TOPS-10/TOPS-20 Batch Reference Manual

AA-H374B-TK

November 1988

This document describes user procedures for the TOPS-10 and TOPS-20 batch systems.

This manual supersedes the manual of the same name and order numbers, AA-H374A-TK, AD-H374A-T1 and replaces the manuals *Getting Started With Batch (TOPS-20)*, AA-C781B-TM and *Getting Started With Batch (TOPS-10)*, AA-D303A-TB.

Change bars in margins indicate material that has been added or changed since the previous printing of this manual.

Operating System:	TOPS-10 Version 7.04 TOPS-20 Version 7.0
Software:	TOPS-20 GALAXY Version 6.0 TOPS-10 GALAXY Version 5.1

digital equipment corporation
maynard, massachusetts

First Printing, January 1980
Updated, July 1982
Revised, November 1988

CONTENTS

PREFACE

The TOPS-10/TOPS-20 Batch Reference Manual describes the commands for controlling jobs operating under the batch system. As a reader of this manual, you are expected to be familiar with the operating system commands available to timesharing jobs and with the commands to the various system programs that you will use. That reference material is not included in this manual. Therefore, it is recommended that you use this manual with the system commands manual for your particular operating system. The REFERENCES section at the end of this preface lists all referenced documents.

SYNOPSIS

Chapter 1 briefly introduces the batch system.

Chapter 2 describes the batch control file, the file you create (directly at the terminal or indirectly with cards) that determines how the system processes your job. This chapter also gives a detailed description of the batch controller and its associated commands.

Chapter 3 discusses the monitor commands available to you for controlling your batch job. Among other things, the chapter shows how to examine the batch input queue and the print queue and how to modify or cancel a job.

Chapter 4 illustrates methods for submitting a batch job from a timesharing terminal and discusses the SUBMIT command in depth.

Chapter 5 explains how to submit a job from cards and gives a detailed description of batch control cards.

Chapter 6 describes batch system output.

TOPS-10/TOPS-20 INFORMATION

The batch system operates in the same way under TOPS-10 and TOPS-20, employing the same set of batch commands and control cards for both systems. However, because of operating system differences, portions of this manual are described separately for TOPS-10 and TOPS-20. Chapter 3, for example, provides two sets of job control commands: one for TOPS-10, another for TOPS-20.

The system prompt character is different on the two systems. This character (. for TOPS-10; @ for TOPS-20) precedes all operating system and batch commands. Many examples throughout this manual include these commands; but, unless stated otherwise, these examples can apply to either system. Simply substitute the applicable prompt character for the existing one when necessary. You can usually take this same action when batch or system commands appear within the text.

CONVENTIONS

The following is a list of the symbols and conventions used in this manual:

$	A dollar sign.
n	A decimal number.
o	An octal number.
x	An alphabetic character.
?	An alphanumeric character.
|item |item |item	A symbol indicating that one of the items must be selected.
[ ]	A symbol indicating that items within the brackets are optional.
[|]	Therefore, the effect of a lined list of items enclosed within brackets is that it is acceptable to have any or none of the elements.
. @	A symbol indicating that either the TOPS-10 (.) or TOPS-20 (@) system prompt character is applicable.
command line/switches	A notation indicating that one or more run-time options [switches] from a nearby list can be specified with the command. Switches sometimes appear in the middle of a command line, as indicated in this manual. Each switch must be preceded by a slash (/).

REFERENCES

As stated earlier, this manual chiefly describes the commands for controlling jobs operating under the batch system. If you wish more information about commands to the system, refer to the operating system commands reference manual for your system (referred to within this document as the system commands manual).

Error messages from system programs that your job invokes are explained in the appropriate manuals. For example, you can find the descriptions of the error messages you receive from the FORTRAN compiler in the FORTRAN reference manual.

Other documents referenced in this manual or that might prove useful to you are:

TOPS-10:

- o <u>ALGOL Programmer's Guide</u>
- o <u>APLSF Language Manual</u>
- o <u>BASIC Conversational Language Manual</u>
- o <u>BLISS-10 Programmer's Reference Manual</u>
- o <u>COBOL-68 Language Manual</u>
- o <u>COBOL-74 Language Manual</u>
- o <u>FORTRAN-10 Programmer's Reference Manual</u>
- o <u>Introduction to DECsystem-10 Software</u>
- o <u>TOPS-10 MACRO Assembler Reference Manual</u>
- o <u>SOS Reference Manual</u>
- o <u>TECO Reference Manual</u>

TOPS-20:

- o <u>ALGOL Programmer's Guide</u>
- o <u>APLSF Language Manual</u>
- o <u>BASIC-PLUS-2 Language Reference Manual</u>
- o <u>COBOL-68 Language Reference Manual</u>
- o <u>COBOL-74 Language Reference Manual</u>
- o <u>EDIT Reference Manual</u>
- o <u>FORTRAN Reference Manual</u>
- o <u>TOPS-20 MACRO Assembler Reference Manual</u>
- o <u>TOPS-20 User's Guide</u>

CHAPTER 1

INTRODUCTION

The batch system is a group of programs that allows you to submit a job to the TOPS-10 or TOPS-20 system on a leave-it basis. (Refer to Appendix D, Batch Components, for a complete description of programs that constitute the batch system.)

You may build and submit your job in one of two ways:

1. By entering your data directly to an interactive computer system by means of a timesharing terminal.

2. By entering your data from punched-cards to the interactive system. The cards are given to an operator, who, at an appropriate time in his schedule, enters them into the computer through a card reader.

After preparing the job, you are free to leave the system. Upon accepting the job, the system classifies it in terms of size, running time, the need for peripherals, and so forth. This classification is used as the basis for determining when the job is to be run. Large jobs may, therefore, be set aside until smaller or more urgent jobs are finished.

Some of the jobs that are commonly processed through the TOPS-10 or TOPS-20 batch system are those that:

1. Are frequently run for production

2. Are large and long running

3. Require large amounts of data

4. Need no actions by you when the jobs are running

Batch allows you to submit your job to the computer through either an operator or a timesharing terminal, and to receive your output from the operator when the job has finished. Output is never returned to your timesharing terminal even if your job is entered from one. Instead, it is sent to a peripheral device (normally the line printer) at the computer site and returned to you in the manner designated by the installation manager.

1.1 JOBS

In timesharing, the term "job" generally refers to your allotment of computer time from login to logout. You create a job when you successfully log in. After you create a job, you can access any system resource to which you are entitled and execute any number of commands or run any number of programs. Actions that you initiate after login denote a job.

In batch processing, the term "job" generally refers to the entire stream (or list) of commands contained in a batch control file, commands that you would type on the terminal under timesharing. When you submit this file for processing, the batch system:

1. Creates a timesharing job for you.

2. Connects the job to the directory (-20) or project-programmer number (-10) where you are connected (-20) or logged-in (-10) at the time you submit the control file.

3. Sets defaults for switches if you have so specified (refer to Section B.1, SWITCH DEFAULTS).

4. Executes the commands stored in your control file, storing the results of its processing in a log file.

Upon completion, the batch system logs out the batch job.

In summary, this manual refers to "job" in all of the following senses:

1. As the batch control file.

2. As the system time available to you between the time you log in and log out.

3. As a unit consisting of one action or a group of actions that you initiate with the system; that is, as:

    a. A single user program and its related data, or several user programs and their data.

    b. A utility program or system program and its related data.

    c. The system, batch, and program-level commands that are required to control the execution of the job. These commands, when read into the batch system, constitute the control file for the associated job.

1.1.1 Running Multiple Jobs from the Same Directory

When working at a terminal, you can run batch and timesharing jobs at the same time from the same directory. However, this practice exposes your files to certain dangers. To avoid these dangers, remember that a batch job functions as a timesharing job, affecting system operations similarly. The following sections discuss the more common difficulties you might encounter when running multiple jobs from the same directory. Problems may arise when more than one batch job is run from the same directory (SUBMIT/UNIQUE:NO) or with timesharing jobs. Problem situations are not limited to batch and timesharing combinations, as these sections describe.

TOPS-20

Recall that a batch job is logged in and out as if it were a timesharing job. But whereas you log a timesharing job in and out, the system automatically logs each batch job in and out. To do this, the system executes LOGIN and LOGOUT, the same commands you use to log timesharing jobs in and out.

The LOGOUT command, whether executed from a batch or timesharing job, expunges all deleted files from your connected directory. This action could cause you problems. Suppose you are working with files under timesharing while at the same time running a batch job. When the batch job terminates, the system will expunge any files you deleted. The UNDELETE system command is ineffective for these files, just as it is for any expunged file.

TOPS-10 and TOPS-20

Exercise care in creating files when running batch and timesharing jobs from the same directory.

The system overwrites the contents of an existing file when you create another file (with the COPY command, for example) of the same name. In overwriting a file, the system assumes that you are keeping track of the files entered into your directory by batch and timesharing jobs, and that therefore the overwrite is desirable. Because you might not think of batch mode when operating in timesharing mode, and vice versa, you could unintentionally create a file with the same name as one created in the other mode. This would destroy the contents of the original file.

On a TOPS-20 system, the RENAME command also will destroy an original file as described above.

1.2 BATCH COMPONENTS

The batch system consists of a group of programs. Some of these programs are used for batch operations only; others are available for various operations of the total computing system. Three programs are strictly batch components: CDRIVE, BATCON, and SPRINT. The remaining programs belong to the whole system: EXEC (-20), NEBULA, GLXLIB, LPTSPL, MOUNTR (-20), CATLOG (-10), OPR, ORION, PULSAR (-10), QUASAR, QUEUE (-10), and SPROUT. Figure 1-1 shows the relationship among the components; they are described in Appendix D.

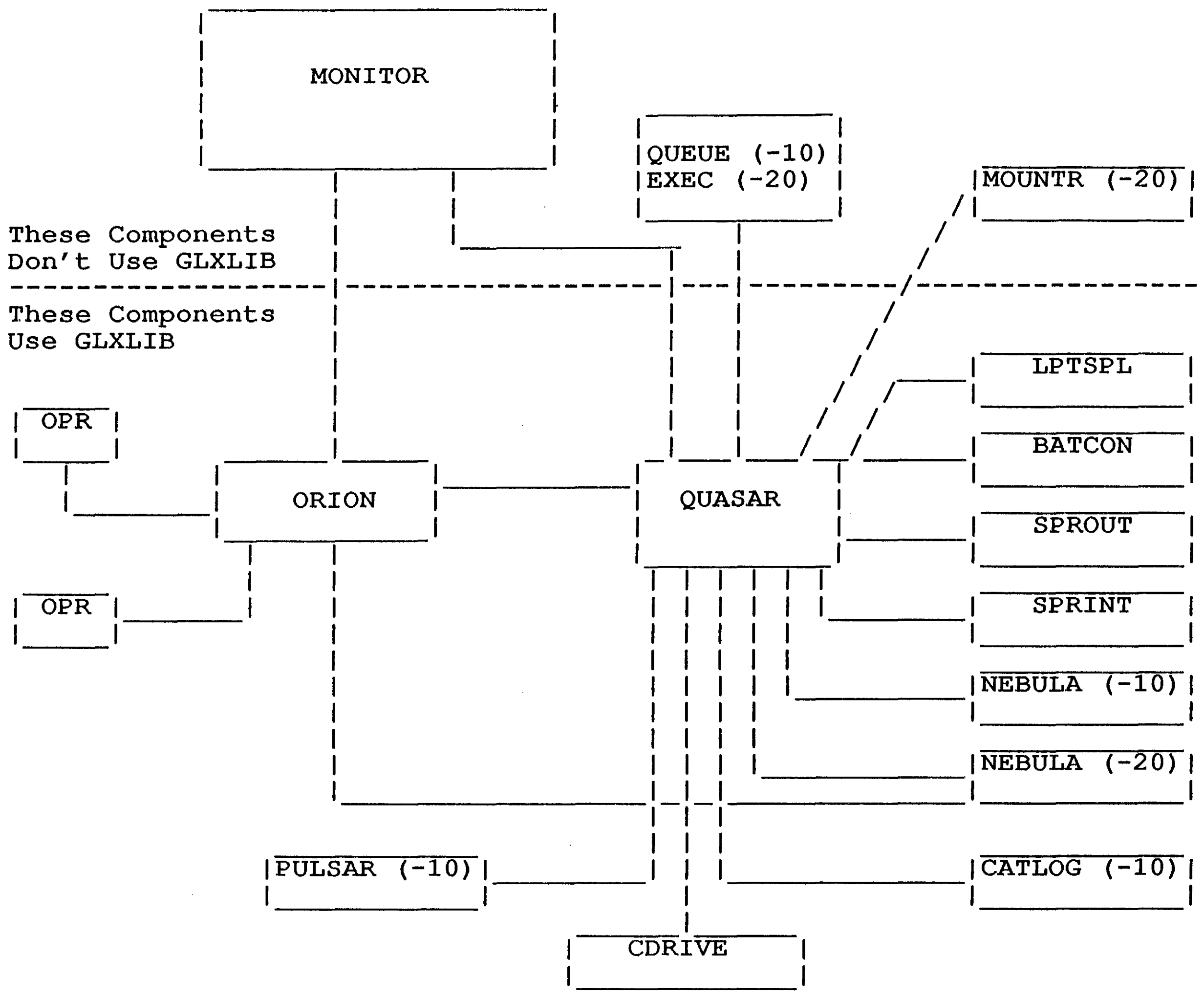

Figure 1-1: The Batch System

1.3 CONTROLLING THE BATCH ENVIRONMENT

You may take any of several actions to control the batch environment. These actions -- some required, others optional -- let the system know what programs and data it needs to process your job; guide the system in its operation; and affect the job as a whole in such ways that determine the day and hour (after you submit the job to the system) that processing is to begin, the directory under which the job is to run, and so forth.

You control the batch environment by issuing commands from either a control file or the terminal. Batch commands (described in Section 1.3.1.2), control card switches (described in Section 1.3.1.1.) and monitor command switches (contained in the control file and/or issued from the terminal) provide you with various run-time options.

A control file consists of commands that tell the TOPS-10 or TOPS-20 system what you want to process. The control file commands can be created as a disk file or as card input and can consist of:

1. System commands (See the TOPS-20 User's Guide or the TOPS-10 Operating System Commands Manual.)

2. System program commands to system programs

3. Batch commands (See Chapters 4 and 5.)

These commands, when submitted to the operating system, must be in a particular order so that your batch job will execute correctly.

The steps that you must take to create a control file from a timesharing terminal are described in Chapter 4. The steps to take to create a control file from cards are described in Chapter 5.

The following sections describe how you can control the batch environment.

1.3.1 The Control File

The control file consists of system and batch commands, and, if needed, program commands or program data. The control file you submit from the terminal, called the batch control file, is read directly by the batch system. A card deck containing control cards is preprocessed and a batch control file is created from the card input.

NOTE

You can submit a card deck stored on disk using the /READER switch with the SUBMIT command. Refer to Section 4.1.1, Submitting a Card Job from a Timesharing Terminal, for a description of /READER.

The control file specifies the programs and data required for your job and, through the batch commands, directs the job's activities. A control file must exist for every batch job.

1.3.1.1 Control Cards - Batch jobs that you submit on cards contain control cards indicating where your programs and data files are stored (possibly within the card deck) and how they should be processed. SPRINT interprets the control cards, writes card-resident data and program files to the devices you specify, and creates a batch control file. The batch control file is passed to the batch processor, also known as the batch controller (BATCON).

1.3.1.2 Batch Commands - The batch commands give you control over the batch run-time environment by allowing, among other options: error recognition and recovery, program-like functions within the control file (GOTO, BACKTO, IF), sending messages to the operator, and dialogue between the operator and the job. Batch commands do not corrupt your core image. Chapter 2, THE BATCH CONTROL FILE, describes the batch commands in detail.

1.3.2 Job Control Commands

You can use the job control commands to control the entire job as a unit: to submit batch jobs to the system for processing, to submit jobs to output devices, to modify the parameters for these jobs, and to cancel or kill the execution of batch or output jobs. The job control commands, SUBMIT, PRINT, MODIFY (-20), and CANCEL, are discussed in Chapter 3, JOB CONTROL.

1.3.3 Running Your Job

After you submit the job, it waits in a queue with other jobs until the batch system schedules it to run under guidelines established by the installation manager. Several factors affect how long your job waits in the queue: for example, its estimated execution time and the priority of your job compared with other waiting jobs.

When the job is started, batch reads the control file to determine what actions are necessary to complete the job. For example, if there are commands to the system programs, batch issues the commands to those programs. Any output produced as a result of those commands is stored in a log file for listing later. With adequate planning, the control file can also provide for corrective actions in the event of errors.

As each step in the control file is performed, batch records it in a log file. For example, if a system command such as COMPILE is executed, batch passes it to the system and writes it in the log file. The system response is also written in the log file. Batch writes in the log file any response from your job that would have been written on the terminal if the job were run interactively.

1.3.4 Receiving Your Output

Your program output will be returned to you in the form that you specified by the commands in your control file. This is normally the line-printer listing, but may also be output on magnetic tape or disk. When your output is directed to the line-printer, you may specify, in the SUBMIT command to the batch process, the approximate number of pages (/PAGES: switch) that you require. This will help batch restrain runaway programs.

If your batch job is submitted through a timesharing terminal, the log file is written and saved on disk in your directory and printed on the line printer. If your batch job is submitted on card input, the log file is written on disk in your directory, printed on the line printer, and then deleted from your directory.

1.3.5 Recovering from Errors

If an error occurs in your job, either within a program that is executing or within the control file, batch writes the error message in the log file and usually terminates the job. You can, however, include commands in the control file to direct batch to branch to recovery sequences in the event of an error and thereby allow completion of the job. The effectiveness of error recovery is dependent on your ability to predict potential trouble spots within the program or within commands used in the control file. (Refer to Section 2.5, ERROR PROCESSING, for detailed descriptions on error recovery for batch jobs.)

1.4 SUMMARY

To enter a batch job:

1. Create a control file either on cards or from a terminal.
2. Submit the job to batch, either indirectly via the operator (for a card job) or directly from a terminal.

Afterwards, obtain and examine the log file listing and the job output to determine if the desired results were obtained.

Sample jobs run through batch from cards and from a terminal are shown in Sections 4.4 and 5.5, SAMPLE JOBS.

CHAPTER 2

THE BATCH CONTROL FILE

2.1 INTRODUCTION

From a user point of view, a batch job is just like a timesharing job. A batch job logs in to the system, types commands, runs programs, and logs out. The major difference between batch and timesharing is that, in a batch job, the batch processing commands that are typed come from a disk file rather than from a timesharing terminal and the typeout from the system and programs is written to a disk file rather than to a timesharing terminal. The disk file that is used to provide 'terminal' input to a batch job is called the control file, and the file used to record the 'terminal' output is called the log file (described in Chapter 6, BATCH SYSTEM OUTPUT).

The control file contains all of the system commands and program commands that you would normally type to perform the same task from a terminal. For example, if you wanted to compile and execute a program called MYPROG.CBL, the type-out on a timesharing terminal would appear as follows:

```
.COMPILE MYPROG.CBL            (Your request)
CBL74:  MYPROG [MYPROG.CBL]
                               (The system's reply)
EXIT

.EXECUTE MYPROG.CBL            (Your request)
LINK:  Loading
[LNKXCT] MYPROG Execution]
                               (The system's reply)
EXIT
```

To create a control file on a TOPS-20 system to tell batch to run the same, you would type the following:

```
@CREATE MYFILE.CTL<RET>
INPUT:  MYFILE.CTL.1
00100   @COMPILE MYPROG.CBL<RET>
00200   @EXECUTE MYPROG.CBL<RET>
00300   <ESC>

*E<RET>

[MYFILE.CTL.1]
@
```

When the job is run, the commands are passed to the system to be executed. The commands and their replies from the system are written in the log file so that the entire dialogue shown in the first example above appears in the log file.

