceding two characters, type " or # twice, and so on.

b. Type either ?, the question mark, or @ , the at sign, to delete everything typed since the previous carriage return. One continues as if at the beginning of the line.

When a command is sent to the computer and action is initiated by the carriage return, the console types back W, followed by a number. W means wait; the number is the current time of day. After the command has been processed, the console types R, followed by two numbers which are separated by a + sign. This means ready for the next command. The sum of these two numbers is the time used in seconds processing this command. At any point between a W and an R the quit signal, two pushes on the reset line key, may be given. It instructs the computer to cease action on the last command and return to the ready status. The computer immediately should type QUIT on the console. If it doesn't, one should press the button more slowly. On the teletype console, the quit signal is generated by depressing and releasing the break key, the left-most button below the keyboard. On the 2741 console the quit signal is given by depressing the

attention (attn.) key. The small lever on the left side of the console must be in the communication (com.) position.

Turning On and Dialing In the Console

To turn on the IBM 1050 console, a user sets the power-on switch at the right of the console stand and picks up the Data-phone receiver. Pushing the second button from the left on the Data-phone gets the dial tone; however, some Data-phones use the second button from the right. Dialing 0 (zero) on a console reaches the Computation Center. However, if the line is busy, the user should hang up and try again. When the computer answers, there is a high-pitched tone. Depressing the Data-phone hold button (the first button on the left) completes the connection to the computer. The 2741 console works the same as the 1050.

On the teletype, pressing the ORIG button turns on the machine and the telephone dial tone. A user must dial 9 for the Computation Center. If the line is busy, he can hang up by pressing the CLR button. If the computer answers, he hears a high-pitched tone.

If a yellow light with the words RCV ALARM lights up instead, the paper is not feeding properly. There is a switch behind the rubber cylinder that should be pushed in by the paper. If everything is in order, then the computer will type a message on the console stating the current number of users on the system and the maximum number of users allowed. If there is an empty space and the proceed light is on, one can start typing the command LOGIN. Note that only the 1050 has a proceed light.

LOGGING COMMANDS

Commands

LOGIN probno name
LOGOUT

Usage

After the console has been turned on and dialed in, the user should type a line of the following form:

LOGIN M3801 WIDNALL

followed by a carriage return. That is, probno is the user's problem number; name is the user's last name of which only the last six characters are used.

If this line is correctly written, the console types PASSWORD. When the secret password is typed, it will not print on the console. At this point, if traffic is too heavy, a minus sign will print on the console and indicate to try again later; otherwise, it will indicate that one is logged in and type out some other information, mainly of a bookkeeping nature (for details see the CTSS Manual section AH.1.01). The console types R and the time used before a user proceeds.

After a console session is finished, the logout command is given by typing LOGOUT, followed by a carriage return. The console will then print out a little bookkeeping information and turn itself off. On certain occasions the computer may automatically log one out. In such a case, a message to this effect is typed out and the console is turned off. If this happens while a user is running a program, the program is stopped and its entire contents are stored in a file automatically created. At a later time one can, with the CONTIN command (not discussed here), restore this program and resume executing from where it left off.

FILE MANIPULATING COMMANDS

Commands

CHMODE	name1A	name2A	mode		
CONTIN	name1A				
DELETE	name1A	name2A	name1B	name2B	etc.
ED	name1A	name2A	name1B		
EDL	name1A	name2A			
LINK	name1A	name2A	probno	progno	
MYSAVE	name1A				
PERMIT	name1A	name2A	mode	probno	progno
PRINT	name1A	name2A			
RENAME	name1A	name2A	name1B	name2B	
RESUME	name1A				
REVOKE	name1A	name2A	probno	progno	
RQUEST	output-type	name1A	name2A		
UNLINK	name1A	name2A			

Remarks

Each user is assigned a certain number of records in magnetic disk memory to be used for his private files. A user, however, may permit files in his directory to be accessed by other users and, in turn, may link to files in other users' directories. This is done with the commands, PERMIT and LINK.

These files are analogous to decks of cards. Most often, the arguments of commands are file names. These files may be input from CTSS consoles or from punched cards. Their actual location is of no importance to the programmer because he always refers to files by name.

