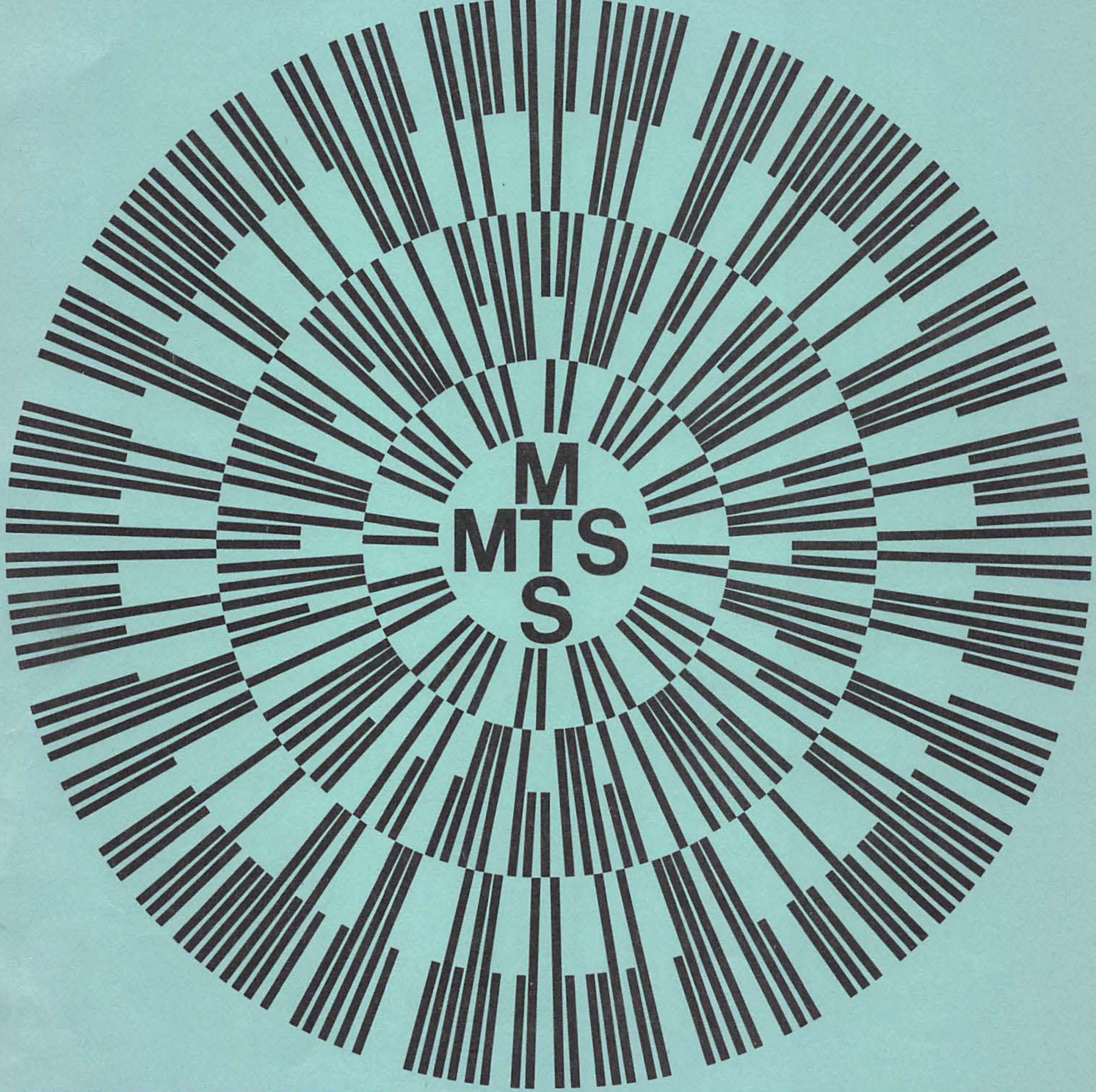
DAVIS

THE UNIVERSITY OF ALBERTA
COMPUTING CENTER
PUBLICATION



BATCH USERS' GUIDE

ACKNOWLEDGEMENTS

This manual was largely compiled from material prepared by the staff of the University of Michigan Computing Center. Their documentation was invaluable and we are indebted to them for allowing us to use it. In particular, the following were most useful:

MTS Users' Manual, Second Edition, Volumes I and II

MTS Users' Manual, Third Edition, Volume 2

Introduction to MTS and the Computing Center (Flanigan)

Computing Center News Items

Computing Center Memos

The Computing Center wishes to personally acknowledge the assistance of Mike Alexander and Don Boettner who helped us to establish MTS at the University of Alberta.