NOTE

You must not place the LOGIN command in the control file; it is handled by the batch system. The LOGIN command will produce an error message.

LOGOUT (-20) and KJOB (-10) are also handled by the system, but can be used, for example, in error recovery procedures.

There are a number of differences, however, between the contents of a control file and what you would type at a timesharing terminal. One difference is that lines in the control file may have labels (or tags) on them. Also, there are additional commands (called batch commands) available to provide control over the batch job that are not needed during a timesharing session. These additional commands direct the system to test for errors, branch to various lines in the control file, initiate and control messages to the operator, and provide a number of other functions.

This chapter describes these control file facilities. Refer to Chapters 4 and 5 for an explanation of how to create and submit a control file.

2.2 LINE IDENTIFIERS

In a timesharing environment, commands that you type at system command level are interpreted by the system command processor. Similarly, when you type commands at program level, the commands are interpreted by the program. For example, when you type the LOGIN system command, it is interpreted by the system command processor. However, when you type the EXIT command to TECO/EDIT, that command is interpreted by the TECO/EDIT program.

Since the batch system attempts to simulate the timesharing environment, each line of the control file is destined for interpretation by either the system command processor or the running program. To unambiguously specify the destination of a particular line in the control file, you may prefix the line with a single character called a 'line-identifier.'

A line identifier is the first character that occurs after a carriage-return/line-feed (CR/LF) in the control file. The batch system determines the destination of a line by interpreting the first character as follows:

Identifier	**Interpretation**
* (asterisk)	The batch controller interprets lines beginning with * as user data or as data to a system program, that is, as a program-level command. However, if the job is at system command level, the line is treated as a comment.

NOTE

TOPS-20:

Certain commands, such as COPY, leave your job at system command level. You should precede data lines for these commands with the at sign (@) rather than the asterisk, because these lines will be interpreted by the system command processor (EXEC). For example, in the control file lines

```
@COPY TTY: TEST.TST
@This is data for TEST.TST
@^Z
```

"This is data for TEST.TST" serves as data to the COPY command. This line is copied to the file TEST.TST from TTY:.

Other commands, such as EDIT, place your job at program level. Here, you should precede data lines with the asterisk so that the controlling program can interpret the lines appropriately.

= (equal sign)

The batch controller interprets this character in the same manner as it does the asterisk. However, the equal sign indicates that you want to inhibit the batch controller from transmitting the carriage-return/line-feed terminating this line. For example, in the DDT sequence

```
=FILE/
```

the "/" is a terminator to DDT. The "=" prevents an unwanted carriage return or line feed from being sent to DDT.

. (period)
@ (at sign)

These characters represent the system prompt characters for TOPS-10 and TOPS-20, respectively. The interpretation of these characters depends on the next nonblank, nontab character.

If that character is numeric, the line, including the line identifier, is treated as user data. If the job is at system command level, the line is treated as a comment.

If the character following the system prompt character is an alphabetic or special character (CTRL/C, CTRL/G, CTRL/Z, ALTmode or ESC, form feed, line feed, vertical tab), the line is treated either as a system command or a batch command as defined in Section 2.6, BATCH COMMANDS. If it is a system command, the job is placed at system command level, the line identifier is ignored, and the line is sent to the system. If it is a batch command, it is processed by the batch controller.

NOTE

TOPS-20:

If you are an enabled operator or an enabled wheel, precede privileged commands with an at sign (@) rather than a dollar sign ($). (A line beginning with a dollar sign is treated as a comment line.)

To specify subcommand level (for instance for the SYSTAT command), use a single at sign. Do not use a double at sign (@@). This produces an error.

See NOTE under the asterisk (*) line identifier for information about the at sign and system command data lines.

! (exclamation point) ; (semicolon)	A line beginning with an exclamation point or semicolon is treated as a comment line. Comment lines are sent to the log file only; they do not affect the running job.
% (percent sign)	This character identifies the labels reserved for use in the event of an error occurrence. (Refer to Section 2.5, ERROR PROCESSING, and 2.5.3, Reserved Labels.)
FF (form feed) VT (vertical tab)	These characters are treated as comments when they appear as the first character in a line of a control file. The character immediately following an FF or VT is then treated as if it were the first character of the line. Form feed and vertical tab do not affect the character that follows when they occur in any position other than the first character position in a line in a control file.

NOTE

This interpretation enables you to enter form feed and vertical tab into the control file to format the log file output.

Lines in the control file that begin with any other special character (including blank and tab) are treated in two ways:

1. If the job is at user level, the line is treated as user data, with the first character considered to be part of that data.

2. If the job is at system command level, the line is treated as a comment.

Any line in the control file that begins with an alphabetic character (and that is not a label; refer to Section 2.3, LABELS) is interpreted as follows:

1. If the job is running a program, the line is passed to the program.

2. If the job is at system command level, the line is treated as a batch or system command as defined in Section 2.6, BATCH COMMANDS.

NOTE

Because of the two possible interpretations, it is recommended that you begin all lines with the appropriate identifier to clarify their meaning.

The circumflex (^) is a special character that denotes control characters. Unlike the special characters described above, the circumflex is not necessarily in the first character position of a line. The batch controller interprets the circumflex as follows:

o If the character following the circumflex is numeric, the circumflex and the digit are passed to the job. For example:

```
^2 remains ^2
```

o If the character following the circumflex is alphabetic, the result is the corresponding control character (that is, the character formed by taking the six least significant bits of the binary representation of the alphabetic character). For example:

```
^C becomes CTRL/C
```

o If the character following the circumflex is another circumflex, the result is a single circumflex. For example:

```
^^C becomes ^C
^^^C becomes ^CTRL/C
```

o Some of the special characters are also affected by the circumflex and result in the corresponding control character. They are:

  Reverse slant(\).
  Closing bracket(]).
  Underscore(_).

o ^[results in ESCape.

NOTE

The circumflex itself is not among the listed special characters and CTRL/^ cannot be generated with the above-mentioned method.

2.3 LABELS

The control file can contain labeled lines. Labels identify particular lines in the control file so the lines can be referenced by such commands as GOTO.

The label, from one to six characters long, must begin with an alphabetic character and can be followed by up to five alphanumeric characters. Two colons (::) must terminate the label. A single line in a control file cannot contain more than one label. However, you may use the same label more than once in the control file.

START::@PRINT FILA.LST is an example of a labeled line using START as the label.

Also, a line may contain nothing but a label. For example,

```
     ERROR::
```

The character following the two colons is treated as if it were in the "first character" position as described in the preceding section. During processing, the label itself is skipped (it does not affect processing, but is merely a reference point), and the remainder of the line is treated as if the label were not there. Spaces between the label and the first character are ignored.

Five labels, %CERR (TOPS-10 systems only), %ERR, %RERR (TOPS-20 systems only), %FIN, and %TERR are predefined by the system for use in special cases. %CERR, %ERR, and %TERR mark special error recovery procedures for the batch job. %RERR marks special restart procedures. Although %FIN functions in several ways, BATCON also references it when performing error recovery. None of these labels can be referenced by such batch commands as GOTO. These five labels are called reserved labels and are described in Section 2.5.3.

2.4 COMMENTS

You may place comments in a control file. They may appear on lines containing batch or system commands, or as lines by themselves.

On command lines, the system identifies comments by an exclamation point (!) or semicolon (;) appearing after the command and before the text to be treated as a comment. For example, in the lines below:

```
               .
               .
               .
     @IF (ERROR)       !Proceed if error occurs
     @R FILCOM         !Compare
               .
               .
               .
```

"Proceed" and "Compare" begin the comments.

You can write a purely comment line into the control file as follows:

```
     ;This begins part 2 of the job.
```

2.5 ERROR PROCESSING

The batch controller tests for error indications in lines of output sent from user programs, from system programs, and from the monitor. The error signal is the question mark (?) appearing as the first character of an output line. (Note that the percent sign (%) is the error signal for the DELETE, DIRECTORY, FDIRECTORY (-20), VDIRECTORY (-20), and DISCARD (-20) commands.) You can declare a single additional (not alternate) error-signaling character by using the batch command ERROR. Use the NOERROR batch command to suppress error processing. (Refer to Section 2.6, BATCH COMMANDS for a description of ERROR and NOERROR.)

When the batch controller detects an error, it waits for the job to request more input. That is, any additional error messages or job output are written into the log file until the job is ready for another command. At this point, error processing begins.

If the error was ?TIME LIMIT EXCEEDED, special handling is involved (see Section 2.5.3.3, "%TERR"). For all other errors detected, the batch controller bypasses all lines indicated to be user-level input until it arrives at a line in the control file to be interpreted as either a system or batch command (that is, a system-level command). If this line contains an IF batch command, it is processed and execution of the control file resumes. (Refer to Section 2.6 for a description of the IF command.) If the batch controller encounters the appropriate reserved label (described below in Section 2.5.3) while searching for IF, it continues processing from the label. Also, if IF is not the next system-level command in the control file, the search begins for a reserved label.

NOTE

> The batch controller does not search past the next executable system command in the control file for the IF batch command. Therefore, if you use the IF command, there must be no intervening system commands and it must be the next batch command in the control file.

It is important to note that once an error has occurred, your job goes into an error state and remains there until the batch controller finds the appropriate reserved label or the IF command and thereupon clears the condition. If the error condition cannot be cleared, that is, if your control file does not contain a reserved label or the IF command, the job is terminated.

2.5.1 Specifying Error Recovery in the Control File

You can specify error recovery in the control file by means of the batch commands, especially the IF batch command. You must put the IF command at the point in the control file where an error may occur. When an error occurs, batch skips over all lines in the control file until it encounters a line beginning with a system-level command. If this line contains an IF command, the IF command is processed and the job continues. If this line does not contain an IF command or the appropriate reserved label, then the job is terminated. Therefore, if a batch job is to recover from an error successfully, the IF command must be placed in the control file where the error is expected to occur but before any other system-level commands. Thus, if you have a program that you are not sure is error free, you can include an IF command to tell batch what to do if an error occurs, as shown in the following example:

```
COMPILE MYPROG.FOR
IF (ERROR) statement
          .
          .
          .
```

In either the IF (ERROR) or the IF (NOERROR) command, you should include a statement that tells batch what to do. You can use any monitor command or batch command. If you wish to simply ignore the error without taking any special action, you may use a comment as the statement. The GOTO and BACKTO commands are also commonly used for this purpose. Refer to Section 2.6 for descriptions of these commands. Be sure, if you use GOTO or BACKTO in the IF command, that you supply a line in the control file that has the label that you specified in the GOTO or BACKTO command.

Two sample jobs are shown below. The first shows the IF (ERROR) command and the GOTO command to specify error recovery. The second example shows the use of the IF (NOERROR) and GOTO commands.

If you have a program that you are not sure will compile without errors, you can include another version of the same program in your job and tell batch to compile the second program if the first has an error. You write the control file as follows:

```
CREATE MYFILE.CTL<RET>
INPUT: MYFILE.CTL.1
00100  COMPILE /COMPILE MYPROG.FOR/LIST<RET>
00200  IF (ERROR) GOTO A<RET>
00300  EXECUTE MYPROG.FOR<RET>
00400  GOTO B<RET>
00500  A::!CONTINUE<RET>
00600  COMPILE /COMPILE PROG2.FOR/LIST<RET>
00700  EXECUTE PROG2.FOR<RET>
00800  B::!CONTINUE<RET>
00900  $

*E<RET>

[MYFILE.CTL.1]
```

When the job is run, batch reads the control file and passes commands to the system. If an error occurs in the compilation of the first program, batch finds the IF (ERROR) command and executes the GOTO command contained in it. The GOTO command tells batch to look for the line labeled A::. Thus, batch skips lines in the control file until it finds label A and then passes commands to the batch job from that point. If an error does not occur while compiling MYPROG, the GOTO A statement is not executed. Instead, MYPROG is executed and then batch skips to the line labeled B::.

A variation of the above procedure is shown below using the IF (NOERROR) command and the GOTO command. The difference is that batch skips the IF (NOERROR) command if an error does occur, and performs it if an error does not occur. The following is the control file that you would create:

```
CREATE MYFILE.CTL<RET>
INPUT: MYFILE.CTL
00100  COMPILE /COMPILE MYPROG.FOR/LIST<RET>
00200  IF (NOERROR) GOTO A<RET>
00300  COMPILE /COMPILE PROG2.FOR/LIST<RET>
00400  EXECUTE PROG2.FOR<RET>
00500  GOTO B<RET>
00600  A::!CONTINUE<RET>
00700  EXECUTE MYPROG.FOR<RET>
00800  B::!CONTINUE<RET>
00900  $

*E<RET>

[MYFILE.CTL.1]
```

When the job is run, batch passes the COMPILE command to the system to compile the first program. If an error does not occur, the IF (NOERROR) command and the GOTO command are executed, batch skips to line labeled A, which is a comment, and passes commands to the batch job from that point. The program MYPROG.FOR is executed and the end of the job is reached. If an error occurs while compiling MYPROG, batch skips the IF (NOERROR) command and continues reading the control file. PROG2.FOR is compiled and then executed. Batch is then told to go to the line labeled B, which is a comment line. The end of the job follows.

The examples shown above illustrate only two ways that you can use the IF commands to specify error recovery in the control file. You can use any of the batch commands or any system command to recover from errors in your job.

However, you do not have to attempt to recover from errors while your job is running. You can correct your errors according to the error messages in the log file when your job is returned to you, and run your job again.

2.5.2 Line Identifiers and Error Processing

Recall from Section 2.2, LINE IDENTIFIERS, that the batch controller sends a line to the appropriate destination by referring to the first character; thus it is important to follow the conventions for identifying lines. The following example illustrates the importance of including line identifiers in the control file.

```
R PROG1
line 1 of input
line 2 of input
line 3 of input
IF (ERROR)
R PROG2
```

If an error occurs at line 2 of the control file above, the system ignores the IF batch command even though it is the next batch command in the file and there are no intervening system commands. The batch controller, in searching for an IF batch command, assumes by default that line 3 is a system or batch command because in addition to lacking an identifier, the line begins with an alphabetic character and is unlabelled. Upon closer examination, the batch controller finds that the line is neither a batch nor system command, and it begins looking for reserved labels in the control file.

For correct interpretation under TOPS-10, this control file should be written as follows:

```
.PROG1
*line 1 of input
*line 2 of input
*line 3 of input
.IF (ERROR)
.R PROG2
```

On TOPS-20 systems, substitute an at sign (@) for the period (.).

2.5.3 Reserved Labels

The reserved labels, %CERR (TOPS-10 systems only), %ERR, %RERR (TOPS-20 systems only), %FIN, and %TERR indicate lines of code to be executed when various errors occur during processing of the batch control file. Like the ordinary labels discussed previously, the reserved labels serve as reference points in your control file.

If the IF batch command is not the next system-level command in the control file after a user- or system-program error occurs, the batch controller searches for one of the reserved labels. On TOPS-10 systems, the batch controller searches for %CERR if the error occurred in a system program (a program residing on the physical device SYS:); it searches for %ERR on a user-program error. On TOPS-20 systems the search is for %ERR when processing both user and system programs. %FIN satisfies the search for these labels; thus, the batch controller will continue processing from a %FIN label if it encounters the label while searching for %CERR or %ERR.

When a time-limit error occurs (?TIME LIMIT EXCEEDED message), the batch controller begins looking for %TERR in your control file. The IF batch command is ineffective for this error; that is, the batch controller does not process an IF command line. As with the other reserved labels, %FIN satisfies the search for %TERR and is the point from which processing continues if it appears before %TERR after a time limit error occurs. (Other uses for %FIN are discussed in the following sections.)

2.5.3.1 %RERR (TOPS-20 systems only) - In the event of a system failure, the batch controller searches for %RERR if the job is submitted with /RESTART:NO and there are no checkpoints in the control file or the first checkpoint has not been reached. If %RERR is not present, the search is for %FIN.

2.5.3.2 %CERR and %ERR - Error Packets - The reserved labels %CERR and %ERR are situated at the beginning line of their respective error packets. A packet, beginning with one of these labels and ending with %FIN, consists of as few or as many commands as you wish to execute for a particular error. Place packets at points in the control file where you think it is important to have an error-recovery routine.

Important to note is that the batch controller bypasses a packet when no error has occurred. During normal processing, the batch controller skips the %CERR and %ERR labels and all lines following them until it reaches %FIN. Then it executes the %FIN labeled line and continues processing the control file from that point. (A %FIN labeled line is always executed, regardless of an error or nonerror condition.) In this way, lines designated for error recovery are bypassed under normal conditions.

The batch controller does not read backwards over sections of the control file that it may have skipped during its search for error packets.

Example

The following TOPS-10 example shows the use of a %ERR error packet. You can apply the example to TOPS-20 by remembering that %ERR serves the same function as the TOPS-10 %CERR label and by noting that some of the commands may differ between the systems. The general control file format and system behavior, as explained below, agree between TOPS-10 and TOPS-20 systems.

```
.COMPILE SAMPLE/LIST                               !1.
.MOUNT MTA:TAPE:/REMARK:42936/VOLID:42936          !2.
.EXECUTE                                           !3.
.DISMOUNT TAPE                                     !4.
.R SORT                                            !5.
*MUMP.SRT=FOR04.DAT/KEY:1:10/RECORD:80             !6.
.PRINT MUMP.SRT                                    !7.
%ERR::.CLOSE                                       !8.
.DUMP                                              !9.
.DIRECTORY                                         !10.
%FIN::.DELETE FOR04.DAT                            !11.
```

Depending on the type of error it finds in the preceding control file the batch controller performs the following operations. The numbers to the left of the operation correspond to the line of the control file being discussed.

If a compilation (system) error occurs:

1. Compilation begins.
1. An error occurs.
2. The batch controller searches for %CERR or %FIN in the absence of an IF batch command on this line.
11. %FIN is found so the file FOR04.DAT is deleted.

If an execution (user) error occurs:

1. The program is compiled.
2. The tape is mounted.
3. Execution begins.
3. An execution error occurs.
4. The batch controller searches for %ERR or %FIN in the absence of an IF batch command on this line.
8. %ERR is found.
8. The output is closed.
9. The DUMP batch command is executed.
10. A directory listing of the user's disk area is produced.
11. The file FOR04.DAT is deleted.

If a SORT (system) error occurs:

1. The program is compiled.
2. The tape is mounted.
3. The program is executed.
4. The tape is dismounted.
5. The SORT program begins.
6. A command to SORT is given.
6. A SORT error occurs.
7. The batch controller searches for %CERR or %FIN in the absence of an IF batch command on this line.
11. %FIN is found so the file FOR04.DAT is deleted.

A SORT error could occur on line 5. It is an unlikely event, but it can be used here to further illustrate the batch controller's actions. When an error occurs on line 5, the batch controller examines line 6 in search of the IF batch command. Because line 6's identifier (first character) shows that the line serves as input to the SORT program, the search moves to line 7 where processing continues as in the SORT error described above. (Refer to section 2.5.2, Line Identifiers and Error Processing, for related information.)