Each file is required to have two names, a primary name and a secondary name, each of which consists of six or fewer characters. A user almost always can assign any file name he wishes, but he should use names that are easy to remember and are reflective of the contents of the file. The secondary name may or may not be arbitrary depending on the contents of the file and the way in which they are to be used.

type of deck	name1	name2
FORTRAN source deck	optional	FADTRN
MAD source deck	optional	MAD
a binary deck for executing	optional	BSS

Each FORTRAN and MAD source program or subprogram should be a separate disk file with its own unique file name.

Several binary files can be COMBINed into a new file of the same secondary name; thus the binary files of the main program and its subroutines may be COMBINed and LOADed under a single BSS name.

Source files also may be ARCHIVed into a single file of the same secondary name. This saves entries in the file directory and disk space.

A source file may be one of two types: card-image or line-marked. A card image is a string of 84 characters. Each line stored in a file is always 84 characters long, regardless of the actual number of characters typed by the user. In a line-marked file the lines vary in length depending on the amount of non-blank information they contain. Consequently, line-marked files take less disk space and are usually preferable; however, they can not be used in all cases.

Usage

The basic commands described here are for printing, storing, rereading, and altering or correcting files. The reader is referred to the CTSS Guide for more specialized commands.

Listing Disk Files

This command is given by typing LISTF followed by a carriage return. The console responds by typing a list of the names of the user's files, starting with the most recent and working backward. A little information about each, such as number of records used, is also given.

LISTF also enables the user to list selectively the contents of a file directory by permitting him to specify the following: file directory, file names, authors, modes, range of dates last used, range of dates last modified, sorting process, and output forms.

The user has the option to suppress the search for linked files or to search only for linked files.

See section AH.5.01 for the description of these numerous selectivity features.

Printing a Disk File

To print a disk file, one types:

PRINT name1A name2A

followed by a carriage return. The computer then will type a printed copy of the set of cards in this file. If only a portion of the file is wanted, it is usually obtained by using the ED command, discussed on the next page.

Deleting a Disk File

One removes files completely by typing:

DELETE name1A name2A name1B name2B etc.

where as many as nine primary-secondary name pairs can be included in the itemization.

Printing or Punching a Disk File

It is often convenient to have the contents of files printed or punched off-line for pickup later at the Computation Center. In order to print out a file, the user types:

RQUEST PRINT name1A name2A

To punch out a MAD or MADTRN file:

RQUEST DPUNCH name1A name2A

To punch out a DATA file for later reloading on CTSS:

RQUEST 7PUNCH name1A name2A

To punch out a BSS file for later running under batch processing:

RQUEST BPUNCH name1A BSS

These commands do not actually cause printing or punching at the time they are given on the console but set up a file named OUTPUT RQUEST in the user's file directory for later action by the Disk Editor, a utility program. Therefore, the files named must be left intact and not deleted until the printing or punching has been done (normally 24 hours).

Creating and/or Editing a Disk File from the Console

To create a new card-image file, or to make changes in one previously created, the ED command may be used. With this command the user can type in his program or change individual lines or characters within a line with a minimum of effort and delay. (1)

The ED command is initiated by typing:

ED name1A name2A name1B

where name1A name2A is the name of the file to be edited or created. If the file name1A name2A does not exist already, ED will assume that a new file is to be created and will start in the INPUT mode. If the file already exists, the EDIT mode will be entered. If the optional argument name1B is specified, a file name1B name2A will be created and name1A name2A will remain unaltered.

INPUT Mode

When this mode is entered, the ED command will type INPUT: on the user's console. While operating in this mode, the ED command will accept input lines from the user's console. Each tab will be interpreted automatically for each input line and will cause a skip to the next logical field (ex.: A tab given after inputting a statement label on a MAD line will cause the next input character to be set in column 12.). The tab settings on the console are ignored by the computer. The user types in his program, each card on a separate line. The program creates a line number in columns 75 to 80 for each line. When he is finished, the return is struck twice to resume the EDIT mode. Then the command, FILE, is typed to create the new file.

EDL may be used to create a line-marked file. It is similar to ED except tab is left in the text instead of being interpreted as a skip to the next logical field.

EDIT Mode

When the user enters this mode, the response, EDIT:, will be typed on the console. At this time the user may type requests of the ED/EDL command. All changes made to a file immediately become effective to a temporary file and, as a result, the user is able to make recursive modifications to his file.