Acknowledgement should also be made to the Computing Centre, University of British Columbia, for information obtained from some of their documentation and to I.B.M., whose manuals provided certain sections for our manuals.

DISCLAIMER

This MTS manual is a combination of earlier manuals, update notices, memos and limited experience with the system itself. Because of this, certain discrepancies are bound to occur and the Computing Center would appreciate being notified of all differences between what this manual says and what the system actually does.

This publication is intended to represent the current state-of-the-system. However, it should not be construed as an obligation to maintain the system as so stated. The MTS system, like most good systems, is continually being improved. As a result, additions, extentions, changes and deletions will occur. Notice of such changes will be made and provision for a manual updating service has been planned.

Errors, comments and suggestions should be sent to:

Information Coordinator
Computing Center
University of Alberta

BATCH USERS' GUIDE

TABLE OF CONTENTS

1.0 Introduction 1.1

2.0 MTS Input Decks 2.1

3.0 The SIGNON Command 3.1

- Limit keyword parameters 3.1
- HASP keyword parameters 3.2
- Passwords 3.3

4.0 Job Submission 4.1

5.0 Comments on Batch Execution 5.1

- Pseudo-device Assignments 5.1
- $DESTROY and $ENTRY Commands 5.1
- Creating files 5.1
- Handling of error returns from executing programs 5.1
- Advantages and Disadvantages of batch 5.2

6.0 Examples of MTS Batch Jobs 6.1

1.0 INTRODUCTION

In batch mode the user prepares a card deck which contains all the MTS commands, translator source statements, data lines, and object decks which are needed in order to accomplish the desired task (or tasks). Once the user has checked his job deck for accuracy, he may submit the deck for processing in MTS in batch mode. The deck is read into the computer and saved in a file for later processing, and the deck and a receipt are returned to the user. The deck is then processed by MTS, together with many other batch decks, and eventually the results of this processing (the job output) may be retrieved by the user with his receipt. (The specific details of submitting batch decks and retrieving job output are given in a later section of this manual.) To provide a better understanding of this batch feature, we now proceed to a brief description of the program which monitors the batch function.

The MTS batch function is controlled by a program known as HASP (<u>H</u>ouston <u>A</u>utomatic <u>S</u>pooling <u>P</u>riority System). HASP was originally developed for the IBM OS system, but has been revised to interface with the MTS system and accept MTS jobs to run in batch; OS jobs cannot be run in the MTS system. A batch job in HASP is processed in five phases: input phase; execution phase; print phase; punch phase; and purge phase. At any one instant, there are normally several batch jobs in each phase competing for the use of the various hardware - software facilities. During the input phase, HASP reads in the job deck from the cardreader and stores it on a disk file to await execution. After the job deck has been read in its entirety, an entry is made on the HASP job queue which indicates the job is now ready for execution in MTS. At the beginning of the execution phase, HASP starts up an MTS task specifying the input file containing the job deck and specifying output files (for print and punch output); a print file is always needed by a batch job, but a punch file is set up only if the batch job indicates that punch output is to be produced. When the job execution is terminated, HASP is so informed, the MTS task is cancelled, and the disk file containing the job deck input is released; at this point execution is over but the job output (print and punch) is still in the specified disk files. After the execution phase, the job is placed in the print queue to initiate the print phase; during the print phase, the job print output is produced on an attached printer under HASP control. After the print phase, the job is placed on a punch queue if it produced punch output; in this case the job enters the punch phase during which its punch output is produced on an attached card punch. After the punch phase, or after the print phase if the job produced no punch output, the job enters the purge phase; here the job is placed on the purge queue and finally purged from the system. During the purge phase all disk files set up by HASP for the job are released, if they have not already been released. The various disk files set up by HASP are actually pseudo-files (i.e., they are not the same as actual MTS files); for convenience, however, this manual will discuss HASP files as if they were actually MTS files. As far as the function served of these HASP files is concerned, this distinction is not relevant.

As mentioned earlier, at any one time there are normally batch jobs in each of the five phases competing for hardware-software facilities. Hence, at any given instant, HASP must select the next job to be given processing time via some selection criteria. To do this task, HASP uses a priority basis to select jobs for processing; at any given time, HASP selects the first available batch job of highest priority for processing. The priority of each job depends upon several factors. In the execution phase, job priority is based on the time and output estimates on the $SIGNON command; the lower the estimates, the higher the priority. Hence, accurate estimates may speed batch job processing. To ensure that a job with large estimates does not remain in the system forever, an aging factor is also used in determining priorities; a job's priority slowly increases as its length of time in HASP increases. In the print phase, a new priority is computed for the job; this priority is based upon the actual number of printed pages produced by the job during its execution. This new priority is used in scheduling the job during the remaining HASP phases through which it passes.