If no errors result:

1. The program is compiled.
2. The tape is mounted.
3. The program is executed.
4. The tape is dismounted.
5. The SORT program begins.
6. The command to SORT is executed.
7. MUMP.SRT is queued for printing.
8. %ERR is encountered, and therefore the batch controller searches for %FIN to bypass the error packet.
11. %FIN is found, and the file FOR04.DAT is deleted.

Finally, if at any time during processing, a ?TIME LIMIT EXCEEDED error occurs, control is transferred to line 11. Refer to Section 2.5.3.3, %TERR.

User-Error Processing - If an error occurs in a user program and you did not include an IF batch command to handle the error, the batch controller searches for %ERR. Further processing depends on the success or failure of the search as follows.

1. If %ERR is found, the error packet is processed.
2. If %FIN is found before %ERR is reached, the batch controller issues an error message and continues processing from %FIN.
3. If the end of the control file is reached before either %ERR or %FIN is encountered, the batch controller issues an error message and terminates the job.

NOTE

In cases 2 and 3 above, the batch controller executes the DUMP batch command.

System-Error Processing - If an error occurs in a system program and you did not include an IF batch command to handle the error, the batch controller acts exactly as described above for user errors with two differences:

1. With TOPS-10, %CERR is used instead of %ERR. (The program resides on the physical device SYS:.)
2. No dump is initiated.

With TOPS-10 or TOPS-20 systems, BATCON issues an error message when it encounters %FIN or reaches the end of the control file in its search for %CERR or %ERR. See Chapter 6 for explanations of the BATCON error messages.

2.5.3.3 %TERR - You can use the %TERR reserved label to handle time-limit errors. When the ?TIME LIMIT EXCEEDED message is issued, indicating a time-limit error, the batch controller gives the job an additional amount of time so that processing can terminate gracefully. This extra time is normally 10% of the job's original allotted time, that is, the time specified on the $JOB card or with the SUBMIT command. (The 10% value is an installation option.) Next, the batch controller searches for %TERR. Further processing depends on the results of this search:

1. If %TERR is found, processing is continued from that point.
2. If %FIN is encountered before %TERR is reached, an error message is issued and processing is continued from %FIN.
3. If the end of the control file is reached before %TERR or %FIN is found, an error message is issued and the job is terminated.

See Chapter 6 for explanations of the BATCON error messages.

If the job exceeds the time limit again during the 10% extra time, no further time is allotted and the job is terminated.

2.5.3.4 %FIN - There are several uses for the %FIN label. When the batch controller detects an error in the batch commands in your control file, it issues an error message and transfers control to the next %FIN label it encounters. In addition, the batch controller skips to %FIN if a batch job is continued after a system crash and it was neither restartable nor checkpointed. (Refer to the description of the CHKPNT batch command in Section 2.6 for details.) Also, as described earlier, %FIN terminates an error packet and acts as a default error label when the batch controller does not find %CERR, %ERR, or %TERR.

With card jobs, SPRINT automatically inserts %FIN in the control file for its own cleanup purposes. Toward this end, SPRINT relies upon %FIN's role as default error label. As mentioned previously, %FIN satisfies the search for %CERR, %ERR, and %TERR. Thus, if you do not include the IF batch command or any of these reserved labels in the control file and an error occurs, the job is processed from the next %FIN encountered.

By placing %FIN at the end of a card input deck, SPRINT ensures that input files are deleted and that other cleanup activities are performed under both normal and error conditions. (A %FIN labeled line is executed whenever it is encountered, even when no error has occurred.) That is, if an error occurs and you have not provided for recovery, the job will not be terminated in an unknown state; rather BATCON transfers control to the %FIN that SPRINT inserted and cleans up the job before terminating it. Also, the system cleans up the job when no error has occurred.

You can also place %FIN near the end of the control file when you create your own at the terminal. Or you can scatter %FINs throughout the file for periodic cleanup. But be careful in your placement of the label because it could interfere with the batch controller's search for another reserved label as discussed above. Also, a label specified in the GOTO batch command cannot follow a %FIN. (That is, the batch controller abandons its search for user labels as well as for reserved labels when it encounters %FIN. See NOTE for the BACKTO, CHKPNT, and REQUEUE commands described in Section 2.6 for the exceptions to this rule.) Therefore, it may be best for you to put the %FIN near the end of the control file. However, if you desire periodic cleanup and have not inserted any of the other reserved labels and have not used the GOTO command, you need not worry in this regard.

2.6 BATCH COMMANDS

The batch commands, preceded by the system prompt character, are an extension of the timesharing command language. As such, they must be distinguishable from system commands (also preceded by the system prompt character) as well as from other batch commands. Thus, when you abbreviate a batch command you must use as many letters (minimum of two) as are necessary to differentiate it from these other commands.

The batch controller examines lines beginning with the system prompt character to determine whether the line is a batch command or a system command. It considers any command that is an ambiguous abbreviation, a single letter, or not one of the batch commands to be a system command and passes it to the system. If it is not a valid system command, the system issues an error message and the batch controller initiates error processing.

It is a good idea to spell out all commands to prevent possible problems with future releases of the system that introduce new batch or system commands. A new command could render one of your abbreviated commands ambiguous.

An important difference between the batch and system commands is that batch commands do not affect the running job; rather, they control the batch system environment in which the job runs. For example, you can use such batch commands as GOTO and BACKTO to alter the batch controller's sequence of operations.

The remainder of this section describes the batch commands in alphabetical order.

@ BACKTO

Function

The BACKTO command directs the batch controller to search the previous lines in the control file for a line beginning with the specified label.

Characteristics

The search starts at the first line of the control file and continues until the batch controller finds either the specified label or the BACKTO command that initiated the search. In the former case the batch controller continues execution at the label; in the latter, the batch controller issues an error message and terminates the job.

Command Format

```
@ BACKTO label
```

Where:

label	is a 1- to 6-character label as defined in Section 2.3.

Comment

Normally, the batch system reads the control file line by line and passes the commands and data to the system and your program. When you put a BACKTO command into the control file, you tell batch to interrupt the normal reading sequence and to search back in the control file to find a line containing the label specified in the BACKTO command. The BACKTO command searches for the label you specified, starting from the beginning of the file and ending at the place the command was given. When the labeled line is reached, batch executes the line and continues from that point.

If batch cannot find the labeled line, batch terminates your job.

NOTE

If a BACKTO command occurs after a %FIN label in the control file, the label specified in the BACKTO command may also be after the %FIN. (Recall from Section 2.5.3.4, %FIN, that %FIN satisfies the search for reserved labels and for labels specified in the GOTO command.)

Example

The following example demonstrates use of the BACKTO command. Note that it introduces the CHKPNT, PLEASE, IF, and GOTO batch commands and makes use of a reserved label.

```
                .
                .
                .
BLKC::.CHKPNT BLKC
.PLEASE BLKCON - ENTERING BLKC DIRECTORY ROUTINE ^[
.MOUNT BLKC:/REMARK:"BLACK PACK PROCEDURES"
!TRAP TO %CERR:: ON FAILURE
.RUN NEW:SPACE
.IF (ERROR) .R SPACE
*BLKC.SPC=BLKC:
.DIRECT @BLKC.CCL
!TRAP TO %CERR:: ON FAILURE
.DISMOUNT BLKC:
.IF (ERROR) !OK, MAYBE SOMEONE ELSE NEEDS THE PACK
.PLEASE BLKCON - BLKC DIRECTORY ROUTINE SUCCESSFUL ^[
                .
                .
                .

.GOTO BCEND
%CERR::
.PLEASE BLKCON - ERROR IN BLKC DIRECTORY ROUTINE <TRY AGAIN?>
.BACKTO BLKC
BCEND::

                .
                .
                .
```

If %CERR:: is reached, a 2-way PLEASE command message will be sent to the operator. At this point, the operator has the option of killing the job or allowing it to continue. If the job is continued, the BACKTO command will be executed and the procedure will be retried.

NOTE

Since the system has a BACKSPACE command, the only abbreviation of BACKTO that will be recognized is BACKT; that is, "BACK label" will be considered a BACKSPACE command and it will be passed to the system.

@ CHKPNT

Function

The CHKPNT command directs the batch controller to save a potential restart point to be used in the event of a system failure.

Characteristics

You can place as many CHKPNT commands as you desire in your control file. If you restart the job, processing begins at the **first occurrence** of the label specified in the most recently executed CHKPNT command. Before this, however, if a step header (lines between $STEP and $ENDHDR) exists, BATCON reexecutes the step header.

Command Format

```
@ CHKPNT label
```

Where:

label is a 1- to 6-character label as defined in Section 2.3.

Comment

The CHKPNT command and the /RESTART switch (specified with the $JOB card or the SUBMIT system command) are interrelated. Consider the actions that the batch controller would perform after a system crash for all combinations of this switch and command:

1. No CHKPNTs /RESTART:NO

   In this instance, after the system logs in the job, the system restarts the job even though no CHKPNT commands are specified and the argument to /RESTART is NO. But the batch controller skips everything in the control file up to the first %FIN label it finds and continues processing from there. On TOPS-20 systems, the search is for %RERR. If %RERR is not present the search is for %FIN.

2. No CHKPNTs /RESTART:YES

   The batch controller resumes processing at the first line of the control file.

3. CHKPNTs /RESTART:YES

   The batch controller resumes processing at the first occurrence of the label specified in the most recently executed CHKPNT command. If the batch controller did not execute a CHKPNT command before the system failed, it would resume processing at the first line of the control file. (This latter action is the same as in number 2 above.)

4. CHKPNTs /RESTART:NO

As in the previous case, the batch controller resumes processing at the first occurrence of the label specified in the most recently executed CHKPNT command. But, if the batch controller did not execute a CHKPNT command before the system failed, it would resume processing at the first %FIN label in the control file, not at the first line. On TOPS-20 systems, processing would resume at the first %RERR label. If %RERR is not present, the search is for %FIN.

Use this combination of the CHKPNT command and the /RESTART switch for a job that requires uninterrupted service up to a point, but that later can be safely interrupted and restarted. For example,

```
START::                     !!!Uninterruptible section!!!
          .
          .
          .
          .CHKPNT LABEL
LABEL::                     !Interruptible section
          .
          .
          .
%FIN                        !Transfer control here after
          .                 !a failure occurs within the
          .                 !uninterruptible section
          .
```

If an error occurs before the line, "CHKPNT LABEL", the system resumes processing at %FIN; otherwise, it continues from the line labeled LABEL.

NOTE

The label specified for a CHKPNT command may occur after a %FIN:: label. (Recall from Section 2.5.3.4, %FIN, that %FIN satisfies the search for most types of labels.)

. DUMP (TOPS-10)

Function

The DUMP command causes the system to display a variety of information relating to your batch job.

Characteristics

The batch controller invokes this command whenever a fatal error occurs and error recovery procedures are lacking. The batch controller then closes the control file. This automatic action is unrelated to your use of the command.

The DUMP command, unlike the IF command and the reserved labels, does not clear errors, even though it can be executed after an error occurs.

Command Format

```
. DUMP
```

NOTE

The DUMP batch command is different from the DUMP system program. To run the program, enter

```
.R DUMP
```

in the control file.

Example

The following example shows the use of the DUMP command in a TOPS-10 batch control file. Note that step processing is introduced here.

```
$STEP TEST
$MOUNT DSKB
$ENDHDR
.CHKPNT START
.PLEASE ** BATCH ** Starting Batch demo^[
.ERROR ?
.OPERATOR $
.PRINT SAMPLE.FIL
.DUMP
.SYSTAT .
;
; Error trapping
;
%ERR::
.PLEASE ** BATCH ** User program error^[
.GOTO EXIT
%CERR::
.PLEASE ** BATCH ** System program error^[
.GOTO EXIT
%TERR::
.PLEASE ** BATCH ** Time limit exceeded^[
EXIT::
;
; [End of DUMPX.CTL]
```

The following is the resulting log file:

```
                        21-Jul-88 16:29:25

BATCON Version  104(4570)                          GLXLIB Version  1(613)

             Job DUMPX Req #1619 for MORRILL,E [27,5342] in Stream 2

     OUTPUT:  Nolog                                TIME-LIMIT: 0:05:00
     UNIQUE:  Yes                                  BATCH-LOG:  Supersede
     RESTART: No                                   ASSISTANCE: Yes
                                                   SEQUENCE:   1291

     Input from => DSKC:DUMPX.CTL[27,5342]
     Output to  => DSKC:DUMPX.LOG[27,5342]


16:29:25 MONITR
16:29:25 MONITR .^C
16:29:25 MONITR
16:29:25 MONITR .LOGIN [27,5342] /ACCOUNT:""/BATINT:YES/BATNAM:"DUMPX"
/BATSEQ:1291/BATSTR:2/NAME:"MORRILL,E"/REQID:1619/DEFER/LOCATE:26
/SPOOL:ALL/TIME:300
16:29:27 USER   JOB 17     RZ062A KL #1026/1042 TTY476
16:29:32 USER   [LGNJSP Other jobs same PPN:69]
16:29:32 USER   16:29   21-Jul-88       Mon
16:29:33 MONITR
16:29:33 MONITR .
16:29:33 HEADER $STEP TEST
16:29:33 HEADER $MOUNT DSKB
16:29:33 BATCH  .MOUNT DSKB
16:29:45 USER   [Mount Request DSKB Queued, Request-ID #1621]
16:29:45 USER   [Structure DSKB Mounted]
16:29:45 MONITR
16:29:45 MONITR .
16:29:45 HEADER $ENDHDR
16:29:45 HEADER [3 lines processed in step TEST header]
16:29:45 BATCH  .CHKPNT START
16:29:45 BATCH  .PLEASE ** BATCH ** Starting Batch demo^[
16:29:45 BATCH  .ERROR ?
16:29:45 BATCH  .OPERATOR $
16:29:45 BATCH  .PRINT SAMPLE.FIL
16:29:48 USER
16:29:48 USER   %WLDLKE Non-existent file DSK:SAMPLE.FIL
16:29:48 USER
16:29:48 USER   ?QUENFI No files in request
16:29:48 USER
16:29:48 MONITR
16:29:48 MONITR .
16:29:48 BATCH  .DUMP
16:29:48 DUMP           -- Batch Stream and Job Data --
16:29:48 DUMP   Stream:
16:29:48 DUMP           Job in error
16:29:48 DUMP           Error: ?   Operator: $   Silenced: No
16:29:48 DUMP           Processing node: KL1026(26)
16:29:48 DUMP           Last step: TEST
16:29:48 DUMP           Last label: START
16:29:48 DUMP           Last CHKPNT: START
16:29:48 DUMP           Last line to job: .DUMP
16:29:48 DUMP           Last line from job: .
16:29:48 DUMP   Last line to OPR: *** BATCH ** Starting Batch demo^[
16:29:48 DUMP           Last line from OPR:
16:29:48 DUMP           Last Batch command: DUMP
```

```
16:29:48 DUMP   Job:
16:29:48 DUMP           Job: 17
16:29:48 DUMP           TTY476
16:29:48 DUMP           User: [27,5342]
16:29:48 DUMP           Program: QUEUE
16:29:48 DUMP           Located at: KL1026(26)
16:29:48 DUMP                   -- End of Dump --
16:29:48 IGNORE .SYSTAT .
16:29:48 IGNORE ;
16:29:48 IGNORE ; Error trapping
16:29:48 IGNORE ;
16:29:48 LABEL  %ERR::
16:29:48 BATCH  .PLEASE ** BATCH ** User program error^[
16:29:48 BATCH  .GOTO EXIT
16:29:48 LABEL  EXIT::
16:29:48 COMENT ;
16:29:48 COMENT ; [End of DUMPX.CTL]
16:29:48 BATCH  .KJOB/BATCH
16:29:49 USER
16:29:49 USER   [LGTOUL Other users logged-in under [27,5342], Jobs:69]
16:29:51 USER   Job 17  User MORRILL,E [27,5342]
16:29:51 USER   Logged-off TTY476  at 16:29:51  on 21-Jul-88
16:29:51 USER   Runtime: 0:00:00, KCS:9, Connect time: 0:00:25
16:29:51 USER   Disk Reads:183, Writes:7
```

In the example, the user issued the DUMP command to cover a possible PRINT error (which occurred). Note that all lines following the DUMP command were skipped until the batch controller encountered the %ERR label. This label cleared the error condition, allowing normal processing to continue.

Most of the dump lines are self-explanatory, indicating, for example, the characters specified for the ERROR and OPERATOR batch commands and the last lines sent to and from the job. However, the following dump lines may need clarification: the third line indicates the network node on which the job was processed; similarly, the last line indicates the network node to which the job's output was sent.

@ ERROR

Function

The ERROR command causes the batch controller to interpret a specified character as an error signal.

Characteristics

The batch controller examines each line of output that is sent from your job to the log file. If any of these lines begins with a question mark or with the character you specified with the ERROR command, the batch controller begins error processing. (Section 2.5, ERROR PROCESSING, discusses this topic.) **The question mark (?) is always recognized as an error indicator irrespective of your use of the ERROR command.** (See the NOERROR command in this section for related information.)

Command Format

```
@ ERROR character
```

Where:

character	is the beginning character of the line that is to be recognized as an error. If omitted, only the question mark (?) will be recognized as an error signal.

NOTE

You must not specify a control character, an exclamation point (!) or a semicolon (;). The exclamation point and semicolon will be interpreted as the comment signal character and not as the error signal character. The control character is limited to its specific function.

Restrictions

You can specify only one additional (besides the question mark) character as an error indicator.

Comment

You could use the ERROR command to flag warnings, indicated by lines of system program output beginning with the percent character (%), in addition to fatal errors, which are denoted by the question mark (?). For example:

```
.ERROR %                   !FLAG WARNINGS
.DIR FILB.CTL              !SEE IF FILB.CTL EXISTS
.IF (ERROR) .GOTO A        !SKIP THE FOLLOWING IF FILE MISSING
.
.
.
A:: .ERROR                 !NOW FLAG ONLY FATAL ERRORS
.
.
.
```

Here, a fatal error occurs if FILB.CTL is missing when a directory of it is requested. Ordinarily, the system merely issues a warning message under these circumstances, but in the example above, the first line (.ERROR %) directs the system to treat warnings as fatal errors.

You could also use the ERROR command to flag any condition that the system ignores. In this case, you would specify a character of your choice with the ERROR command, then place that character at the beginning of an output line in a program. You would issue this line from the program when the condition occurs.

@ GOTO

Function

The GOTO command directs the batch controller to search the control file in a forward direction for a specified label.

Characteristics

When the label is found, control is transferred to the statement associated with the label. If the batch controller encounters a %FIN label before it reaches the label specified in the GOTO command, it will transfer control to this %FIN label. If neither the label nor a %FIN is found before the end of the control file is reached, the batch controller will issue an error message (refer to Chapter 6 for the batch controller messages) and the job will be terminated.

Command Format

```
@ GOTO label
```

Where:

label is a 1- to 6-character label as defined in Section 2.3.

Restrictions

The search initiated by the GOTO command cannot bypass a %FIN label.

Comment

You can use the GOTO command as the statement in an IF command to aid you in error processing. For example:

```
        .
        .
        .
IF (ERROR) GOTO ABC
        .
        .
        .
ABC::TYPE MYPROG
```

When batch encounters a GOTO command in the control file, it searches forward in the control file to find the label specified in the GOTO command. Batch then resumes processing of the control file at the line that has the specified label. If the label is not found, batch issues the message

```
? BTNCNF Could not find label xxxxxx   (TOPS-20)

hh:mm:ss BATECF ?End of control file while searching
for label xxxxxx  (TOPS-10)
```

and the job is terminated.