One may think of a pointer which is positioned at a line in the edited file. When the user enters the EDIT mode from the INPUT mode, this pointer will be positioned at the last input line typed by the user. When the user first starts the ED command to edit an existing file, the pointer is positioned before the first line in the old file. If the end of file is reached by the pointer, the comment END OF FILE REACHED BY: is typed on the user's console followed by the request which caused the end of file to be reached. At this time the pointer will be positioned after the last line in the file. The user must reposition the pointer to any line he wishes to edit. When in the EDIT mode, any illegitimate EDIT request will cause the comment NOT A REQUEST: to be typed on the user's console followed by the request which caused the error. In many cases it is possible for the user to stack EDIT requests (enter several requests at once). If one of the requests causes an error message to be typed, any stacked requests will be ignored. This is done in case one of the stacked requests depends on the successful completion of the request in error.

Any number of initial tabs or spaces, including 0, may occur in a request line. Arguments and the request must be separated by at least one space or any number of tabs or spaces. Normal mode for EDL is VERIFY ON, for ED, VERIFY OFF.

Requests of the ED/EDL Command

	ED	EDL
REQUEST:	LOCATE string	
ABBREVIATION:	L string	
RESPONSE:	line found*	line found*
ERRORS:	END OF FILE	

The LOCATE request moves the pointer forward from its present position to the first line which contains the entire character string specified by string. The full line of 84 characters is scanned, so that string may specify line numbers. String should include the leading zeros of the line numbers to avoid any undesired match with numerical program constants.

* In verify mode, otherwise no response.

	ED	EDL
REQUEST:	NEXT n	
ABBREVIATION:	N n	
RESPONSE:	next line*	next line*
ERRORS:	END OF FILE	

This request moves the pointer forward from its present position in the file. The number of lines to be skipped over is n. If n is zero or not specified, it is assumed to be one and the pointer will be moved to the next line in the file. If the NEXT request is given after the end of file has been reached, the pointer will be reset to the beginning of the file and moved n lines from there.

REQUEST:	DELETE n	
ABBREVIATION:	D n	
RESPONSE:	none	none
ERRORS:	END OF FILE	

The DELETE request deletes n lines from the file starting with the line at which the pointer is positioned. The pointer is left at the position vacated by the last line deleted by this request. If n is zero or left unspecified, only the current line will be deleted.

REQUEST:	PRINT n -L-	
ABBREVIATION:	P n -L-	
RESPONSE:	printed lines	printed lines
ERRORS:	END OF FILE	

The PRINT request prints n lines from the file starting with the line at which the pointer is positioned. Upon completion of this request, the pointer will be left pointing to the last line printed. If n is zero or left unspecified, one line will be printed. Normally lines are printed without line numbers. If the optional character L is present in the PRINT request, line numbers will be printed to the right of the printed lines.

---

* In verify mode, otherwise no response.

	ED	EDL
REQUEST:	RETYPE line	
ABBREVIATION:	R line	
RESPONSE:	none	none
ERRORS:	none	

This request causes the line at which the pointer is positioned to be replaced by line, which is a normal input line and may contain tabs and backspaces. The pointer is not moved by this request.

REQUEST:	TOP	
ABBREVIATION:	T	
RESPONSE:	none	none
ERRORS:	FILE WORD COUNT ZERO NOTHING IN FILE INPUT:	

This request resets the pointer and positions it before the first line in the file. If the file is empty, the INPUT mode is entered.

REQUEST:	BOTTOM	
ABBREVIATION:	B	
RESPONSE:	INPUT:	INPUT:*
ERRORS:	none	

This request positions the pointer after the last line in the file. Upon completion of this request, the INPUT mode will be resumed. All subsequent lines will be treated as input and added to the bottom of the file.

REQUEST:	(Carriage return)	
ABBREVIATION:	none	
RESPONSE:	INPUT:	INPUT:*
ERRORS:	none	

This request resumes the INPUT mode. All subsequent lines will be treated as input and inserted <u>after</u> the line at which the pointer is positioned.

---

* In verify mode, otherwise no response.

	ED	EDL
REQUEST:	INSERT line	
ABBREVIATION:	I line	
RESPONSE:	none	none
ERRORS:	none	

The INSERT request may be used to insert a single line without changing to the INPUT mode. It is inserted following the line at the present pointer position and is a normal input line.