It should be noted that HASP is only one of several programs running on the Model 67 concurrently; the timesharing aspects of the machine apply to HASP as well as to MTS and to user programs in MTS. To control the competition between batch jobs and terminal jobs for hardware facilities, there is a maximum number of jobs which HASP will allow in each of the processing phases (other than the input phase). These maxima vary during the day and over the month dependent upon the character of the MTS processing input; at all times, the Center attempts to maintain a reasonable balance between batch processing and terminal usage in order to privide acceptable service to both user groups.

There is some question regarding the HASP aging factor in distribution 2.0!

2.0 MTS INPUT DECKS

An input deck in MTS is normally composed of three types of cards. Command cards are used to convey information and requests to MTS; this is done by providing MTS with a command name or a command name abbreviation on each command card. Command cards always have a dollar sign ($) punched in column 1 in a batch deck. Program cards include both source cards and object cards; source cards contain statements in some computer language and must be translated to machine language, while object cards contain only machine language statements. The format of source cards is determined by the particular translator which is to translate the source program; the format of object module cards is dictated by the system loader.

Data lines are lines containing the pertinent information for the user's program during its execution. The content and format of these cards depend completely upon the program being executed and the type of read statements used by the program. Samples of data cards will be given in later examples.

To allow input lines (command or data) in the source stream which are longer than a card, MTS has a convention for continuation lines. If a source line terminates with a minus sign (-), then MTS deletes the minus sign and takes the next source line as a continuation of the current source line. In batch runs, the minus sign must be in column 80 of the input card. Input lines are limited to a total length of 255 characters, i.e., three cards.

The physically first card of an MTS input deck must be an MTS $SIGNON command; this command is described in a later section of this manual. The physically last card of an MTS input deck is normally an MTS $SIGNOFF cammand; under some conditions, this command may be omitted. For beginning users of MTS, it is advisable to always use the $SIGNOFF command to terminate an input deck. Both the $SIGNON command and the $SIGNOFF command may be abbreviated as $SIG.

To create an end-of-file while running a batch job, a $ENDFILE command is placed at the point where an end-of-file is needed. There is an automatic end-of-file at the end of the batch job.

Any object decks in the input deck should be terminated by either an LDT card (see the LOADER MANUAL) or a $ENDFILE.

3.0 THE SIGNON COMMAND

The physically first card on any MTS batch input deck must be a command card with a $ in column 1 and the letters SIG in columns 2-4 or the letters SIGNON in columns 2-7. The letters constitute the name of the command to MTS, and this name must be followed by one or more blanks. Because all input decks must start with this command, and to help in separating various input decks, orange cards are used only as the first card of a deck. These cards may be found in the keypunch room; they are the standard OS/360 job control card.

The SIGNON command directs MTS to accept the following deck as a job to be executed and supplies certain pieces of information to MTS. The full command should appear something like:

```
$SIGNON□CCID□'comments'
```

where

□ stands for 1 or more blanks; there should be blanks only at these points on the card;

ccid is the four-character Computing Center identification number assigned to the user; this number is obtained by filling out an application at the Computing Center General Office (normally, for a course, the instructor obtains all the numbers for the course);

comments are enclosed in primes ('); this must be the last field on the $SIGNON command. It is strongly recommended that the user place his name here.

In a batch run, immediately following the $SIGNON command, there must be a card which contains the user password (see the next section) in its initial columns, and which is otherwise blank. If CCID is acceptable to MTS and the password is correct, the deck is accepted as a job to be executed and processing of the job proceeds. The job is allowed to continue for 30 seconds of CPU time and/or 50 pages of output (including translations), at which time the job is terminated if it has not already itself caused termination. These default time and page estimates may be changed by placing new estimates on the $SIGNON command (see the next paragraph); generally, students jobs, and many other jobs, will run within these limits and should not change the limits. Since each CCID is given a maximum charge when it is assigned, it is to the advantage of the user to use good estimates to prevent loops which waste major amounts of CPU time. Furthermore, due to the HASP priority scheduling of batch jobs, good estimates may speed the MTS processing of batch jobs.

Limit Keyword Parameters

For those jobs in which it is necessary to change the time, card and/or page limits, this may be done with additional entries after the CCID, and before the name field .

An estimate for the total CPU time (not total elapsed time) for the job is specified by

$\underline{T}$ime = n $[\frac{s}{m}]$

where n is the time (integer or decimal) in Seconds or Minutes.