If you do not include a GOTO command in the control file, batch reads the control file sequentially from the first statement to the last.

@ IF

Function

The IF command directs the batch controller to test for the condition specified in the command string.

Characteristics

If the condition specified in the IF command string is true, the statement is executed. Otherwise, this command and its statement are treated as comments and the batch controller proceeds to the next line in the control file.

Command Format

```
@ IF (condition) statement
```

Where:

(condition)	is (ERROR)	- Directs the batch controller to execute the statement if an error has occurred.
	or	
	(NOERROR)	- Directs the batch controller to execute the statement if no error has occurred.

The parentheses must be included.

statement	can be a batch or system command, user or system program data, or a comment. This parameter is optional.

Comment

The batch controller recognizes the existence of an error when it encounters an output line beginning with a question mark or with the character you specified in the ERROR command. As described in Section 2.5, ERROR PROCESSING, when an error occurs, the IF command must be the next system-level command in the control file.

If you omit the statement parameter while specifying (ERROR) as the condition parameter and an error occurs, the batch controller clears the error and continues to process the control file. In this situation the batch controller acts in the same manner as it does when you use the NOERROR batch command. That is, it overlooks the error, proceeding with normal operations.

NOTE

You cannot use the IF command to intercept a time-limit error (?TIME LIMIT EXCEEDED message). (Refer to Section 2.5.3.3, %TERR, for information on time-limit errors.)

Related Items

Reserved labels

The reserved labels provide for the execution of multiple control-file lines (called error packets) in the event of an error. The batch controller bypasses these lines under normal circumstances. (Refer to Section 2.5.3.2, %CERR and %ERR, for a description of error packets.)

If an IF command does not follow the line that produced the error, the batch controller searches for the appropriate reserved label and transfers control to it. (See Section 2.5.3, Reserved Labels, for more information.)

Examples

1. Include the IF (ERROR) command in your control file at a place where you suspect an error may occur. Note that the IF (ERROR) command must be the next command in your control file (that is, the next line that begins with a system prompt after an error occurs); otherwise, batch terminates your job.

```
!DO A DIRECTORY IF AN ERROR OCCURS
@IF (ERROR) @VDIRECTORY
     .
     .
     .
```

2. Use the IF (NOERROR) command to direct batch or the system to perform tasks for you when an error does not occur at the point in your control file where you place the IF (NOERROR) command.

```
!IF NO ERROR OCCURS, GIVE A SECOND LINE OF INPUT
@IF (NO ERROR) *FILE.SCM=A.TXT,B.TXT
```

Refer to Section 2.5 for more examples of using IF (NOERROR) and IF (ERROR).

If an error occurs and batch does not find an IF command as the next command line in the control file, batch terminates the job.

@ MESSAGE

Function

The MESSAGE command has the same function as the PLEASE command.

Comment

Direct use of the MESSAGE command is strongly discouraged (because of possible compatibility problems with future versions of the batch system) and for this reason, the syntax for the command is not specified here. Use the PLEASE batch command instead.

.
@ NOERROR

Function

The NOERROR command instructs the batch controller to ignore all error messages except those that indicate a time-limit error (?TIME LIMIT EXCEEDED message) and those issued by the batch controller itself. (Refer to Chapter 6 for batch controller messages.)

Command Format

```
    .
    @ NOERROR
```

Comment

When batch reads the NOERROR command, it ignores any error messages that would normally cause it to terminate your job. The only exception is the message ?TIME LIMIT EXCEEDED. Batch always recognizes this as an error message, gives you an extra 10% of your allotted time, and terminates your job.

You can use NOERROR commands in conjunction with ERROR commands in the control file to control error reporting. For example, if you wish to ignore errors at the beginning and end but not in the middle of the control file, place ERROR and NOERROR commands at the appropriate places in the control file. In addition, you can also specify which messages must be treated as fatal errors.

```
          .
          .
          .
    NOERROR
          .
          .
          .
    ERROR %
          .
          .
          .
    ERROR
          .
          .
          .
    NOERROR
```

The first command tells batch to ignore all errors in your job. The second command tells batch to recognize as errors any message that starts with a question mark (?) or a percent sign (%). You change the error reporting with the next command to tell batch to go back to recognizing only messages that begin with a question mark as fatal. The second NOERROR command tells batch to ignore all error messages again. If the ?TIME LIMIT EXCEEDED message is issued at any time, batch will print the message, extend the time by 10%, and then terminate the job.

@ NOOPERATOR

Function

The NOOPERATOR command directs the batch controller to terminate dialogue mode between your job and the system operator. (Refer to Section 2.7, JOB-OPERATOR COMMUNICATION, for information on dialogue mode.)

Command Format

```
@ NOOPERATOR
```

@ OPERATOR

Function

The OPERATOR command directs the batch controller to enable dialogue mode between your job and the system operator. (Refer to Section 2.7, JOB-OPERATOR COMMUNICATION, for information on dialogue mode.)

Command Format

```
@ OPERATOR character
```

Where:

character	is the character that begins the message to be sent to the operator (such as #). Specification of the character is optional and if it is omitted, the dollar sign ($) will be assumed to be the operator signal character.

NOTE

You must not specify a control character, a semicolon (;), or an exclamation point (!). The exclamation point and semicolon will be interpreted as comment signal characters, not as communication signal characters. A control character is limited to its specific function.

Related Command

```
@ PLEASE
```

Refer to the Related Command section of the PLEASE command description.

@ PLEASE

Function

The PLEASE command directs the batch controller to type a specified message to the system operator.

Command Format

```
    @ PLEASE message^[<RET>

                        or

    @ PLEASE message<RET>
```

Where:

message	is the message to be typed to the operator.
^[	generates the ESCape character. If this character is present, processing continues normally after the message has been sent to the operator. If the character is omitted, the job will wait for a response from the operator before resuming its normal processing.
<RET>	means the carriage-return/line-feed is required.

Example

Refer to the example given with the BACKTO or REQUEUE batch command.

Related Command

@ OPERATOR

The OPERATOR command allows you to send multiple lines of output to the operator, not only from a running system or user program but also from a data file; whereas the PLEASE command permits only a one-line message to be sent from the control file to the operator. Both commands allow for response from the operator; however, the operator response to PLEASE simply goes to the log file. With OPERATOR in effect, the response goes to the requesting system program, user program, or to the monitor.

The OPERATOR command does not affect operation of the PLEASE command. The two commands are completely independent.

Refer to Section 2.7 for details on the OPERATOR command and for a more complete comparison of these two commands.

@ REQUEUE

Function

The REQUEUE command indicates to the batch controller that the job is to be requeued for processing at a later time.

Characteristics

When the job is requeued, the previous step header (if one exists) is reexecuted before processing continues at the specified label. If you omit the label, the job will be requeued at the label specified in the last CHKPNT command executed before the REQUEUE. If you omit the label and the batch controller did not execute a CHKPNT command, the job will be restarted at the beginning of the control file.

After the batch controller requeues the job, QUASAR waits five minutes before scheduling the job to run (making it available to the batch controller). The job then must compete with other jobs in the batch input queue.

NOTE

The label specified for a REQUEUE may be after a %FIN label.

Command Format

```
@ REQUEUE label
```

Where:

label is a 1- to 6-character label as defined in Section 2.3.

Example

Assume you want to save a disk file on tape and need an error recovery routine to cover possible tape mount failures. You could write your control file as follows, using the @REQUEUE command:

```
    .
    .
    .
LABEL::
@CHKPNT LABEL         !Checkpoint in case system crashes while
                      !waiting for mount
!Ask for a tape mount to save a disk data file created at
!beginning of job:
@MOUNT TAPE ABC: /VOLID:DEF/LABEL-TYPE:ANSI
@IF (NOERROR) @GOTO DUMP
!Job proceeds from this point if mount failed
@PLEASE hold job until a tape drive is available ^[
!No operator response required
!Data file is still on disk; restart job at tape mount:
@REQUEUE LABEL
DUMP::
@R DUMPER             !Save file
    .
    .
    .
```

Here, the REQUEUE command causes the batch controller to defer processing a section of the control file until the error condition is eliminated.

.@ REVIVE

Function

The REVIVE command directs the batch controller to resume normal listing in the log file (that is, all output from the job is placed in the log file).

Command Format

```
.
@ REVIVE
```

Related Command

```
.
@ SILENCE
```

In its initial state, the batch controller sends all output to the log file; thus, the REVIVE command reinstates this initial condition, clearing the effect of the SILENCE command.

$\overset{.}{@}$ SILENCE

Function

The SILENCE command directs the batch controller to suppress all output normally sent to the log file. After a SILENCE command has been issued, the batch controller will still write all error messages and batch commands to the log file.

Command Format

```
.
@ SILENCE
```

Related Command

```
.
@ REVIVE
```

This suppression of output is cleared by the REVIVE command.

2.7 JOB-OPERATOR COMMUNICATION

The batch controller provides for communication between your job and the operator. Three methods of communication are available.

With the first method it is possible for the batch controller to enter dialogue mode, allowing your job to communicate in a two-way fashion with the operator. The second method provides a means for your job to communicate only in a one-way fashion with the operator. The third method provides for two-way communication; however, this method differs from the first one in that the operator's response goes to the log file rather than to your job as data to be processed.

These three communication methods, referred to respectively as "dialogue mode", the "double quote" facility, and using the PLEASE batch command, are described in detail below in Sections 2.7.1, 2.7.2, and 2.7.3.

2.7.1 Dialogue Mode

Under timesharing, it may be common for your jobs to request data that you supply at your terminal. System and user programs as well as the monitor can request input. (When your job is at monitor command level, the monitor is requesting input.)

Such requests for input can be satisfied while your job is running in batch mode; however, while this mode is in effect, the data comes from either the control file or the system operator.

Data that your job requests and receives from you at your terminal during timesharing ordinarily comes from the control file during batch processing. Dialogue mode extends your job's ability to communicate during batch processing by allowing the operator to provide data upon the job's request.

Two commands are associated with dialogue mode: OPERATOR, used to open dialogue mode and NOOPERATOR, used to terminate dialogue mode.

When dialogue mode is active, it is possible for your job to "converse" with the operator. "Conversation" takes place in this manner: the batch controller sends specified lines of output from user and system programs and from data files to the operator as well as to the log file. The operator's response to the message satisfies the job's current request (from any running program or from the monitor) for data.

Input entered by the operator is interpreted at either monitor or user level depending upon the level at which input is requested. For example, if the monitor is waiting for data, the input is interpreted at monitor level; if a program is waiting for data, the operator's response is interpreted at user level. Note that even batch commands such as GOTO would be interpreted at one of these two levels; thus, the operator cannot effectively enter a batch command to alter the sequence of your control file.

The following section, The OPERATOR Command, further describes dialogue mode.

2.7.1.1 The OPERATOR Command - As stated above, the OPERATOR command activates dialogue mode. (Refer to Section 2.6 for a description of the command format.)

While dialogue mode is in effect, the batch controller checks to see if any output line from your job begins with the character specified in the OPERATOR command. When the batch controller encounters such a line it sends that line along with all subsequent output lines, regardless of their initial character, to the operator's terminal and to the log file.

After copying text in the above manner, the batch controller suspends the job until the operator responds to the sent text. That is, the batch controller does not read the next line in the control file as it would if dialogue mode had not been activated.

When the operator responds, the batch controller resumes its processing of the job, sending the response to the requesting software. It sends the response to the log file also, where it is identified as an operator response. (That is, "OPERAT" precedes the response. See Section 6.1, THE LOG FILE, for details.) Output to the operator is discontinued until the job issues another line beginning with the character specified in the OPERATOR command. The batch controller then repeats the process described above.

Input provided by the operator is limited to one line per message. If you anticipate more than one line for the operator's response, you should issue another message beginning with the specified dialogue character.

NOTE

Several system software components use the foregoing method to communicate with the operator, particularly when reading or writing a multivolume tape file. Among these components are the BACKUP (-10), DUMPER (-20), and SORT utilities, and part of the run-time system for COBOL. For example, the COBOL run-time system issues the message

```
$MOUNT SCRATCH TAPE ON LOGICAL DEVICE TAP1:;
     PHYSICAL DEVICE MTA3:
$TYPE CONTINUE TO PROCEED
```

when writing to multiple tape reels.

Activate dialogue mode through the OPERATOR command whenever you include one of these components in a batch control file and you:

1. Specify a multivolume tape file to be read or written.
2. Suspect that a file to be created and written to tape will require more than one reel.

Also, for compatibility with this system software, define the dialogue prompt character as the dollar sign ($).

Example

The following example shows the use of dialogue mode with the OPERATOR batch command:

```
@OPERATOR
!TEST OF @OPERATOR USING FILE DEMO3.DAT
@TY DEMO3.DAT
!
!       NOW WE WILL SET UP A PIP COMMAND TO
!       ACCEPT OPERATOR COMMANDS
!
@R PIP
*TTY:=DEMO4.DAT

@NOOPER
!
!END OF OPERATOR MODE
!
@TY DEMO5.DAT
!THE ABOVE MESSAGE SHOULD NOT GO TO OPERATOR
```

The control file above produces the following log file. (Refer to Chapter 6, BATCH SYSTEM OUTPUT, for an explanation of the log file.) The log file shows how a line from DEMO3.DAT beginning with a dollar sign causes subsequent lines from the file to type out at the operator's terminal. The system then waits for the operator's response. Next, a similar technique of sending lines from a data file to the operator is used, and the operator responds accordingly. After the system executes the NOOPERATOR batch command, the line from file DEMO5.DAT, beginning with a dollar sign, is treated as an ordinary line sent to the terminal; it is sent to the log file only.

```
                         8-Nov-88  8:35:06

BATCON Version  104(4127)              GLXLIB Version  1(525)

          Job OPER Req #101 for MORRILL in Stream 0

   OUTPUT:  Log                                  TIME-LIMIT: 0:05:00
   UNIQUE:  Yes                                  BATCH-LOG:  Append
   RESTART: No                                   ASSISTANCE: Yes
                                                 SEQUENCE:   1690

   Input from => MISC:<MORRILL>OPER.TST.2
   Output to  => MISC:<MORRILL>OPER.LOG



8:35:07 MONTR    2102 Development System, TOPS-20 Monitor 4(3212)
8:35:07 MONTR   @SET TIME-LIMIT 300
8:35:07 MONTR   @LOGIN MORRILL 341
8:35:07 MONTR   @ Job 27 on TTY217 8-Nov-88 08:35:10
8:35:10 MONTR   @
8:35:10 MONTR   [MISC Mounted]
8:35:10 MONTR
8:35:10 MONTR   [CONNECTED TO MISC:<MORRILL>]
8:35:10 BATCH   @OPERATOR
        !TEST OF @OPERATOR USING FILE DEMO3.DAT
8:35:10 MONTR   TY DEMO3.DAT
8:35:11 MONTR   $Now we are in OPERATOR mode where you can
8:35:11 MONTR   type any command you wish and have it processed
```

```
8:35:11 MONTR   by the job.  So try:
8:35:11 MONTR
8:35:11 MONTR         SYS     SYS
8:35:11 MONTR
8:35:11 MONTR   This will provide a SYSTAT summary for the job.
8:35:32 BAOPR   From Operator: SYS SYS
8:35:32 MONTR   @ Thu 8-Nov-88 08:35:32  Up 15:21:28
8:35:32 MONTR    16+10 Jobs   Load av (class 1)   0.06   0.05   0.16
8:35:32 MONTR
8:35:32 MONTR
                !
                !      NOW WE WILL SET UP A PIP COMMAND TO
                !      ACCEPT OPERATOR COMMANDS
                !
8:35:32 MONTR   @R PIP
8:35:32 MONTR   @**TTY:=DEMO4.DAT
8:35:34 USER    $Please type a valid PIP command or ^Z to exit.
8:35:34 USER    For example, you could type:
8:35:34 USER
8:35:34 USER          TTY:=DEMO4.DAT
8:35:34 USER
8:35:34 USER    This command will produce this text message.
8:35:34 USER    *
8:36:03 BAOPR   From Operator: ^Z
8:36:03 USER    ^Z
8:36:03 USER    **
8:36:03 BATCH   @NOOPER
                !
                !END OF OPERATOR MODE
                !
8:36:03 MONTR   *^C
8:36:03 MONTR   @TY DEMO5.DAT
8:36:03 MONTR   @$This line should appear only in the log file, not at
8:36:03 MONTR   the operator's terminal.
                !THE ABOVE MESSAGE SHOULD NOT GO TO OPERATOR
8:36:03 MONTR
8:36:04 MONTR   @Killed Job 27, User MORRILL, Account 341, TTY 217,
8:36:04 MONTR     at  8-Nov-88 08:36:04,  Used 0:00:00 in 0:00:54
```

Communication with the operator takes place in the following manner, as the operator's log file shows:

```
8:35:06         Batch-Stream 0  --Begin--
                Job OPER Req #101 for MORRILL

8:35:11  <14>   Batch-Stream 0 JOB #27   --Message from Batch User--
                Job OPER Req #101 for MORRILL
                $Now we are in OPERATOR mode where you can
                type any command you wish and have it processed
                by the job.  So try:

                      SYS     SYS

                This will provide a SYSTAT summary for the job.
```

```
OPR>RESP 14 SYS SYS
OPR>
8:35:34  <15>    Batch-Stream 0 JOB #27   --Message from Batch User--
                 Job OPER Req #101 for MORRILL
                 $Please type a valid PIP command or ^Z to exit.
                 For example, you could type:

                         TTY:=DEMO4.DAT

                 This command will produce this text message.
                 *

OPR>RESP 15 ^Z
OPR>
8:36:05          Batch-Stream 0  --End--
                 Job OPER Req #101 for MORRILL
```

2.7.1.2 The NOOPERATOR Command - This command terminates dialogue mode; that is, the batch controller no longer acknowledges in any special way the character specified with the OPERATOR command. (Refer to Section 2.6 for a description of the command format.) The NOOPERATOR command is in effect at the time the batch job is initialized.

2.7.2 Using the Double Quote (")

It is possible to send a one-line comment from a running program or data file to the operator without using the dialogue mode described above. Any line of output whose first character is a double quote (") is printed on the operator's console as a comment and has no other effects. The line also is placed in the log file as normal user output.

Remember that this facility provides for one-line messages only; thus, if subsequent comment lines are to be typed at the operator's console, they too must begin with a double quote.

Using the double quote facility does not affect dialogue mode. However, if you specify a double quote with the OPERATOR command (which is permissible), the double quote is treated as a portion of the message sent to the operator.