REQUEST:	CHANGE *string1*string2* n -G-	
ABBREVIATION:	C *string1*string2* n -G-	
RESPONSE:	changed line*	changed line*
ERRORS:	END OF FILE	

This request examines n lines starting at the line at which the pointer is positioned. Upon completion, the pointer will be left positioned at the last line examined by this request. If n is zero or left unspecified, it is assumed to be one and only the current line will be examined. The character * is the delineator separating string1 and string2 and may be any character not appearing in the strings. Because string1 and string2 are arbitrary character strings, they may be different lengths. If the optional character G (GLOBAL) is present, every occurrence of string1 in the entire line will be replaced by string2. If G is not present, only the first occurrence of string1 will be replaced by string2 in each examined line. Example:

LINE:	ALPHA= ALPHA+ALPHA
REQUEST:	C *ALPHA*BETA*
NEW LINE:	BETA= ALPHA+ALPHA
REQUEST:	C *ALPHA*DELTA* 1 G
NEW LINE:	BETA= DELTA+DELTA

REQUEST:	VERIFY	VERIFY ON
ABBREVIATION:	VE	V ON
RESPONSE:	none	none
ERRORS:	none	

The VERIFY request turns on the verify mode. In the verify mode, completion of any of the requests -- FIND, LOCATE, and CHANGE -- will cause the printing of the line or lines being edited. Information about the FIND request can be found in section AH.3.02. Requests can not be stacked while in the verify mode. This is the normal mode for EDL.

	ED	EDL
REQUEST:	BRIEF	VERIFY OFF
ABBREVIATION:	BR	V OFF
RESPONSE:	none	none
ERRORS:	none	none

The BRIEF request turns on the brief or normal mode. Within the brief mode, the FIND, LOCATE, and CHANGE requests will not give the responses expected in the verify mode. This is the normal mode for ED.

REQUEST:	FILE -name1B-	
ABBREVIATION:	FL	none
RESPONSE:	ready message from CTSS	same
ERRORS:	NO FILE NAME GIVEN	
or	FILE WORD COUNT ZERO NOTHING IN FILE INPUT:	

This request terminates the editing process and writes the new edited file on the disk. The optional name1B specifies that the new file is created as name1B name2A. If name1B is not specified, the old file will be replaced by the edited file or a new file name1A name2A will be created. As an example, FILE SUB2 will create a new file with a primary name SUB2 from the file which is being edited. If the original file were called SUB1 MAD, one would have both SUB1 MAD and SUB2 MAD.

Additional requests which may be used with the ED command will be found in section AII.3.02. A practice console presentation is in section AA.2.

Creating a Disk File Off-line

Because the console is a relatively slow input device, it is desirable to be able to enter programs and data into disk files from card decks. This function is performed by a program known as the Disk Editor. Decks inputted this way become card-image files.

The deck order of a bulk input job is:

1. Initial control card for the Disk Editor program, beginning in column 1:

   INPUT problem no. programmer no. name1 name2

2. Input deck

3. Terminal Control Card for the Disk Editor program, beginning in column 8:

   EOF

Bulk input to CTSS is submitted in the Computer Room, 26-150, and is returned in the metal card files provided there.

Resuming the Status of a Program Previously Saved by an Automatic Logout or by the User

The user may preserve the machine conditions and the associated program either just loaded or just interrupted by the quit button via the MYSAVE command. Resumption of the SAVED file is by a CONTIN command. Formats are:

```
MYSAVE  name1A

CONTIN  name1A
```

At any time an automatic LOGOUT may be initiated by the system. If the user is executing a command or running a program, a SAVED file called prog'L' SAVED (prog is the user's programmer number) will be created in a temporary mode. A temporary file will be deleted as soon as it is used. If a user intends to resume a prog'L' SAVED that was created during a previous console session, he first should change the mode to be permanent, in case he needs to RESUME it more than once. Temporary mode files are deleted by normal LOGOUT. If a user intends to RESUME a prog'L' SAVED at a future time, he should change it to permanent mode immediately.