An estimate for the total number of pages of output for the job is specified by

$\underline{P}$ages = i

where i is the number of pages.

An estimate for the total number of cards to be punched for the job is specified by

$\underline{C}$ards = i

where i is the number of cards

The limit specifications may be given in any order on the $SIGNON command.

The following examples will clarify the use of limit specifications.

```
$SIGNON□CCID□T=55□'name'
```
sets the time limit to 55 seconds.

```
$SIG□CCID□P=25□T=0.75m□C=100□'name'
```
sets limits of 25 pages, 45 seconds and 100 cards.

The global time and/or page estimates apply to all phases of the job, including the time used and the pages produced by the translators. It is also possible to control the local time and/or page estimates for each object deck run in the system by placing such estimates on the pertinent $RUN commands.

HASP Keyword Parameters

An additional entry on the $SIGNON command is

```
COPIES=n
```

This causes n ≤ 255 copies of the output to be produced when the batch job is run under HASP. The copies include the lead and tail sheets, and the charges on the tail sheet reflect the charges for producing <u>all</u> of the copies.

Passwords

When a CCID is assigned to a user, a maximum charge (in dollars) and a maximum disk space (in pages) are provided for that CCID. The user running under a CCID may not exceed these limits; when the charge is used up, the CCID is no longer accepted by MTS. To prevent someone else from using (accidently or otherwise) a CCID, the user should attach to the CCID a password of the user's own choosing. Once assigned, a password protects a CCID by not allowing any use of the CCID by persons who cannot provide the current password. A password may consist of 1 to 6 non-blank characters; any printing characters are acceptable.

To assign a password, the MTS command

```
$SET□PW=xxxxxx
```

should occur in the deck; from the moment this command is executed by MTS, the CCID used to sigon the job has the password xxxxxx. The Computing Center assigns a password to each CCID at the time the CCID is itself assigned to the user; the user must therefore use this particular password the first time he runs in MTS with the pertinent CCID. Thereafter, the user must always use whatever password he has most recently assigned to a CCID via the $SET command. As stated previously, a password is provided in batch runs by keypunching the password, beginning in column 1, in the first card after the $SIGNON card; the remainder of this card is blank. The first two cards of a batch deck should therefore be something like:

```
$SIGNON CCID 'name'
password
```

For protective purposes, the printing mechanism of the keypunch should be turned off (don't interpret) when punching the password. In the output produced by the program, the password card will not be printed so that such output will nowhere contain a listing of the password. To protect the password and CCID combination, a source deck containing this information should not be left in a public place (such as the Computing Center workroom). When and if you throw away a batch input deck, be sure to remove the password card from the deck before dumping the deck in a wastebasket. Also, in a batch run, it is advisable to prevent the printing of any $SET commands which are used to set new passwords for a CCID. To do this, the following sequence of MTS commands should be given:

```
$SET ECHO=OFF
$SET PW=xxxxxx
$SET ECHO=ON
```

The first and third commands in this sequence control the printing of MTS commands in the batch run output. When ECHO is set OFF, MTS commands are not printed in the batch run output; when ECHO is set ON, MTS commands are printed in the batch run output. The net effect of the above sequence of three commands is that the current password is changed to xxxxxx, but the MTS command which contains the new password is not printed in the batch run output.

The $SET command may be used as often or as infrequently as the user desires; thus, the user may vary his password as often as he feels necessary. Since the Computing Center does not accept responsibility for time lost or files lost under non-password-protected CCID's, each user should use a password. Also, each user should check the initial MTS message on each run; this message includes the comment

```
     LAST SIGNON WAS ...
```

This may help indicate illegal use of the pertinent CCID. If at any time a user suspects that illegal use is being made of his CCID, he should immediately change his password, taking care to keep the new password protected, and he should report his suspicions to the Computing Center.

4.0 JOB SUBMISSION

Submit your deck at the DECK INPUT window; the deck will immediately be run through a card reader (under HASP control - see the introduction to this manual) and returned to you with a card in front of the deck. This card bears a number (your _job number_) which identifies your job within MTS; the card also serves as a receipt so that you must use it to retrieve your output. When your job has been processed, you may then retrieve your output by presenting your receipt at the PRINTER OUTPUT window. The time for this job processing by MTS from input of your batch deck to retrieval of your output will normally be of the order of a few minutes, but under heavy system loads this time may be a half hour or more. If you are not able to wait for your job output to become available, you may return to the Center later in the day, or the next day, to retrieve your job output. Except in cases of long machine downtime, batch jobs are almost always run the following morning after submission.