Example

The following demonstrates use of the double quote communication facility. The control file consisting of the line, @TYPE QUOTE, produces the following log file:

```
                          8-Nov-88  8:33:53

BATCON Version  104(4127)             GLXLIB Version  1(525)

            Job TEST Req #99 for MORRILL in Stream 0

        OUTPUT:  Log                           TIME-LIMIT: 0:05:00
        UNIQUE:  Yes                           BATCH-LOG:  Append
        RESTART: No                            ASSISTANCE: Yes
                                               SEQUENCE:   1688

        Input from => MISC:<MORRILL>TEST.CTL.11
        Output to  => MISC:<MORRILL>TEST.LOG


 8:33:53 MONTR   2102 Development System, TOPS-20 Monitor 4(3212)
 8:33:53 MONTR  @SET TIME-LIMIT 300
 8:33:53 MONTR  @LOGIN MORRILL 341
 8:33:53 MONTR  @ Job 18 on TTY217 8-Nov-88 08:33:56
 8:33:56 MONTR  @
 8:33:56 MONTR  [MISC Mounted]
 8:33:56 MONTR
 8:33:56 MONTR  [CONNECTED TO MISC:<MORRILL>]
 8:33:56 MONTR  @TYPE QUOTE
 8:33:57 MONTR  "Send this line to the operator.
 8:33:57 MONTR  "These lines test the 'double quote' facility.
 8:33:57 MONTR  @
 8:33:59 MONTR  Killed Job 18, User MORRILL, Account 341, TTY 217,
 8:33:59 MONTR    at  8-Nov-88 08:33:59,  Used 0:00:00 in 0:00:02
```

and causes the messages below to appear on the operator's terminal.

```
 8:33:53        Batch-Stream 0  --Begin--
                Job TEST Req #99 for MORRILL

 8:33:57        Batch-Stream 0 JOB #18   --Message from Batch User--
                Job TEST Req #99 for MORRILL
                "Send this line to the operator.


 8:33:58        Batch-Stream 0 JOB #18   --Message from Batch User--
                Job TEST Req #99 for MORRILL
                "These lines test the 'double quote' facility.


 8:33:59        Batch-Stream 0  --End--
                Job TEST Req #99 for MORRILL
```

2.7.3 Using the PLEASE Command

The PLEASE command, like the communication methods described above, allows you to send messages to the system operator (Refer to Section 2.6, BATCH COMMANDS, for a description of the command function and format). Although this command shares several characteristics with these methods, it differs from each in fundamental areas. The PLEASE command is different from both methods in that a message can go to the operator's terminal only from the control file, not from a running program or from a data file.

Like dialogue mode, the PLEASE command permits two-way communication between the job and the system operator; however the communication is dead-end. With the PLEASE command, the operator's response has no bearing upon the way in which the job is processed. Rather, the response serves mere informational purposes, going only to the log file.

You can also use the PLEASE command for one-way communication from your job to the operator. Like the double quote (") facility, which too is for one-way communication, PLEASE permits a one-line only message to be sent to the operator. But, as mentioned above, with the PLEASE command, the line comes from the control file rather than from a running program.

Example

The following respectively demonstrates one-way and two-way job-operator communication using the PLEASE command. The control file consisting of the lines,

```
@PLEASE send this line to the operator's terminal ^[
@PLEASE send this line to the operator's terminal
```

produces the following log file:

```
                        8-Nov-88  8:33:04

BATCON Version  104(4127)             GLXLIB Version  1(525)

              Job PLEASE Req #97 for MORRILL in Stream 0

      OUTPUT:  Log                               TIME-LIMIT: 0:05:00
      UNIQUE:  Yes                               BATCH-LOG:  Append
      RESTART: No                                ASSISTANCE: Yes
                                                 SEQUENCE:   1686

      Input from => MISC:<MORRILL>PLEASE.CTL.6
      Output to  => MISC:<MORRILL>PLEASE.LOG


 8:33:05 MONTR   2102 Development System, TOPS-20 Monitor 4(3212)
 8:33:05 MONTR  @SET TIME-LIMIT 300
 8:33:05 MONTR  @LOGIN MORRILL 341
 8:33:05 MONTR  @ Job 18 on TTY217 8-Nov-88 08:33:08
 8:33:08 MONTR  @
 8:33:08 MONTR  [MISC Mounted]
 8:33:08 MONTR
 8:33:08 MONTR  [CONNECTED TO MISC:<MORRILL>]
 8:33:08 BATCH  @PLEASE send this line to the operator's terminal^[
 8:33:08 BATCH  @PLEASE send this line to the operator's terminal
 8:33:35 BAOPR  From Operator: OK
 8:33:35 MONTR
 8:33:36 MONTR  Killed Job 18, User MORRILL, Account 341, TTY 217,
 8:33:36 MONTR    at  8-Nov-88 08:33:36,  Used 0:00:00 in 0:00:27
```

Communication with the operator takes place in the following manner, as the operator's log file shows:

```
 8:33:04         Batch-Stream 0  --Begin--
                 Job PLEASE Req #97 for MORRILL

 8:33:09         Batch-Stream 0 JOB #18   --Message from Batch User--
                 Job PLEASE Req #97 for MORRILL
                 PLEASE send this line to the operator's terminal$

 8:33:09  <13>   Batch-Stream 0 JOB #18   --Message from Batch User--
                 Job PLEASE Req #97 for MORRILL
                 PLEASE send this line to the operator's terminal

OPR>RESP 13 OK
OPR>
 8:33:36         Batch-Stream 0  --End--
                 Job PLEASE Req #97 for MORRILL
```

2.8 MOUNTING DISKS AND TAPES (TOPS-10)

With a limited number of disk and tape drives at a computer site, and with many users competing for these mountable resources, conflicts and delays can arise during the processing of jobs. Consider the following:

1. One job has resource A and needs resource B, while another job has resource B and needs resource A.
2. A job mounts a resource but does not use it, causing other jobs to wait for the resource indefinitely.

With batch jobs, these conditions are particularly troublesome because they tie up batch streams.

To alleviate these problems, place the ALLOCATE and MOUNT system commands (described in your operating system commands manual) at the beginning of the control file in what is referred to as a step header. This restartable section of the control file begins with the $STEP command and ends with the $ENDHDR command, as shown in the following example:

```
$STEP SAMPLE        ;THE STEP NAME IS SAMPLE
$ALLOCATE BLKX
$MOUNT BLKY
$ENDHDR
.SHOW ALLOCATION
.REQUEUE CONTIN::
CONTIN::
```

In a step header, you precede the ALLOCATE and MOUNT commands with a dollar sign ($) rather than the system prompt.

A step is the set of control file lines from one step header to the next. As of the latest version of GALAXY, only one step is allowed in the control file. Thus, the whole control file represents a step. Note that the step name is specified with the $STEP command. After a job is submitted, the system prescans the control file before scheduling the job to run. During this preliminary examination, the file is checked for ALLOCATE and MOUNT commands contained in the step header. If the ALLOCATE command is present, the operator is notified of your job's device needs; if the MOUNT command is present, the devices are actually assigned to your job. Then the job is scheduled. If, during the prescan, the batch job is unable to secure the devices it requested in the MOUNT command, the operator cancels the request; and the job is terminated with a fatal error message. This activity is recorded in the log file.

Thus, when you use step header commands, the job does not start before obtaining ownership of the devices it needs.

NOTE

> Step header processing is optional. You are not required to specify step header commands every time you use mountable devices. But if you do not, your job may encounter the problems discussed above.

The control file above produces the following log files:

```
                        1-Dec-88 17:14:53

BATCON Version  104(4656)               GLXLIB Version  1(1067)

            Job LOCK Req #283 for MORRILL,E [27,5342] in stream 1

     OUTPUT:  Log                              TIME-LIMIT: 0:05:00
     UNIQUE:  Yes                              BATCH-LOG:  Append
     RESTART: No                               ASSISTANCE: Yes
     CORE:    512 pages                        SEQUENCE:   616


     Control file: DSKC:LOCK.CTL[27,5342,BATCH]
     Log file:     DSKC:LOCK.LOG[27,5342,BATCH]

17:14:53 MONITR
17:14:53 MONITR .^C
17:14:55 MONITR
17:14:55 MONITR .LOGIN [27,5342,BATCH]/ACCOUNT:""/BATINT:YES
/BATNAM:"LOCK"/BATSEQ:616/BATSTR:1/NAME:"MORRILL,E"/REQID:283/CORE:512P
/DEFER/LOCATE:26/SPOOL:ALL/TIME:300
17:14:55 USER   JOB 3     RZ117A KL #1026/1042 TTY367
17:15:00 USER   [LGNJSP Other jobs same PPN:50]
17:15:00 USER   17:14    1-Dec-88       Tue
17:15:00 MONITR
17:15:00 MONITR .
17:15:00 HEADER $STEP SAMPLE    ;THE STEP NAME IS SAMPLE
17:15:00 HEADER $ALLOCATE BLKX
17:15:00 BATCH  .ALLOCATE BLKX
17:15:01 USER   [Allocate request BLKX queued, request #285]
17:15:01 MONITR
17:15:01 MONITR .
17:15:01 HEADER $MOUNT BLKY
17:15:01 BATCH  .MOUNT BLKY
17:15:03 USER   [Mount request BLKY queued, request #288]
17:15:03 USER   % No UFD created
17:15:03 USER   [Structure BLKY mounted]
```

```
17:15:03 MONITR
17:15:03 MONITR .
17:15:03 HEADER $ENDHDR
17:15:03 HEADER [4 lines processed in step SAMPLE header]
17:15:03 BATCH  .SHOW ALLOCATION
17:15:04 USER
17:15:04 USER   Allocation for job 3 MORRILL,E [27,5342]
17:15:04 USER        Volume set          Resource     Type        All  Own
17:15:04 USER   --------------------    ----------  ----------   ---  ---
17:15:04 USER   ---                     RP04        Disk unit     2    1
17:15:04 USER   ---                     RP20        Disk unit     1    1
17:15:04 USER   BLKY                    BLKY        Structure     1    1
17:15:04 USER   BLKX                    BLKX        Structure     1    0
17:15:04 USER   DSKC                    DSKC        Structure     1    1
17:15:04 USER   DSKB                    DSKB        Structure     1    1
17:15:05 MONITR
17:15:05 MONITR .
17:15:05 BATCH  .REQUEUE CONTIN::
17:15:05 BATJRQ [Job requeued by user]
17:15:05 BATCH  .KJOB/BATCH
17:15:05 MONITR
17:15:06 USER
17:15:06 USER   [LGTOUL Other users logged-in under [27,5342], Jobs: 50]
17:15:09 USER   Job 3  User MORRILL,E [27,5342]
17:15:09 USER   Logged-off TTY367  at 17:15:09  on  1-Dec-88
17:15:09 USER   Runtime: 0:00:01, KCS:14, Connect time: 0:00:13
17:15:09 USER   Disk Reads:136, Writes:11

                        2-Dec-88  8:22:02

BATCON Version  104(4656)                   GLXLIB Version  1(1067)

           Job LOCK Req #12 for MORRILL,E [27,5342] in stream 0
       OUTPUT:  Log                         TIME-LIMIT: 0:05:00
       UNIQUE:  Yes                         BATCH-LOG:  Append
       RESTART: Yes                         ASSISTANCE: Yes
       CORE:    512 pages                   SEQUENCE:   616


       Control file: DSKC:LOCK.CTL[27,5342,BATCH]
       Log file:     DSKC:LOCK.LOG[27,5342,BATCH]

 8:22:02 MONITR ^C
 8:22:02 MONITR
 8:22:02 MONITR .LOGIN [27,5342,BATCH]/ACCOUNT:""/BATINT:YES
/BATNAM:"LOCK"/BATSEQ:616/BATSTR:0/NAME:"MORRILL,E"/REQID:12
/CORE:512P/DEFER/LOCATE:26/SPOOL:ALL/TIME:300
 8:22:02 USER   JOB 23     RZ120A KL #1026/1042 TTY370
 8:22:04 USER   08:22    2-Dec-88       Wed
 8:22:04 MONITR
 8:22:04 MONITR .
 8:22:04 HEADER $STEP SAMPLE    ;THE STEP NAME IS SAMPLE
 8:22:04 HEADER $ALLOCATE BLKX
 8:22:04 BATCH  .ALLOCATE BLKX
 8:22:04 USER   [Allocate request BLKX queued, request #105]
 8:22:04 MONITR
 8:22:04 MONITR .
 8:22:04 HEADER $MOUNT BLKY
 8:22:04 BATCH  .MOUNT BLKY
 8:22:05 USER   [Mount request BLKY queued, request #108]
 8:22:05 USER   % No UFD created
 8:22:05 USER   [Structure BLKY mounted]
 8:22:05 MONITR
 8:22:05 MONITR
```

```
8:22:05 HEADER $ENDHDR
8:22:05 HEADER [4 lines processed in step SAMPLE header]
8:22:05 BATBLA [Beginning processing at label CONTIN]
8:22:05 LABEL  CONTIN::
8:22:05 BATCH  .KJOB/BATCH
8:22:05 MONITR
8:22:07 USER   Job 23  User MORRILL,E [27,5342]
8:22:07 USER   Logged-off TTY370  at  8:22:08  on  2-Dec-88
8:22:07 USER   Runtime: 0:00:00, KCS:10, Connect time: 0:00:05
8:22:07 USER   Disk Reads:128, Writes:13, Blocks saved:3875
```

2.9 NORMAL PROCESSING OF THE BATCH CONTROL FILE

Figure 2-1 shows the paths taken with normal batch processing.

```
          _________
         | START   |
         |_________|
       ______|______
      |A NEW LINE   |
      |IN THE CTL   |
      | FILE IS     |
      | EXAMINED    |
      |_____________|
       ______|______
      | END OF      |  YES       _______
      | CTL FILE    |--------->| EOJ   |
      |     ?       |          |_______|
      |_____________|
             | NO
       ______|____                 ___
      | IS IT A   |               | A |
      | LABELED   |   NO          |___|
      | LINE ?    |------------->|
      |___________|              |
             | YES               |
        _____|____               V
      | %-TYPE   |    NO      | EXECUTE    |               ________
      |  LABEL   |---------->| THE LINE   |------------->| START  |
      |   ?      |            |____________|              |________|
      |__________|                  ^
             | YES                  |
        _____|____                  |
      | %FIN::   |    YES           |
      |   ?      |------------------
      |__________|
             | NO
    _________|_______
   | SEARCH FOR      |
   | NEXT LABEL      |
--->| IN CTL FILE     |
|  |   OR EOF        |
|  |_________________|
|     _______|_____
|    | END OF      |  YES          | ERROR      |             ______
|    | CTL FILE    |------------->| MESSAGE    |----------->| EOJ  |
|    |    ?        |               | ISSUED     |            |______|
|    |_____________|               |____________|
|           | NO
|      _____|____
|    | %FIN::   |    YES            ___
|    |   ?      |------------->| A |
|    |__________|              |___|
|____________| NO
```

Figure 2-1: Normal Processing

CHAPTER 3

JOB CONTROL

3.1 INTRODUCTION

Job control refers to your ability to manipulate a job as an entire unit. Manipulation of a job includes such actions as sending a job to or removing it from a queue, modifying certain job parameters after a job has entered a queue, and killing a job that has begun execution. The job control commands differ slightly between TOPS-10 and TOPS-20; but with both operating systems, the commands allow you to submit jobs to the system and handle them thereafter.

The commands associated with job control differ from the batch and SPRINT commands. As system commands, the job control commands refer to and affect your job as a whole; they are not concerned with the internals of your control file as are the batch and SPRINT commands.

Whenever you submit a batch job, the system places it in a queue or hopper where it waits for the batch controller to select it for execution. Once the job has executed, an entry for it is made in one of the output queues. There is one input queue for jobs: the batch input queue. There are several output queues: the card punch, plotter, paper tape, and print queues. Your particular system may not have devices associated with all of these output queues, however.

The following sections describe the commands that give you access to jobs you have submitted to either the batch input queue or the printer output queue. Chapter 4, SUBMITTING A JOB FROM A TERMINAL, discusses how to enter a job into the batch input queue, and Section A.3.1.4 describes the PRINT command, the command that allows you to enter jobs into the printer output queue.

3.2 JOB CONTROL WITH TOPS-10

Under TOPS-10, the QUEUE-class commands are your primary interface to the batch input queue and to the various output queues on the system. The following sections focus upon the batch input queue and the print output queue. With these queues, you can substitute the SUBMIT and PRINT commands for any applicable form of the QUEUE command. Refer to the _TOPS-10 Operating System Commands Manual_ for complete information on the QUEUE command and the QUEUE-class commands, SUBMIT and PRINT.

The CANCEL and SHOW commands also provide you access to the input and output queues. Refer to the _TOPS-10 Operating System Commands Manual_ for information on these commands.

3.2.1 TOPS-10 Batch Input Queue

Use the SUBMIT command to examine, modify, and delete jobs in the batch input queue as well as to enter jobs into the queue.

It is a good idea to examine the batch input queue entry for batch jobs being processed and those waiting to be processed before deleting those jobs or modifying their parameters. To examine the batch input queue, use the following form of the SUBMIT command:

```
SUBMIT/switch
```

Where /switch represents one of the following switches:

/CHECK	This switch displays only those entries in the batch input queue associated with your project-programmer number.
/FAST	This switch eliminates the display of column headings and of parameter settings for jobs.
/LIST:arg	This switch also displays queue entries. If you do not specify an argument, a standard report is generated, as shown below. Use the following arguments to modify this standard format:
ALL	Lists characteristics of the queue entries in detail, reporting the settings for most of the parameters that apply to jobs.
FAST	Same as /FAST
JOBS	Same as /LIST without an argument.
/STREAM:n	This switch displays queue entries only for the batch stream specified by n.

You can cause the display to be written to your disk area by giving the following form of the SUBMIT command:

```
SUBMIT jobname = /LIST
```

The resulting file is called a queue listing file.

The switches above provide varying amounts of information about jobs submitted to the batch system. The /LIST:ALL switch gives the most comprehensive report on jobs:

```
.SUBMIT/LIST:ALL

Batch Queue:
Job Name  Req#  Run Time            User
--------  ----  --------  ------------------------
* SIRUS     72  00:05:00  PCO    [30,4033]        In Stream:0  /Uniq:Yes
          /Restart:Yes  /Assist:Yes  /Seq:2098
    Job# 59 Running DIRECT Runtime 0:00:40
* GAL0      73  00:10:00  ALEC CARLSON [50,5227] In Stream:1  /Uniq:No
          /Restart:Yes  /Assist:Yes  /Seq:2099
    Job# 61 Running UMOUNT Runtime 0:00:00
* BREN7     81  00:05:00  TUCKER,B [27,5342]     In Stream:2  /Uniq:Yes
          /Restart:No  /Assist:Yes  /Seq:2101
    Job# 60 Runtime 0:00:00
  TEST      83  00:05:00  TUCKER,B [27,5342]      /After: 8-Nov-88 23:00
          /Uniq:Yes  /Restart:No  /Assist:Yes  /Seq:2103
  TEST9      2  00:05:00  TUCKER,B [27,5342]/Dep:8 /Uniq:Yes/Restart:No
          /Assist:Yes  /Seq:884
There are 5 Jobs in the Queue (3 in Progress)
```

Note that the listing is divided into columns, displaying:

- o Job name.
- o Request id number. You may use this system-assigned number in identifying a job to be modified or killed. To do this, include /REQUESTID: with /MODIFY or /KILL, and give the request id number as an argument to the switch.
- o Maximum amount of time allotted for the job.
- o User name with associated project-programmer number.
- o Parameter settings for the job. These appear in the last column and continue, if necessary, on the next line.