If a user was in the process of editing a source file when he received an automatic logout, editing may be continued by resuming the prog'L' SAVED file. When resumed, the editing commands will announce the presence of one of the intermediate files created by the ED program, if one is present; the user must type either yes to the question about deleting it or type no and then RENAME it. The commands will not proceed unless the intermediate file is disposed of in some way. Formats are:

```
CHMODE  name1A  name2A  mode

RENAME  name1A  name2A  name1B  name2B
```

RESUME name1A

where

i. name1b name2b is the new file name.

ii. mode argument is:

0 - Permanent (octal 000)

T - Temporary (octal 001)

R - Read-only (octal 004)

W - Write-only (octal 010)

V - Private (octal 020)

P - Protected (octal 100)

Linking to Files in Other Directories

When the owner of a file grants someone permission to link, he specifies who will be permitted to access and in what apparent mode he can locate the file. Usage is:

i. to grant permission - done by owner

PERMIT name1A name2B mode prob prog

ii. to withdraw permission - done by owner

REVOKE name1A name2A prob prog

iii. to form a link - done by borrower

LINK name1A name2A prob prog

iv. to remove a link - done by borrower

UNLINK name1A name2A ... name1n name2n

(See section AH.3.05 for more details.)

Links allow several users to access a single file, saving file space.

COMPILING COMMANDS

Commands

FOR4	name1A of file in FORTRAN IV, Version 13
MAD	name1A of file in MAD
MADTRN	name1A of file in FORTRAN II

Remarks

To compile a FORTRAN program on CTSS, both a MADTRN and a FOR4 compiler are available. Both compilers convert FORTRAN programs into the MAD language and then use the MAD compiler to convert to machine language. The MADTRN compiler compiles FORTRAN II; FOR4 compiles a subset of FORTRAN IV, Version 13, which is used under IBSYS.

The MADTRN compiler assumes a working FORTRAN II deck and, therefore, MADTRN diagnostics are minimal and pertain to the intermediate MAD file, not the FORTRAN file. The FOR4 compiler has more error diagnostics, most of which pertain to the FORTRAN file and some of which are unreliable. For instance, NO END CARD is a common comment when the user accidentally used a line-marked input file.

The FOR4 program is frozen code; sensitive areas and bugs are known to exist. If a user has trouble in the following suspect areas, he should recode around it. These include:

1. COMMON and continuation card combination
2. LOGICAL IF (2)
3. DATA statement
4. Mixed expressions (MADTRN does not give error messages on these; FOR4 does.)
5. Continuation cards

Preparations for Compiling FORTRAN II Programs by MADTRN

1. The FORTRAN II source file must have the secondary name MADTRN.

2. The FORTRAN II source file must be a card-image file created either by the ED command or through the bulk input facility. One can not use the commands, EDL and XPAND, in sequence to produce this card-image file.

Preparations for Compiling FORTRAN IV Programs by FOR4

1. and 2. as above

3. A link must be set up to the file, F4LIBE BSS, which is a program that contains the entries for all the built-in functions which do not process complex or double-precision data exclusively. (3)

LINK F4LIBE BSS M1416 CHFL04

Note: This is read CHFL zero 4.

4. In subsequent loading for execution, the command ii. below, is faster because it eliminates a search in F4LIBE BSS for the entry names.

i. LOAD name1A of (LIBE) F4LIBE program

ii. LOAD name1A of F4LIBE program

5. The FORTRAN IV, Version 13 source language accepted by FOR4 is a subset of that implemented by IBSYS; any restrictions are stated in the CTSS Programmer's Guide, section AH.2.17.

Changes Needed to Convert a FORTRAN II Program to FORTRAN IV So That It Can Be Compiled with FOR4

1. through 4. as above

5. The off-line input/output statements must be modified. (See the section on off-line input/output.)

6. Built-in function names must be changed to conform to their counterparts in FORTRAN IV, Version 13. (See list below.)

7. Library function names must be changed by deleting the terminal F and in some cases by changing the name, e.g., LOGF becomes ALOG.

Built-in Function Specification

Names of FORTRAN II's Built-in Functions	Corresponding Names of FORTRAN IV, Version 13's Built-in Functions
ABSF XABSF	ABS IABS
DIMF XDIMF	DIM IDIM
-----	------- IFIX
FLØATF -----	FLØAT -------
INTF XINTF	AINT INT
MAXØF XMAXØF	AMAXØ MAXØ
MAX1F XMAX1F	AMAX1 MAX1
MINØF XMINØF	AMINØ MINØ
MIN1F XMIN1F	AMIN1 MIN1
MØDF XMØDF	AMØD MØD
SIGNF XSIGNF	SIGN ISIGN

Preparations for Compiling MAD Programs

1. The MAD source file must have a secondary name MAD.

2. The MAD source file may be a card-image file (created by ED or by the bulk input facility) or be a line-marked file (created by the EDL command).