5.0 COMMENTS ON BATCH EXECUTION

Pseudo-device Assignments

The pseudo-device names *MSOURCE* and *SOURCE* initially refer to the HASP input file containing the input deck.

The pseudo-device names *MSINK* and *SINK* initially refer to the HASP printer output file assigned to an MTS batch job.

The pseudo-device name *PUNCH* refers to the HASP punch output file assigned to an MTS batch job.

$DESTROY and $EMPTY Commands

No confirmation is required for these two commands when they are executed from a batch job.

Creating Files

If a batch job creates and enters data into a file, and the job must be rerun, the following procedures for creating files should be followed to insure that all runs are identical.

```
Use either     (1)    $CREATE  Fname
                      $EMPTY   Fname

or             (2)    $DESTROY Fname
                      $CREATE  Fname
```

Handling of Error Returns from Executing Programs

A program, executed as a result of $RUN or $LOAD and $START, is responsible for posting a completion code in register 15 when returning to MTS on program termination. A zero code indicates a normal termination and a non-zero code, usually, indicates an abnormal termination.

Normally, MTS takes no action on abnormal termination and subsequent commands in the job are executed. However, the user can cause a dump of the registers and storage to occur automatically when an abnormal termination is encountered by MTS. This is done by either the $ERRDUMP or the $SET ERRORDUMP=ON commands. Note, however, that the job is not terminated and subsequent commands are executed.

The FORTRAN G compiler has a conditional execution facility. The EXEC parameter may be used to terminate a job due to unsuccessful compilation. After the last source program is compiled if the completion code is equal to or greater than the value of the EXEC parameter the job will be terminated immediately and the subsequent commands will <u>not</u> be executed.

Advantages and Disadvantages of Batch

Since there are restrictions on the direct use of line printers, card readers and card punches by a user signed on at a terminal, a user who wishes to accept input from a card reader or direct output to a line printer or card punch must use the batch facility. This can be done by either submitting a batch job or running *BATCH from a terminal to create a batch job (see the writeup on *BATCH in the Terminal Users' Guide).

Batch may often be more economical than an interactive use of MTS. Since the charge for a job is based, in part, upon elapsed real-time, the terminal user will find that he is being charged something for just sitting and thinking. When running in batch, not only does the card reader not have to think about the next line to input but the input-output rates are much higher in batch.

A disadvantage may arise in batch when the user wishes to compile and execute in the same job. Since there is presently no way for a run command to be made conditional, every run command will be executed even though the compilation which was to produce the object file was not successful. Usually a program interrupt occurs early in the execution of this file. This effect may or may not be desirable but the user should be aware of it.

6.0 EXAMPLES OF MTS BATCH JOBS

1. Sample FORTRAN compilation and run.

```
$SIGNON  X007  'JAMES BOND'
password
$RUN *FORTG SPUNCH=-OBJECT
       .
       .
       .
       FORTRAN program
       .
       .
       .
$ENDFILE
$RUN -OBJECT 1=*SOURCE* 2=*SINK* MAP=*SINK*
       .
       .
       .
       data
       .
$ENDFILE
$SIGNOFF
```

Comments: This job will compile the FORTRAN program in the source stream, putting the listing on the printer and the object program in a temporary file called -OBJECT. The object program is then run producing a map on the printer. The program will use logical unit 1 for input from the job stream and write the output on logical unit 2 which will be the printer.

2. Batch job to initialize the file PHROG from cards in the input stream.

```
$SIGNON P314  'G.J. NOHOPE'
$CREATE PHROG
$EMPTY PHROG
$NUMBER
   .
   .  cards to go into file
   .
$UNNUMBER
$SIGNOFF
```

Comments: The $EMPTY command is only important if the job is to be rerun.

3. Batch job to list the file LISTING-B on the printer.

```
$SIGNON Q123  'FIFO STACK'
$LIST LISTING-B
$SIGNOFF
```

4. Batch job to assemble a program, punching the object deck produced

```
$SIGNON  XXXX C=100 'L. USER'
$RUN *ASMG SPUNCH=*PUNCH*
    .
    .
    .
    assembly input
    .
    .
    .
$SIGNOFF
```

5. Batch job to run the object deck produced in the previous example. No map is wanted, and the program will read from SCARDS and print on SPRINT.

```
$SIGNON  XXXX  TIME=20 PAGES=62 'L. USER'
$RUN
    .
    .
    .
    object deck
    .
    .
    .
$ENDFILE
    .
    .
    .
    data for the program
    .
    .
    .
$ENDFILE
$SIGNOFF
```