SUBMIT/LIST provides this type of listing:

```
.SUBMIT/LIST

Batch Queue:
Job Name  Req#  Run Time            User
--------  ----  --------  ----------------------
* SIRUS     72  00:05:00  PCO    [30,4033]        In Stream:0
    Job# 59 Running DIRECT Runtime 0:00:40
* GAL0      73  00:10:00  ALEC CARLSON [50,5227]In Stream:1
    Job# 61 Running UMOUNT Runtime 0:00:00
* BREN7     81  00:05:00  TUCKER,B [27,5342]     In Stream:2
    Job# 60 Runtime 0:00:00
  TEST      83  00:05:00  TUCKER,B [27,5342]     /After: 8-Nov-88 23:00
  TEST9      2  00:05:00  TUCKER,B [27,5342]     /Dep:8
There are 5 Jobs in the Queue (3 in Progress)
```

And SUBMIT/FAST provides the following listing:

```
      .SUBMIT/FAST

      Batch Queue:
      * SIRUS     72  00:05:00  PCO    [30,4033]
      * GAL0      73  00:10:00  ALEC CARLSON [50,5227]
      * BREN7     81  00:05:00  TUCKER,B [27,5342]
        TEST      83  00:05:00  TUCKER,B [27,5342]
        TEST9      2  00:05:00  TUCKER,B [27,5342]
```

If no arguments appear in the command string (that is, if you give only the command name SUBMIT), all entries in the batch input queue are listed as with SUBMIT/LIST. If you give the command name QUEUE by itself, all entries for all the queues are listed.

3.2.2 Modifying Jobs under TOPS-10

To add or modify parameters for jobs waiting in the batch input queue, use the following form of SUBMIT:

```
      SUBMIT jobname=/MODIFY/switch(es)
```

where jobname is the name of the job as specified in the original SUBMIT command (refer to Chapter 4, SUBMITTING A JOB FROM THE TERMINAL, for a description of the SUBMIT command format), /MODIFY indicates that at least one of the job's parameters is to be changed or that a parameter is to be added for the job, and /switch(es) specifies the parameters to be changed or added. Refer to Section A.3.1.2, Modify Commands, for a list of the switches you can use with /MODIFY.

The following command modifies the /AFTER parameter that is displayed for job TEST in the first queue listing above:

```
      .SUBMIT TEST=/MODIFY/AFTER:19:00
      [1 Job Modified]
```

The listing below shows that the /AFTER parameter for job TEST is changed to 19:00.

```
.SUBMIT/LIST

Batch Queue:
Job Name  Req#  Run Time            User
--------  ----  --------  ----------------------
* SIRUS     72  00:05:00  PCO    [30,4033]      In Stream:0
    Job# 59 Running DIRECT Runtime 0:00:40
* GAL0      73  00:10:00  ALEC CARLSON [50,5227]In Stream:1
    Job# 61 Running UMOUNT Runtime 0:00:00
* BREN7     81  00:05:00  TUCKER,B [27,5342]    In Stream:2
    Job# 60 Runtime 0:00:00
  TEST      83  00:05:00  TUCKER,B [27,5342]    /After: 8-Nov-88 19:00
  TEST9      2  00:05:00  TUCKER,B [27,5342]    /Dep:8
There are 5 Jobs in the Queue (3 in Progress)
```

3.2.3 TOPS-10 Print Queue

Use the PRINT command to examine, modify, and delete jobs in the print queue as well as to enter jobs into the queue. To perform the first three functions, use the same forms of the PRINT command that were specified for the SUBMIT command, above. For example,

1. PRINT/CHECK, /LIST:, or /FAST provides a listing of the print queue entries.

   NOTE

   The print queue listing displays the maximum number of pages, not time (as the input queue listing shows), allowed for the job. This maximum reflects the value specified with the /LIMIT switch.

2. PRINT jobname=/MODIFY/switch(es) modifies the parameters for a print job.

   NOTE

   To distinguish between duplicate job names, include /REQUESTID switch in the command line when modifying or killing jobs in the print or batch input queue. You can use this switch in place of jobname.

3.3 JOB CONTROL WITH TOPS-20

The following sections describe how to examine the batch input queue and the print queue and how to cancel or modify a job in these queues. For complete information on the commands discussed below, refer to the TOPS-20 Commands Reference Manual.

3.3.1 Examining the Queues under TOPS-20

You may want to examine a job's parameters before modifying them, or before killing the job. This section explains how to do this.

The INFORMATION command reports on job and system parameters. To check the status of batch jobs being processed and those waiting in the input queue, give the following form of the INFORMATION command:

```
@INFORMATION (ABOUT) BATCH-REQUESTS
```

(You may abbreviate this command to "I B")

To examine jobs in the print queue, give the following form of the INFORMATION command:

```
INFORMATION (ABOUT) OUTPUT-REQUESTS
```

(You may abbreviate this command to "I O")

You may specify the following switches with the INFORMATION command when examining the queues:

/USER:user name	This switch, when followed by a valid user name, causes the INFORMATION command to display only the queue entries for the user named.
/ALL	This switch lists characteristics of the queue entries in detail, reporting the settings for most of the parameters that apply to your job.
/FAST	This switch eliminates the display of all switch values and column headings.
/DESTINATION-NODE:	This switch shows the output queues on the specified Common File System (CFS) node.

The following examples illustrate the use of these switches with the INFORMATION command. Note that the listings are divided into columns displaying:

o Jobname.

o Request id number. You may specify this unique system-assigned number in response to the ID prompt when using the CANCEL and MODIFY commands.

o Maximum runtime allotted for a batch job as specified with /TIME.

or

Maximum number of pages allotted for a print job as specified with /LIMIT.

o User name.

o Parameter settings for the job. These appear in the last column and, if necessary, continue on the next line.

INFORMATION (ABOUT) BATCH-REQUEST/ALL provides a comprehensive report of jobs in the batch input queue:

```
@INFORMATION (ABOUT) BATCH-REQUESTS /ALL

Batch Queue:
Job Name  Req#  Run Time            User
--------  ----  --------  ----------------------
* LINK     109  01:00:00  DNEFF.BUILD            In Stream:2  /Uniq:No
          /Restart:No  /Assist:Yes  /Seq:3773
    Job# 72 Running EXEC Last Label: LINK Runtime 0:00:00
  AVCO20     2  00:05:00  JENNESS                /Proc:AVCO  /Uniq:No
          /Restart:No  /Assist:Yes  /Seq:2923
  AVCO20     4  00:05:00  JENNESS                /Proc:AVCO  /Uniq:No
          /Restart:No  /Assist:Yes  /Seq:2924
  SOURCE     1  00:05:00  BLOUNT                 /After: 8-Nov-88  0:00
          /Uniq:Yes  /Restart:No  /Assist:Yes  /Seq:3099
  SYSERR     8  00:05:00  BLOUNT                 /After: 7-Nov-88 23:59
          /Uniq:Yes  /Restart:Yes  /Assist:Yes  /Seq:3102
  SAVE      35  00:05:00  FICHE                  /After:13-Nov-88 10:00
          /Uniq:Yes  /Restart:No  /Assist:Yes  /Seq:3709
  QAR       58  00:05:00  BLOUNT                 /After:14-Nov-88 11:34
          /Uniq:Yes  /Restart:No  /Assist:Yes  /Seq:3730
  SAMPLE   110  00:05:00  TUCKER             /Dep:3/Uniq:Yes/Restart:No
          /Assist:Yes  /Seq:3774
There are 8 Jobs in the Queue (1 in Progress)
```

INFORMATION (ABOUT) OUTPUT-REQUESTS/USER:TUCKER provides the following listing:

```
     @INFORMATION (ABOUT) OUTPUT-REQUESTS /USER:TUCKER

     Printer Queue:
     Job Name  Req#  Limit           User
     --------  ----  -----  ------------------------
       SAMPLE   110     27  TUCKER                        /Forms:NARROW
     There is 1 Job in the Queue (None in Progress)
```

I B/FAST gives the following information:

```
     @I B/FAST

     Batch Queue:
       AVCO20      2  00:05:00  JENNESS
       AVCO20      4  00:05:00  JENNESS
       SOURCE      1  00:05:00  BLOUNT
       SYSERR      8  00:05:00  BLOUNT
       SAVE       35  00:05:00  FICHE
       QAR        58  00:05:00  BLOUNT
       SAMPLE    110  00:05:00  TUCKER
```

3.3.2 Modifying Jobs under TOPS-20

The MODIFY command adds or changes switch values for a job waiting in the batch or print queue. In response to the REQUEST TYPE prompt, specify the argument BATCH to modify batch jobs; specify the PRINT argument to modify print jobs. For both types of jobs, you can specify jobname or request id number for the ID prompt.

3.3.2.1 Modifying Batch Jobs - Modify a batch job with the MODIFY command, the BATCH argument, and one or more of the switches listed in Section A.3.1.2, Modify Commands.

The following command modifies the dependency count that is displayed for job SAMPLE in the first queue listing above:

```
@MODIFY (REQUEST TYPE) BATCH (ID) SAMPLE/DEPENDENCY-COUNT:7
[1 Job Modified]
```

The listing below shows that SAMPLE's dependency count is now 7.

```
@I B

Batch Queue:
Job Name  Req#  Run Time        User
--------  ----  --------  -------------------
  AVCO20     2  00:05:00  JENNESS             /Proc:AVCO
  AVCO20     4  00:05:00  JENNESS             /Proc:AVCO
  SOURCE     1  00:05:00  BLOUNT              /After: 8-Nov-88  0:00
  SYSERR     8  00:05:00  BLOUNT              /After: 7-Nov-88 23:59
  SAVE      35  00:05:00  FICHE               /After:13-Nov-88 10:00
  QAR       58  00:05:00  BLOUNT              /After:14-Nov-88 11:34
  SAMPLE   110  00:05:00  TUCKER              /Dep:7
There are 7 Jobs in the Queue (None in Progress)
```

3.3.2.2 Modifying Print Jobs - Modify a print job with the MODIFY command, the PRINT argument, and one or more of the PRINT command switches. The command format is the same as for modifying a batch job.

3.4 CANCELLING OR KILLING JOBS

The CANCEL command withdraws requests made with a previous PRINT or SUBMIT command. Use CANCEL to stop jobs that are currently running or that are waiting in the queues.

The following command kills job SAMPLE that is shown in the queue listing above:

```
CANCEL BATCH 110
[1 Job Canceled]
```

On a TOPS-20 system, you can give the following command to see that job SAMPLE is no longer in the batch input queue:

```
@I B

Batch Queue:
Job Name  Req#  Run Time        User
--------  ----  --------  -------------------
  AVCO20     2  00:05:00  JENNESS             /Proc:AVCO
  AVCO20     4  00:05:00  JENNESS             /Proc:AVCO
  SOURCE     1  00:05:00  BLOUNT              /After: 8-Nov-88  0:00
  SYSERR     8  00:05:00  BLOUNT              /After: 7-Nov-88 23:59
  SAVE      35  00:05:00  FICHE               /After:13-Nov-88 10:00
  QAR       58  00:05:00  BLOUNT              /After:14-Nov-88 11:34
There are 6 Jobs in the Queue (None in Progress)
```

(NOTES)

1. To distinguish between duplicate job names in the input and output queues, specify the request id number for the ID prompt in the MODIFY or CANCEL command line.
2. To cancel all jobs in the queue under your name, specify * as the jobname. For example, the commands

   ```
   CANCEL BATCH *
   ```

   and

   ```
   CANCEL PRINT *
   ```

   cancel all jobs under your name in the batch input and print queues respectively.

On TOPS-10 systems, you can also cancel a batch job with the SUBMIT command:

```
.SUBMIT jobname=/KILL
```

And if you are logged in under [1,2], you can cancel a job for a particular user:

```
.SUBMIT jobname [P,PN]=/KILL
```

where [P,PN] is the project-programmer number of the desired user.

3.5 SUBMITTING RELATED JOBS

The system does not necessarily run jobs in the order in which you submit them. It dynamically computes a number of parameters to determine the running order of jobs. These parameters include those established during batch system installation, those that the operator sets, and those that you specify on the $JOB card or with the SUBMIT command. One parameter that you specify is particularly important for ensuring that jobs run in a certain order. Give this parameter with the /DEPEND: switch.

It is often useful to submit several jobs that must run in a specified order; for example, one job updates a master file before another job processes it. In such cases, the running of one job is dependent upon the completed execution of the other. Although these jobs could be combined into one large job, it may be more convenient (or necessary) to keep them distinct. For instance, different people might submit them at different times.

To coordinate the running of distinct jobs, specify the /DEPEND: switch with either the $JOB card or the SUBMIT command. The priority that /DEPEND: establishes, called the initial dependency count, becomes part of the queue entry for the job. If an input queue entry has a dependency count greater than zero, the system will not schedule the associated job to run. When the count becomes zero, the job is eligible to be run by the batch controller. In this way, the job does not run until another job or you set its dependency count to zero. This technique ensures that dependent jobs do not run out of sequence.

Alter the dependency count for a job in this manner:

TOPS-10:

```
SUBMIT jobname=/MODIFY/DEPEND:n
```

TOPS-20:

```
MODIFY (REQUEST TYPE) BATCH (ID) jobname or request
id/DEPENDENCY-COUNT:n
```

Where jobname is the name of the batch job and n represents the new dependency count value. If you place a plus or minus sign before the new value (n), that value is added to or subtracted from the dependency count. If the subtraction results in a negative number, the system treats the number as if it were zero. If you do not specify a sign, the dependency count is changed to the value you give for n.

NOTE

> To distinguish between jobs with the same name, use the /REQUESTID switch along with the switches above on a TOPS-10 system. On a TOPS-20 system, use the request id number for ID.

The priority established with the /DEPEND: switch overrides the priority you set with the /PRIORITY: switch. That is, a job will never run unless its dependency count is less than or equal to zero, regardless of the value you give for /PRIORITY:.

The following examples show how you can control the scheduling of related jobs using the /DEPEND: switch.

1. Assume that your company has a three-phase payroll processor consisting of a salary-calculating program, a check-writing program, and a final report-writing program. All three programs are submitted to the batch queue at the same time, but by different people. To make sure the programs do not run out of this sequence, the people would submit the jobs to the system as follows (with TOPS-10):

   ```
   .SUBMIT J1
   .SUBMIT J2/DEPEND:1
   .SUBMIT J3/DEPEND:1
   ```

   Where: J1 is the salary-calculating program
   J2 is the check-writing program
   J3 is the final report-writing program

The control files for these jobs would look like this:

a.
```
START::  !!!Beginning of J1!!!
.
.
.
END::   .SUBMIT J2=/MODIFY/DEPEND:0
```

b.
```
START::  !!!Beginning of J2!!!
.
.
.
END::   .SUBMIT J3=/MODIFY/DEPEND:0
```

In this example, each job sets a dependent job's dependency count to zero, making the dependent job eligible for execution. Note that it is a good idea to place the /DEPEND: switch at the end of a control file to ensure that the system has completed processing the file before a dependent job becomes eligible for execution.

2. Suppose you have seven interrelated jobs to run. You do not care about the execution order of six of the jobs, but one job must run after all of the other six. Submit the jobs as follows (with TOPS-20):

```
@SUBMIT J7/DEPENDENCY-COUNT:6
@SUBMIT J1
@SUBMIT J2
.
.
.
@SUBMIT J6
```

Where J7 is the job that must run last.

At the end of the control files for jobs J1 through J6 insert the line:

```
@MODIFY BATCH J7/DEPENDENCY-COUNT:-1
```

These six lines, in random fashion, will reduce J7's dependency count to zero, making it eligible to run.

CHAPTER 4

SUBMITTING A JOB FROM A TERMINAL

4.1 INTRODUCTION

To submit a job, defined in Section 1.1, JOBS, is to give it to the system for processing. When you submit a batch job from the terminal, it goes to the batch input queue, where it waits to be processed. This chapter discusses how to prepare a job for submittal from the terminal (as opposed to the card reader) and how to submit the job. In describing the effects of switches available with the SUBMIT system command, the chapter also explains what can happen to your job when it is in the batch input queue and after the batch controller selects it for processing.

4.1.1 Submitting a Card Job from a Timesharing Terminal

NOTE

Refer to Chapter 5, SUBMITTING A JOB FROM CARDS, for information on SPRINT, the input spooler for card jobs, and on the control cards you use to direct SPRINT.

Normally you punch a card job on cards then submit the job from the card reader. However, you can submit a card job from the terminal as well as from the card reader. Even though you have access to a timesharing terminal, you may want to submit a card job, which contains an extra set of commands for the SPRINT program, for such reasons as the following:

1. The card facility allows you to append card jobs to one another to form one large job.
2. The card facility allows you to create system-independent jobs. These types of jobs, described in Section 5.4, SYSTEM-INDEPENDENT JOBS, run under both TOPS-10 and TOPS-20 systems.
3. You may be responsible for creating and debugging batch jobs that others will submit from a card reader. You could use the terminal to do your job on-line, then punch the files on cards.

For you to submit a card job from a timesharing terminal, your disk directory must contain a file for the job. You submit the file directly to SPRINT by issuing the SUBMIT command with the /READER switch.

To create the disk file, use one of your system's on-line editors as described in Section 4.2. You may place commands, programs, and data in your disk file as well as card images of the control cards for SPRINT. Each card image may be up to 80 characters long, exclusive of any line terminators you use with the editor. The disk file should look like a card job you would submit from the card reader. However, you do not need the $PASSWORD card if the directory on the $JOB card matches the directory under which you are logged in.

4.2 CREATING AND SUBMITTING BATCH JOBS FROM A TERMINAL

You can submit batch jobs from the card reader (see Chapter 5, SUBMITTING A JOB FROM CARDS) or from the terminal. For you to submit a job from a timesharing terminal, your disk directory (or ersatz device CTL: for TOPS-10) must contain a control file for the job. When you issue the SUBMIT command, the control file is then submitted directly to the batch controller (by way of the batch input queue) from the disk.

Create the control file using one of the on-line editors available on your system. You may place any system or batch command in the control file as well as system and user program commands and data. The following illustrates the technique for using an editor to create a simple control file:

TOPS-10:

```
.MAKE JOB.CTL
*I.EXECUTE/COMPILE DATA.FOR/LIST
$$                  the system echoes the $ signs
*EX$$               on these two lines when you type ESCape.
.
```

Above, the MAKE command is given under TOPS-10 to let the system know that a new file is to be created using the TECO program. The control file name follows the MAKE command. 'I' tells TECO you want to insert information, and the EX command tells TECO to end this editing session. (See the TECO Reference Manual for details on TECO.)

After this one-line control file is created, it can then be submitted with the following abbreviated form of the SUBMIT command:

```
.SUBMIT JOB
```

The control file, JOB.CTL, compiles and executes DATA.FOR and produces a program listing. The system assumes that the extension for the control file name is .CTL unless specified otherwise with the SUBMIT command.

TOPS-20:

```
@CREATE BATCH.CTL
INPUT:  BATCH.CTL.6
00100       @COMPILE PROG.FOR/LIST
00200       @EXECUTE PROG
00300       $    This $ sign echoes when you
                 type ESCape during the edit session
*E

[BATCH.CTL.7]

@
```

Above, the CREATE command is given under TOPS-20 to let the system know that a new file is to be created using the EDIT program. The control file name follows the command. After EDIT types the second line, INPUT: BATCH.CTL.6, and the first input line number, 00100, control file lines are entered. The E command ends the edit session. (Refer to the EDIT Reference Manual for information on EDIT.)

The control file is then submitted with the command:

```
     @SUBMIT (BATCH JOB) BATCH
```

The control file, BATCH.CTL, causes the compilation and execution of PROG.FOR. It also produces a program listing. The system assumes that the control file type is .CTL unless specified otherwise with the SUBMIT command.

If an error occurs in your job, batch terminates the job. To avoid having your job terminated because an error occurs, you can specify error recovery in the control file using special batch commands. Error recovery is described in Section 2.5.

Any system command that you can use in a timesharing job can be used in a batch job with the following exceptions. The ATTACH and SET TIME-LIMIT commands are illegal in a batch job. If you include either of these commands in your job, batch processes the command and the TOPS-10/TOPS-20 command processor places an error message into your log file. Your batch job terminates unless you specify error recovery.