Usage

After the appropriate MADTRN or MAD file has been created, one of the following commands is used to compile the program or to translate it into machine language:

```
FOR4      name1A
MAD       name1A     or   MAD      name1A  (LIST)
MADTRN    name1A     or   MADTRN   name1A  (LIST)
```

If errors exist in the source program, diagnostic messages are printed on the console. Otherwise, each compiler produces a file named name1A BSS, the machine language version of the program, which subsequently can be used for execution. Because MADTRN and FOR4 convert to MAD and then to machine language, a name1A MAD file is created by these two compilers.

The optional parameter (LIST) produces a file name1A BCD, a listing file giving the machine instructions produced. This listing is sometimes helpful in debugging. Because of its size, it is usually printed off-line using the RQUEST command. Other parameters for the three commands may be found in sections AH.2.10 and AH.2.17.

COMMANDS FOR EXECUTING PROGRAMS

Commands

both: LOAD name1A of a BSS file name1B of a BSS file etc.
START (after the LOAD command has finished)

or: LOADGO name1A name1B etc.
(a combination of LOAD and START)

Usage

To run a program, one gives the command:

LOAD name1A name1B etc.

where name1A, name1B, etc. are the first names of the main program and any subprograms that may be required. This is analogous to loading the binary decks respectively stored in files name1A BSS, name1B BSS, ... into the computer. After the program and subprograms have been loaded, the CTSS library is searched for any standard subprograms which may be required, such as exponential function or square root. When this is completed, the machine types R for ready, and the user may type START to begin executing his program.

Alternatively one may issue the command:

LOADGO name1A name1B etc.

which starts the execution automatically after loading is completed.

When execution begins, the computer will type EXECUTION. To terminate execution anytime before the program is finished, one can quit by pushing the key marked RESET LINE twice.

INPUT/OUTPUT INSTRUCTIONS USED IN MAD AND FORTRAN PROGRAMS

On-Line (4)

Reading Data from the Console

If a program requires data input which is to be provided from the console, the appropriate READ-type statement should be used. (See the list below.) When control comes to such a statement, it will wait for the data to be typed on the typewriter, accepting it after each carriage return. It is desirable to precede such statements by a statement which will print a comment on the console asking for data.

Writing Data on the Console

When control transfers to a PRINT-type instruction during program execution, the current values of the data indicated will be typed on the console.

Off-Line (5)

Pseudo-Tapes on CTSS

It is often convenient to read and write data off-line so that during input, large amounts of data need not be typed in each time the program is executed and on output the console is not tied-up with long printouts. This is done by CTSS with pseudo-tapes. A pseudo-tape is really a disk file, but it is treated by a FORTRAN or MAD program as if it were a tape. A pseudo-tape's primary name must be .TAPE. and its secondary name must be an unsigned integer constant, for example .TAPE. 19. It may be in BCD or binary mode.

Reading a Pseudo-Tape File

Pseudo-tape files for input may be created the same way as other files which are created from the console by using the ED or EDL commands, or submitted as a card deck for bulk input. They also may be pseudo-tape output from a previous execution. They may be either card-image or line-marked files when used for program input.

To read a BCD pseudo-tape file, one uses the FORTRAN II statement:

```
READ INPUT TAPE n,f,list
```

(or the FORTRAN IV or MAD equivalent; see the table below). Reading starts at the beginning of the file when it is initially referenced during an execution and proceeds sequentially through the file. The FORTRAN II and FORTRAN IV statements, END FILE n and REWIND n (or the MAD equivalents), merely cause the file to

be closed (that is, rewind the pseudo-tape). Any subsequent reading would then start at the beginning of the file.

Writing a Pseudo-Tape File

To write a BCD pseudo-tape file, one uses the FORTRAN II statement:

```
WRITE OUTPUT TAPE n,f,list
```

(or the FORTRAN IV or MAD equivalent). If the corresponding pseudo-tape file does not already exist, one is automatically created. If one does exist, the new information will be appended to the end of the file. Thus, writes may be staggered over several executions of a program using the pseudo-tape file. But, if the user wants a clean tape, he should delete any old pseudo-tapes of the same name prior to execution. The user also should REWIND the tape when he is through to make sure that the file is closed.

Pseudo-tape files are always written as line-marked files, not card-image ones. Therefore, any modifications to these files must be made using the EDL command, not ED.

Partial List of Input/Output Instructions (5)

Type of Input	MAD	FORTRAN II	FORTRAN IV, VERSION 13
from console	READ FORMAT f,1 READ DATA READ AND PRINT DATA	READ f,1	READ f,1
BCD Pseudo-tape	READ BCD TAPE n,f,1	READ INPUT TAPE n,f,1	READ (n,f)1
Binary Pseudo-tape	READ BINARY TAPE n,1	READ TAPE n,1	READ (n)1

Type of Output	MAD	FORTRAN II	FORTRAN IV, VERSION 13
to the console	PRINT FORMAT f,1 PRINT COMMENT $message$ PRINT RESULTS PRINT BCD RESULTS PRINT OCTAL RESULTS READ AND PRINT DATA	PRINT f,1	PRINT f,1
BCD Pseudo-tape	WRITE BCD TAPE n,f,1	WRITE OUTPUT TAPE n,f,1	WRITE (n,f)1
Binary Pseudo-tape	WRITE BINARY TAPE n,1	WRITE TAPE n,1	WRITE (n)1

OTHER COMMANDS OF INTEREST

Other commands, listed with the appropriate section in the CTSS Guide, may be of interest to beginning users. The reference material for commands mentioned above also is listed.

Command	Section	Command	Section
ARCHIV	AH.4.01	MADTRN	AH.2.11
COMBIN	AH.6.01	MYSAVE	AH.3.03
DELETE	AH.6.03	PERMIT	AH.3.05
ED	AH.3.02	PM	AH.8.03
EDL	AH.3.07	PRBSS	AH.5.05
FAP	AH.2.07	PRINT	AH.5.03
FIB (7)	AH.1.03	RENAME	AH.6.03
FOR4	AH.2.17	RESUME	AH.6.06
LINK	AH.3.05	RQUEST	AH.6.06
LISTF	AH.5.01	SAVE	AH.3.03
LOAD	AH.7.01	SQUASH (6)	AJ.4.02
LOADGO	AH.7.01	START	AH.7.03
LOGIN	AH.1.01	STORAP (6)	AJ.8.01
LOGOUT	AH.1.02	TTPEEK	AH.1.04
MAD	AH.2.10	XPAND (6)	AJ.4.03

REFERENCES

The basic reference for CTSS is the "Compatible Time-Sharing System Programmer's Guide," published by the M.I.T. Press and available at the TECH COOP, 84 Massachusetts Avenue, Cambridge, Massachusetts 02139. One should be sure to purchase the unbound version of the second edition. Updates may be picked up at the Publications Office of the Computation Center, Room 32-1006, where the new user can place his name on the mailing list to receive further updates.

The Publications Office issues other memos which are available to all users. Its memo series includes CC-251, an index of all CC memos, and CC-255, an index of FORTRAN and MAD references, both of which are a must for the new user. M.I.T. staff members may obtain IBM manuals from the Publications Office. The most pertinent manuals are:

FORTRAN II Programming, C28-6054
FORTRAN IV Language, Version 13, C28-6390

MAD and FORTRAN manuals for non-staff members may be purchased at the COOP.

The Program Consultants, Room 26-151, extension 4114, are available to help with any CTSS programming question. Any mechanical trouble with the console should be directed to extension 4128.

FOOTNOTES

(1) Line-marked files may be created with the EDL command. See section AII.3.07.

(2) Bug Example

Input in FORTRAN IV		Output in HAP
bug		
IF (A)B(1)=3.	:	W'R (A)B(,1)=3.
fix-up		
IF (A)C=3. B(1)=C	:	W'R (A),C=3. B(1)=C

The subscripted variable should not be used in the indicated position of a logical IF.

(3) The entry points of the programs in F4LIBE BSS may be displayed on the console.

(4) The on-line instructions are listed together in the end section rather than in che descriptive text.

(5) Key to arguments in instructions is contained in this primer only. (IBM, etc. use other abbreviations.)

f = the label of a format statement if needed for conversion.

l = a list of variables.

n = an unsigned integer constant appearing as the secondary name of a .TAPE. disk file.

(6) These are public file commands. See section AJ.0 for a description of their use.

(7) The facility for initiating a job requested to be done after the user has logged out is provided by the FIB command.