Do not include a LOGIN command in your control file since batch logs in the job for you. If you put in a LOGIN command, your job terminates. In addition, you do not need to include a LOGOUT command. Batch logs out your job automatically when it reaches the end of your control file.

4.3 THE SUBMIT COMMAND

The previous section introduced you to the SUBMIT command; this section gives you a more detailed description of the command and discusses the switches that you may specify with it. Use the SUBMIT command to submit files to be run as batch jobs. The submitted files, called control files, contain, in addition to batch commands, the system commands, program commands, and program data that you would type if you were to type the job at a terminal.

NOTE

TOPS-20

The SUBMIT command is an EXEC command that will not destroy your core image; you can CTRL/C out of a program, submit a batch job, and return to the program. Also, you may use command recognition with the SUBMIT command. (Refer to the TOPS-20 User's Guide for a description of command recognition.)

All programs and data that are to be processed when the job is run must be made up in advance or be generated during the running of the job. You can have the programs and data on magnetic tape, but if you do, you must include the TOPS-10/TOPS-20 commands MOUNT and DISMOUNT in your control file so that the operator will mount and dismount the tape(s) to be read. (Refer to Section 4.4.2, Reading from or Writing to a Tape File, for an example of a control file with these two commands.)

If your programs and data reside on an on-line disk, you need not include the MOUNT and DISMOUNT commands, as there is no action required by the operator.

The SUBMIT command format varies between TOPS-10 and TOPS-20. The switches that apply to SUBMIT are virtually identical for both systems, however. Section 4.3.1 covers the SUBMIT command for TOPS-10; Section 4.3.2 covers the SUBMIT command for TOPS-20. Section 4.3.3 describes all of the switches applicable to SUBMIT for both systems. Refer to your commands reference manual for complete details on the SUBMIT command.

4.3.1 The SUBMIT Command with TOPS-10

The SUBMIT command:

Leaves the terminal in monitor mode.

Does not require LOGIN when you wish to only examine the queues.

Runs the QUEUE program.

The SUBMIT command is equivalent to the following form of the QUEUE command:

```
.QUEUE INP:jobname=control file,log file
```

Command Format

```
.SUBMIT jobname/switches=/switch(es) control file/switch(es),
                         log file/switch(es)
```

Where:

jobname	is the name of the job being entered into the queue.
/switch(es)	are keywords chosen from section 4.3.3, indicating your choice of SUBMIT command options. They have different effects according to their position in the command line. Placed before the equal sign (=) they serve as queue-operation switches (such as /DEPEND), affecting the job as a unit. Placed before the control file name they apply to both the control file and the log file; otherwise they act only on the nearest preceding filename:

```
.SUBMIT JOB=/switch1/switch2,file.ctl/switch3,file.log/switch4
```

	In the line above, switch1 and switch2 apply to both the control file and the log file. Switch3 applies only to the control file and switch4 applies only to the log file.
control file	is the name of the control file. This file contains all monitor-level and user-level commands for processing by the batch controller. If you do not include a device in the control file specification, and if the file is not found in your search list, the SUBMIT command searches ersatz device CTL: for the control file.
log file	is the name of the log file. This file is used by the batch controller to record its processing of the job.

Only the two files mentioned above can be specified in a request to the batch input queue under TOPS-10. The name of the control file is required; the log file name is optional and, if omitted, will automatically be named the same as the control file but with the .LOG filename extension. If the job name is omitted, it defaults to the name of the log file, if present, or the name of the control file. If an extension is omitted, the following are assumed:

.CTL for the control file
.LOG for the log file

4.3.2 The SUBMIT Command with TOPS-20

Command Format:

```
@SUBMIT (BATCH JOB) /switch(es) control file/switch(es),...
```

Where:

switch(es)	are keywords chosen from Section 4.3.3, indicating your choice of SUBMIT command options. They have different effects according to their position in the command line: placed before all names of control files, they apply as defaults for all files specified in the command, otherwise they act only on the nearest preceding filename:

```
@SUBMIT/switch1/switch2 name1.CTL/switch3,name2.CTL/switch4
```

	In the line above, switch1 and switch2 apply to both control files. Switch3 applies only to the first control file and switch4 applies only to the second control file.
control file	is the name of the control file. The default file type is .CTL.
,...	means that after a comma you can give more arguments (control file and switches) of the form already shown.

NOTE

The SUBMIT command with TOPS-20 allows an arbitrary number of file specifications, all of which are interpreted as batch control files. For example, the command SUBMIT FILE.CTL, FILE.LOG submits two batch jobs, one with control file FILE.CTL and one with control file FILE.LOG. To specify a log file for a batch job, use the /LOGNAME switch as described below.

4.3.3 Switches for the SUBMIT Command

The following is a brief description of each of the switches you can use with SUBMIT. You use the switches to define limits for your job. Such limits as pages of output and the time that your job will run can be specified as switches. If a switch is specified more than once, the input specified in the last switch is used. You can put a switch anywhere in a command string. Appendix B explains these switches in detail.

If you repeatedly use the same switches and the same switch values, then you may wish to set default values for those switches with the SWITCH.INI file (TOPS-10). Under TOPS-20, you can set the switch values with the SET DEFAULT command. Appendix B discusses this topic.

Switch	Meaning
/ACCOUNT:name	The job's CPU time will be charged to the account specified. Name can have a maximum of 39 characters.
/AFTER:date-time	The job cannot be run until after the specified date and time.
/AFTER:+hh:mm	The job cannot be run until after the time the job is submitted plus the amount of time specified. To run the job after a given amount of time has elapsed, specify the time in the form +hh:mm (for example, /AFTER:+1:00 to run the job an hour from now). If you omit the switch, batch schedules your job as it normally would using its defaults. If you omit the colon and/or value, the monitor responds with an error message and terminates the SUBMIT command.
/ASSISTANCE:arg	This switch specifies that the job needs or does not need operator assistance. The arguments are YES and NO.
/BATCH-LOG:arg (-20) /BATLOG:arg (-10)	The log file is written according to the arguments you specify with this switch: APPEND causes the file to be appended to an existing one of the same name; SPOOL causes the file to be written to the system spooling area; SUPERSEDE causes the file to overwrite another one of the same name.

/BATOPT:name (-10)	Specifies a LOGIN option line to read for LOGIN switches to apply to the batch job. The option name that you specify with the /BATOPT switch must match a line in the SWITCH.INI file that appears as: LOGIN:option-name/switches
/BEGIN:n	Processing will begin on the nth line of the control file. If you do not specify this switch, processing starts at the first line.
/CARDS:nn	The job can punch a maximum of nn cards (up to 10,000 in decimal).
/CHECK (-10)	This switch lists all the jobs that are in the batch input queue under your project-programmer number.
/CONNECTED-DIRECTORY:<directory> (-20)	This switch allows your job to run under a directory different from the directory under which you submitted the job. This switch is valid for an enabled wheel or an enabled operator only.
/CORE:(-10) nnK,nnP, nnB, or nnW	The job can use a maximum of nnK words of memory, or the specified number of memory pages (nnP), words (nnW), or blocks (nnB). These values are expressed in decimal. The default is usually 32K.
/DELETE	This switch deletes the file after the job has finished execution. With TOPS-10, if the file is the batch log file, it will not be deleted until it has been printed.
/DEPEND:nn	This switch specifies the initial interjob dependency count (in decimal) for the job. Refer to Section 3.5, SUBMITTING RELATED JOBS, for details about this switch.
/DEPENDENCY-COUNT:nn (-20)	Same as /DEPEND.
/DESTINATION:node(-10) /DESTINATION-NODE:node(-20)	These switches specify the DECnet network node or IBM remote job entry station to whose line printer the log file and all spooled output are to be sent. The node name must be of six or fewer characters. On TOPS-20 systems, this switch has no effect if the node name specified is a DECnet node name. On TOPS-10 systems, you can use this switch to get a queue listing of batch jobs whose output is being sent to the specified node.

/DISTRIBUTION:"text" (-10)	Specifies text to place in the distribution field on the banner page of output listings. For batch input requests, the distribution text is printed on the banner pages of the log file listing. You can use this field to include mailing information or the location where the operator should leave the listing. The text field may be up to 39 alphanumeric characters, including punctuation and spaces if the text is placed in quotation marks.
/ERPROTECTION (-10)	This switch produces an error message if processing the job requires a violation of protection codes. This is the default function.
/FAST (-10)	This switch displays batch input queue entries in an abbreviated format.
/FEET:nn	The job can punch a maximum of nn (decimal) feet of paper tape.
/JOBNAME:name	This switch assigns a 1- to 6-character name to the job.
/KILL (-10)	This switch allows you to cancel or kill the running of a job. (Refer to Section 3.4, CANCELLING OR KILLING JOBS, for details.)
/LIST:arg (-10)	This switch lists the entries for the batch input queue on your terminal. The arguments are ALL, JOBS, and FAST. (See Section 3.2, JOB CONTROL with TOPS-10 for details.)
/LOGDISPOSITION:arg (-20)	This switch specifies the disposition of the log file after the batch job has been processed. The arguments are DELETE and KEEP (default).
/LOGNAME:name (-20)	This switch specifies a 1- to 6-character alphanumeric name for the log file. The filename can be followed by a 3-character filename type. /LOGNAME has no effect when you also specify the /READER switch.
/METERS:n (-10)	This switch specifies the maximum number of meters of paper tape the job can punch.
/MODIFY (-10)	This switch allows you to modify a job's parameters. (Refer to Section 3.2.2, Modifying Jobs Under TOPS-10, for details.)

/NEW (-10)	This switch allows the system to accept your request even if the file does not yet exist. Use this switch, for example, if the disk structure containing the file is not yet mounted.
/NONEW (-10)	Opposite of /NEW.
/NONOTIFY (-10)	Same as /NOTIFY:NO.
/NOTIFY:arg	This switch directs the system to print a message at your terminal letting you know when your job has completed. The arguments are YES and NO.
/OKPROTECTION (-10)	This switch suppresses the error message when a protection code violation occurs.
/OPTION:option (-10)	This switch executes the line beginning with 'SUBMIT: option' in the SWITCH.INI file. 'Option' is the identifier you supply for this line in the SWITCH.INI file. Refer to the discussion of SWITCH.INI in your system commands manual for details.
/OUTPUT:arg	This switch determines whether the log file is printed. The arguments are: LOG (TOPS-10 only; print the file), NOLOG (suppress printing), ERROR (TOPS-10)/ERRORS (TOPS-20)(print only if an error occurs), and ALWAYS (TOPS-20 only; equivalent to LOG). LOG or ALWAYS is the default.
/PAGES:nn	This switch specifies the maximum number of pages (in decimal) that the job can print. The default limit is 200 pages; however, the system administrator may change this value at batch system generation time.
/PATH:[dir] (-10)	This switch allows your job to run under the directory you specify.
/PHYSICAL (-10)	This switch suppresses logical device names for the specified file.
/PRESERVE	This switch causes the system to save the control file after completion of the job. With TOPS-10, you can use this switch to save the log file.

/PRIORITY:nn	This switch assigns the job a priority value that determines when the job will run relative to other jobs in the specific queue. This priority is called the external priority. See Appendix B for more information on this switch.
/PROCESSING: (-10) /PROCESSING-NODE: (-20)	These switches specify the IBM host system on whose CPU the JCL batch job is to be run. The node name must be of six or fewer characters. On TOPS-20 systems, if the local node name is specified, then the batch job is processed locally. However, if any other DECnet node name is specified, then the batch job is queued by QUASAR but never executed. You can also use this switch to obtain a batch queue listing for the specified node.
/PROTECTION:arg (-10)	This switch specifies a protection code for the log file or for a batch queue listing file that is written to the disk.
/READER	This switch causes a file of card images on disk to be processed as if you had punched the file on cards and submitted it by way of the card reader.
/RESTART:arg (-10) /RESTARTABLE:arg (-20)	This switch specifies whether the job should be restarted after a crash has occurred. The arguments are YES and NO. NO is the default. YES is the default if you use this switch without an argument. (Refer to the description of the CHKPNT batch command in Section 2.6, BATCH COMMANDS, for details on these switches.)
/NORESTART (-10)	Same as /RESTART:NO
/SEQUENCE:n	This switch allows you to specify an identifying decimal number for the job. If you omit this switch, the system assigns a unique "sequence" number to the job. The system also assigns a request id number to each job. It is recommended that you use the request id number when modifying or killing jobs with duplicate names.
/SITGO (-10)	This switch causes your job to be processed by the SITGO compiler.
/STREAM:n (-10)	This switch lists queue entries for the specified batch stream.

/TAG:label	This switch causes processing to begin at the specified label. Label is a 1- to 6-character label for a command line in the control file.
/TIME:hh:mm:ss	This switch sets specific limits upon the amount of CPU time allowed the job. The default is 5 minutes. If you need more than 5 minutes of CPU time, you must include the /TIME switch in the SUBMIT command to indicate the approximate amount of time that you will need. If you specify the switch without the colon and a value, batch assumes that you need one hour of CPU time. If you do not specify enough time, batch terminates your job when the time is up. The value in the /TIME switch is given in the form hh:mm:ss (hours:minutes:seconds). Refer to Section B.2 for additional information on time and date switches.
/TPLOT:mm	This switch specifies the maximum amount of plotter time that the job can use (in minutes).
/UNIQUE:arg	This switch gives a "uniqueness" value for the associated job. If the argument is either YES or 1 (default), then no other job can run concurrently (if submitted from the same connected directory) with this job. If the argument is NO or 0, then other jobs can run concurrently with this job.
/USER:name (-20) /USERNAME:name (-10)	This switch allows an enabled operator or an enabled wheel to run a job under a user name different from the name under which the job was submitted. Name is a valid user name.

4.4 SAMPLE JOBS

The following examples show how to perform various functions from a batch control file. The examples offer methods for compiling and running user programs, for transferring data, and for using system programs and utilities.

Unless noted, the examples apply to both TOPS-10 and TOPS-20. Simply substitute the appropriate system prompt character for the existing one where necessary.

4.4.1 Reading from or Writing to a Disk File

To compile and run a program, store the following system commands in your control file:

COMPILE name/language	Causes the program to be compiled or assembled by the appropriate compiler or assembler. 'Name' is the name of the disk file containing the program. 'Language' can be the name of any language supported by the COMPILE command, for example: ALGOL COBOL FORTRAN MACRO
EXECUTE name	Executes the previously compiled or assembled program.

For example, this control file

```
@TYPE BATCH.CTL
00010      @COMPILE TEST1.FOR/LIST
00020      @EXECUTE TEST1
```

will compile and execute the FORTRAN program below. The program reads a file of numbers (DATA.FIL) from your disk area, computes their squares, and records the results in the log file.

```
@TYPE TEST1.FOR
00010               INTEGER*4 A,B
00020      3        FORMAT (I3)
00030               OPEN (UNIT=01,FILE='DATA.FIL')
00040      4        FORMAT (1X,I3)
00050               DO 5 A = 1,5
00060               READ (01,3) B
00070               B = B**2
00080               TYPE 4,B
00090      5        CONTINUE
00100               CLOSE (UNIT=01)
00110               STOP
00120               END
```

Note that the program provides instructions for the system to read from and write to the disk file. Choose similar instructions from the particular programming language you are using.

After the control file to compile and execute the FORTRAN program is created and saved, submit the job to batch.

```
@SUBMIT BATCH/JOBNAME:MYJOB/TIME:20/PAGES:750/AFTER:10:00<RET>
```

When the monitor reads this request, it assumes the following:

1. The name of the job is MYJOB.
2. The name of the control file is BATCH.CTL.
3. The log file will be named BATCH.LOG.
4. The log file will be left in your disk area after it is printed.
5. The control file will be left in your disk area.

6. The maximum number of pages that can be printed is 750.

7. The maximum amount of CPU time that the job can use is 20 minutes on TOPS-20 systems and 20 seconds with TOPS-10. (Refer to Section B.2 for information on time and date switches.)

8. The job will process only after 10:00 A.M.

If you make an error in the SUBMIT command when you submit this job, the monitor displays an error message on your terminal to explain your error so that you can correct it.

NOTE

> An alternative to identifying the program type with the /language switch is to use the file extensions/types .ALG, .CBL, .FOR, or .MAC. For example, COMPILE TEST/FORTRAN is equivalent to COMPILE TEST.FOR.

4.4.2 Reading from or Writing to a Tape File

To read from or write to a tape file, use the following system commands in addition to the COMPILE and EXECUTE commands discussed in the previous example:

MOUNT	Causes a tape mount message to be sent to the system operator. The batch job will not be processed beyond this command until the operator has mounted the appropriate tape. If you use this command, be sure to give your tape to the operator before you submit your job.
DISMOUNT	Rewinds the magnetic tape onto one reel and deassigns the tape drive from your job. The tape drive is now available for use by other jobs.

Your program should provide the necessary tape read/write statements. Choose these statements from the particular programming language you are using. For example, the following FORTRAN program reads a file of numbers (DATA.FIL) from a tape file, computes their squares, and records the result in the log file:

```
@TYPE TEST2
00010              INTEGER*4 A,B
00020       3      FORMAT (I3)
00030              OPEN (UNIT=16,DEVICE='TAPE',FILE='DATA.FIL')
00040       4      FORMAT (1X,I3)
00050              DO 5 A = 1,5
00060              ACCEPT 3,B
00070              B = B**2
00080              TYPE 4,B
00090       5      CONTINUE
00100              CLOSE (UNIT = 16)
00110              STOP
00120              END
```

This is the control file for a job that will compile and execute the program above:

```
@TYPE BATCH1.CTL
00010      @MOUNT TAPE: REEL1
00020      @COMPILE TEST2/FORTRAN
00030      @EXECUTE
00040      DISMOUNT TAPE:
```

4.4.3 Reading Data from the Control File

A program is able to read data from the input stream (from the control file) because the batch controller makes the input stream appear as a timesharing terminal to the program. Therefore, to read data from the input stream, the program must use a terminal read statement. For example, a FORTRAN program would use an ACCEPT statement. Also, if the program uses a terminal write statement (such as TYPE in FORTRAN), the output is written to the log file.

The following FORTRAN program reads a list of numbers from the job's control file. The program calculates the square of each number and records the result in the log file:

```
@TYPE TEST1
00100              INTEGER*4 A,B
00200      3       FORMAT (I3)
00300      4       FORMAT (1X,I3)
00400              DO 5 A = 1,5
00500              ACCEPT 3,B
00600              B = B**2
00700              TYPE 4,B
00800      5       CONTINUE
00900              STOP
01000              END
```

This control file:

```
@TYPE BATCH2.CTL
00010      @COMPILE TEST1/FORTRAN
00020      EXECUTE TEST1
00030      *3
00040      *5
00050      *7
00060      *8
00070      *9
```

will compile and execute the program above. The COMPILE and EXECUTE system commands perform these functions as in previous examples. Note that the control file also contains the data for the program. Each data line begins with an asterisk (*). (Refer to Sections 2.2, LINE IDENTIFIERS, and 2.5.2, Line Identifiers and Error Processing, for supplementary information.)

4.4.4 Using an Interpretive Language (APL, BASIC, CPL, SNOBOL)

When using an interpretive language, place the following commands in the control file:

@ name	Names the interpreter (APL, BASIC, CPL, SNOBOL) for the language you wish to use.
*statement *statement *statement	Specifies program statements or interpreter commands.
*monitor statement	Returns control from the interpreter to the system. The monitor statement format varies according to the language you use as follows:

Statement	Language
MONITOR	BASIC, CPL
)MONITOR	APL
system prompt character (use without an asterisk)	SNOBOL

The following CPL program reads numbers from the batch control file, calculates the square root of each number, and prints the result in the log file:

```
@TYPE TEST.CPL
10.        DCL X FLOAT;
20.        DO I = 1 TO 5;
30.        GET LIST (X);
40.        X = X**(1./2);
50.        PUT SKIP LIST (X);
60.        END;
```

This is the control file for the job that will run the program above:

```
@TYPE BATCH3.CTL
00010      @CPL
00020      *LOAD 'TEST.CPL' NUMBER 10
00030      *XEQ
00040      *256,4,64,65,9
00050      *MONITOR
```

4.4.5 Using a System Program or Utility (TOPS-20)

When using a system program or utility, place the following in your control file:

@ name	Names the system program or utility you wish to run. If a compiled and linked program (one with a .EXE file extension/type) is stored in your disk area, you may run it by specifying: @ RUN name
*command	Specifies commands or data (if required) to be read by the program.

For example, the following illustrates a control file for a job that uses the SORT/MERGE program:

```
@TYPE BATCH4.CTL
00010      @SORT
00020      *OUT.FIL=IN.FIL/RECORD:80/ALPHA/ASCII/KEY:7:10:A
```

NOTE

TOPS-20:

A TOPS-20 system command (one of the commands discussed in the TOPS-20 Commands Reference Manual, such as COPY) differs from a system program or utility in that you must precede input lines for a command with an at sign (@) rather than an asterisk. (See Section 2.2, LINE IDENTIFIERS, for details.)

4.4.6 Printing a File

You need not submit a print request to the batch input queue. You can submit it directly to the print queue with the PRINT system command. For example, typing the following command on a timesharing terminal would produce a printout of the file TEST.FOR:

```
PRINT TEST.FOR
```

However, your batch job can also submit a print request to the batch queue. For example, the following command, when placed in a batch control file will print the file TEST.FOR:

```
@  PRINT TEST.FOR
```

Many switches are available for use with the print command. Some of them are described in Appendix B.

4.4.7 Suppressing Printing of the Log File

As a paper-saving measure, it is often desirable to suppress printing of the log file. Accomplish this by using the following switch with the SUBMIT command:

```
SUBMIT name/OUTPUT:NOLOG
```

The /OUTPUT:NOLOG switch suppresses printing of the log file. Since a batch job submitted from a timesharing terminal always writes (by default) a copy of the log file in your disk area, you can examine that copy of the log file rather than the printed copy.

Alternatively, you may have the log file printed only if an error occurs during execution of the job. Accomplish this by using the following switch with the SUBMIT command:

```
SUBMIT name/OUTPUT:ERRORS (TOPS-20)

SUBMIT name/OUTPUT:ERROR (TOPS-10)
```

Again, whether the log file is printed or not, a copy of it is stored in your disk area.

CHAPTER 5

SUBMITTING A JOB FROM CARDS

5.1 INTRODUCTION

A card job is one whose storage medium is generally a deck of 80-column cards. A file of card images on disk can also constitute a card job. (For information on creating and submitting this type of job, refer to Section 4.1.1, Submitting a Card Job from a Timesharing Terminal.)

The CDRIVE program reads the cards that you submit to the card reader and writes them on the disk. SPRINT, the input spooler, processes the cards, creating a batch control file, then enters the job into the batch input queue (or the print queue if it is a print job), where it waits to be selected for processing. BATCON processes the job by acting upon the statements contained in the batch control file.

Your control-card input to batch may contain any combination of commands. These commands are in four groups as follows:

1. System commands, which consist of commands in a format similar to what you would issue for the same command on a timesharing terminal. Examples of these commands are COPY, DEASSIGN, PRINT, and RENAME.

2. System program commands, which consist of commands that pertain to a system or user program. An example is the command to the FILCOM program to specify files to be compared.

3. Batch commands, as described in Section 2.6.

4. Batch control-card commands, some of which are listed below:

$JOB card	identifies your project-programmer number (TOPS-10) or user name (TOPS-20).
$PASSWORD card	gives your password (not needed if submitting a card job from the terminal and the directory under which you are logged in matches the directory on the $JOB card).
$EOJ card	marks the end of the card deck.

This chapter describes SPRINT's activities, introduces the control cards you use to control SPRINT, and explains how you can create system independent jobs that will run under either TOPS-10 or TOPS-20. Refer to sections 5.2, 5.3 and 5.4 for information on these topics. Sample card jobs are provided at the end of the chapter.

5.2 SPRINT

After CDRIVE reads your sequential input stream and places the card images in a disk file, SPRINT separates the input into appropriate files, according to the control cards that you supply. In this way, SPRINT prepares your job for batch processing, but it does not initiate any batch processing itself. This processing is done by the batch controller.

SPRINT creates three types of files during its operations:

1. Files containing information that the batch controller passes to the running job.

2. The batch control file.

3. The job's log file.

All three files are placed into the disk directory you specify with the $JOB card.

The first type of file is either a data file or a source program. SPRINT creates these files according to the control cards in the input stream. The File Cards, described in the following section, are the specific control cards that cause SPRINT to create and copy information into these files. Commands in the batch control file determine how the system acts upon programs and data.

The second type of file that SPRINT creates, a batch control file, contains all the monitor, user, and batch controller commands from the original card file. It also may hold some additional batch controller and/or monitor commands that SPRINT entered as a result of its operations. For example, when SPRINT reads the $GOTO command, it places the GOTO batch command into the control file; $MACRO causes SPRINT to insert the COMPILE monitor command into the control file. This file is subsequently processed by the batch controller.

The third type of file that SPRINT creates is a log file for the job. Into this file goes a report of SPRINT's and BATCON's activities along with a report of any operator intervention during processing of the job. The log file is automatically printed unless you submitted the job using the NOLOG or ERROR options with the /OUTPUT: switch. The system usually deletes the log file after printing it. Chapter 6, BATCH SYSTEM OUTPUT, further discusses the log file.

5.3 CONTROL CARDS

SPRINT enters commands into the control file when you use certain control cards. You intersperse control cards through the input stream to direct SPRINT in separating the input into the appropriate files. The control cards contain a dollar sign ($) in column 1 and an alphabetic character in column 2. The command must be followed by at least one space, which can then be followed by other information. These are the only cards that SPRINT reads and interprets; the remainder of the input goes to a data file, a source program file, or to the batch control file.

A batch job can do almost anything a timesharing job can do. If you wish to perform complicated tasks, you may include system commands in your deck to direct batch to execute these tasks. Section 5.4 describes the way to include system commands for the desired control.

The $JOB card, the $PASSWORD card, and the $EOJ card are required for all jobs. The $JOB card must be the first card in the deck and must be immediately followed by the $PASSWORD card. The $EOJ card must be the last card in the deck.

Comments can also be included either on separate cards or on cards containing other information. If the entire card is to contain a comment, the card should contain a dollar sign ($) in column 1 and an exclamation point (!) in column 2. The exclamation point (!) is called the comment character. If the card contains a command followed by a comment, only the exclamation point (!) should precede the comment. If the comment is too long to be contained on a single card, begin the next card with a dollar sign ($) in column 1 and the exclamation point (!) in column 2 and then continue the comment.

Note that if you create your own batch control file at a terminal, you bypass SPRINT and, therefore, cannot use control cards. Recall that you can create a card job at the terminal, however, and submit that file from the terminal using the SUBMIT command with the /READER switch. (See Section 4.1.1 for information on the topic.)

5.3.1 File Cards

Of the control cards, the $CREATE, $DATA, $DECK (-10), $language (without a filename specified), and $RELOCATABLE (-10) cards are also referred to collectively as File Cards. The File Cards cause subsequent cards up to the next control card to be placed into a disk file.

The typical card deck includes a language card ($ALGOL, $COBOL, and so forth) immediately prior to the source program. This language card informs batch of the system program to be employed for processing (compiling) the succeeding cards. The $DATA card likewise immediately precedes the data cards to inform batch that the succeeding cards contain data for the program. In both cases, the information is stored (on a spooled card reader file) to establish program files and data files. The $DATA card also causes batch to execute the program, using the data cards as input. The $EOJ card informs batch that all cards pertaining to the job have been entered. At this time batch has access to the program to be compiled and the data to be used by the program; it knows what compiler or assembler is to be used, and has built a control file containing the EXECUTE command so that the program will be run.

By using system commands and the $CREATE control card, you can process any program that does not have special control cards. You put a $CREATE card in front of a program, data, or any other group of cards to make batch copy the cards into a disk file and, if you request, to print the file on the line printer. You put the $TOPS10/$TOPS20 card in front of monitor and batch commands to cause batch to copy these commands into the control file. The $CREATE card and the $TOPS10/$TOPS20 card are described in detail in Section 5.3.7

For example, a BASIC program does not have a specific control card. To run a BASIC program under batch from cards, you can combine the $CREATE card and the $TOPS10/$TOPS20 card with system commands. You can also use a $CREATE card to copy the data which a BASIC program will use. The $DATA card cannot be used, because the $DATA card puts an EXECUTE command into the control file, and BASIC does not use the EXECUTE command to run. The $TOPS10/$TOPS20 card causes batch to copy the monitor commands into the control file.

5.3.2 Abbreviations

You need specify only the first few letters of a command or switch name; as long as the name is unique within its class, the system accepts it. The first characters are generally sufficient to ensure uniqueness. However, it is recommended that you completely spell out command and switch names because future releases of the batch system could introduce new commands or switches that conflict with your abbreviations. As a result, many of your card decks could become invalid.

5.3.3 Comments

You can use the system's standard comment conventions on control cards. The exclamation point (!) or semicolon (;) indicates the beginning of a comment.

The system treats all the characters following the exclamation point or semicolon through the end of the card as a comment. While a comment field cannot actually be continued, you can accomplish the same effect by using an additional card that contains only the rest of the comment.

Below is an example showing how you can use comments.

```
$DECK MYFILE.FOR                !Beginning of source file
        .
        .
        .
$EOD                            ;End of source file
$FORTRAN MYFILE.FOR
$!Begin compilation
```

5.3.4 Continuation of Operative Information

You may continue operative information from one control card to the next by placing a hyphen as the last non-TAB, non-space character before the end-of-card or before the beginning of a comment (if present). That is, the hyphen indicates continuation onto the next card if it immediately precedes one of the following:

1. The end of the card.

2. A string of spaces and/or TABs followed by the end of card.

3. An exclamation point or semicolon followed by a comment.

4. A string of spaces and/or TABs followed by a comment.

The following two examples illustrate the continuation of operative information, first without comments and then with comments, respectively.

1.
   ```
   $DEC MYFILE.ONE/ASCII-
   /SUPPRESS
   ```

2.
   ```
   $DECK MYFILE.ONE-!This is a comment
   /ASCII/SUPPRESS
   ```

5.3.5 Parameter Defaults

All defaults for control card parameters are established during system installation.

5.3.6 End-Of-File Card

When you submit your job from the card reader, you can use the standard end-of-file card or the end-of-job control card ($EOJ) to signal the end of the job. Columns 1 and 80 of the end-of-file card have punches in rows 12, 11, 0, 1, 6, 7, 8, and 9, with rows 2, 3, 4, and 5 blank. These punches are necessary only in column 1, but it is suggested that you punch both columns 1 and 80 so that the card reader will recognize the card in any orientation.

5.3.7 Control Card Descriptions

This section lists and describes all of the control cards/SPRINT commands in alphabetical order.

NOTE

> While processing the control file, SPRINT replaces the dollar sign ($) that precedes the following SPRINT commands with the system prompt character (. for TOPS-10, @ for TOPS-20). By this conversion, these SPRINT commands become batch commands capable of being processed directly by the batch controller. Thus, for instance, $ERROR and $NOERROR become the batch commands .ERROR or @ERROR and .NOERROR or @NOERROR. Refer to the corresponding batch commands in Section 2.6 for complete descriptions of these commands:

```
$BACKTO          $NOERROR          $REQUEUE
$CHKPNT          $NOOPERATOR       $REVIVE
$ERROR           $OPERATOR         $SILENCE
$GOTO            $DUMP
```

$ALGOL	See the $language card description.
$BACKTO	See the BACKTO batch command description in Section 2.6. Refer also to NOTE at the beginning of this section.
$BLISS	See the $language card description.
$CHKPNT	See the CHKPNT batch command description in Section 2.6. Refer also to NOTE at the beginning of this section.
$COBOL	See the $language card description.

$CREATE

Function

$CREATE is a File Card that directs the input spooler to copy all cards following it into a data file on disk. If the appropriate switch is included, batch also prints this file on the line printer.

Characteristics

The cards that you may use to terminate the copying vary according to whether you use the /DOLLARS or the /NODOLLARS switch. Refer to the brief switch descriptions below, or to Appendix B for a more detailed explanation of the /DOLLARS and /NODOLLARS switches.

By using the appropriate switches to the $CREATE card you may further direct the input spooler to send the newly created disk file to any output queue.

The $CREATE card followed immediately by a $EOD or any other appropriate terminator (depending upon your use of the /DOLLARS or /NODOLLARS switch) will create a null file on disk.

Card Format

```
$CREATE dev:<directory>filename.ext /switch(es)
```

Where:

dev:	is a device name that can be a system logical name. The default is DSK:.
filename.ext	is the user-assigned name and extension/type of the file to be created. If omitted, the filename will be CR???? (where ? represents a character randomly chosen by SPRINT to produce a unique filename) On a TOPS-10 system, SPRINT creates the file DK???? if you omit the filename with $DECK. (See the section, Related Card, below.)
<directory>	is a directory name, which may be different from the one specified on the $JOB card. If omitted, the directory specified on the $JOB card is used.
switch(es)	are switches that control the mode of reading and interpreting of the input stream and the placement of the output (if desired) in the appropriate output queue. All available switches are described below.

Switches

The following is a brief description of the switches that you can use on the $CREATE card. These switches are explained in detail in Appendix B.

Switch	Meaning
/026	The card deck is read as 026 card code.
/ASCII	The card deck is read as ASCII card code.
/BCD	This switch has exactly the same effect as the /026 switch; that is, the card deck is read as 026 card code.
/BINARY (-10)	The card deck is read in checksummed-binary card format.
/CPUNCH	The disk file created by the $CREATE card is placed into the card punch output queue.
/DOLLARS	Only $JOB, $EOD, and $EOJ are recognized as control cards. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or user data.
/IMAGE:nn	The card deck is read in image mode. The switch may be followed by a decimal number in the range 2 through 80. Refer to Appendix B for more complete information on this switch.
/PLOT (-10)	The disk file created by the $CREATE card is placed into the plotter output queue.
/PRINT	The disk file created by the $CREATE card is placed into the line printer output queue.
/PROTECTION:nnn (-10)	The file being created is protected according to the protection code you specify with nnn (a 1- to 3-digit octal code).

/SUPPRESS	When batch reads the cards in your deck, it normally copies everything on the card up to column 80 (or up to any column you specify on the /WIDTH switch). However if you do not want trailing spaces copied (to save space on the disk, for example), you can tell batch, by means of the /SUPPRESS switch, not to copy any trailing spaces into the disk file.
/NOSUPPRESS	Trailing blanks are not suppressed.
/TPUNCH	The disk file created by the $CREATE card is placed into the paper-tape punch output queue.
/WIDTH:nn	Only columns 1 through nn (inclusive) of each card are read. Specify nn as a decimal number.

Related Card

$DECK

$DECK performs the same function as $CREATE but is valid only with TOPS-10. It is recommended that you use the $CREATE card.

$DATA

Function

$DATA is a File Card that causes SPRINT to copy the data deck that follows it into a spooled card-reader file and to insert an EXECUTE system command into the control file.

Characteristics

SPRINT maintains a list of filenames of all source or relocatable programs that have been processed since the beginning of the job or since the last $DATA or $EXECUTE card was read. Each time a program is copied by SPRINT, its name is placed in the list. When the $DATA card is read, SPRINT places an EXECUTE command into the control file and copies the filenames of the programs into the EXECUTE command string. When the next $language, $INCLUDE or $RELOCATABLE card is encountered, SPRINT clears the list of filenames so that the next entries into the list reflect only those filenames copied since the last $DATA or $EXECUTE command was read.

When the job is run, the programs are loaded and executed. No compilation is performed because the programs are either in relocatable binary form or were previously compiled because of the $language card. If two $DATA cards appear in a row, the same programs are reloaded and executed again.

When the program (comprised of the loaded programs) runs, it can read the data deck by referencing the card reader. In FORTRAN, the statement READ (2,F) refers to the card reader.

Card Format

$DATA /switch(es)

Where:

/switch(es)	are switches that control the mode of reading and interpreting of the input stream.

Switches

The following is a brief description of the switches that you can use on the $DATA card. These switches are explained in detail in Appendix B.

Switch	Meaning
/026	The card deck is read as 026 card code.
/ASCII	The card deck is read as ASCII card code.
/BCD	This switch has exactly the same effect as the /026 switch; that is, the card deck is read as 026 card code.
/BINARY (-10)	The card deck is read in checksummed-binary card format.

/DOLLARS	Only $JOB, $EOD, and $EOJ are recognized as control cards. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or user data.
/IMAGE:nn	The card deck is read in image mode. The switch must be followed by a decimal number in the range 2 through 80. Refer to Appendix B for more complete information on this switch.
/MAP	A loader map will be generated and printed.
/NOMAP	A loader map will not be generated.
/SUPPRESS	When batch reads the cards containing your data, it normally copies everything on the card up to column 80 or up to any column you specify on the /WIDTH switch. However, if you do not want trailing spaces copied (to save space on the disk, for example), you can tell batch, by means of the /SUPPRESS switch, not to copy any trailing spaces into the disk file.
/NOSUPPRESS	Trailing blanks are not suppressed.
/WIDTH:nn	Only columns 1 through nn (inclusive) of each card are read. Specify nn as a decimal number.

The defaults for all modes are reset by individual switches in other control cards such as in the $CREATE card and $JOB card.

Restrictions

This card can be used only when the programs in the job have been entered with a $language, $INCLUDE, or $RELOCATABLE card, since SPRINT maintains a list of the filenames of programs that are input with these commands. If you wish only to have the programs compiled, no $DATA or $EXECUTE card or EXECUTE command should appear in the job.

$DECK (-10)

Comment

$DECK has exactly the same function, characteristics, format, and switches as the $CREATE card. It is valid only with TOPS-10, however. It is recommended that you use the $CREATE card.

$DUMP

Function

See the DUMP batch command description in Section 2.6. Refer also to NOTE at the beginning of this section.

Card Format

```
$DUMP
```

$EOD

Function

The $EOD card terminates the input that is preceded by a File Card and that is being copied to a disk data file by SPRINT. Refer to Section 5.3.1 for information on File Cards.

Card Format

```
$EOD
```

Comment

If the $EOD card does not follow the card input that was preceded by a $CREATE card, batch recognizes the next card with a dollar sign ($) in column one as a new batch command and as the end of the card input; that is, an EOD card is assumed if one is not present.

If SPRINT is not copying input and it encounters $EOD, it simply ignores the card.

$EOJ

Function

The $EOJ card is the last card in a job deck; it terminates the job. If you omit the $EOJ card, an error message is issued. However, your job is still scheduled and may be processed if another job follows it.

Card Format

```
$EOJ
```

Related Card

Standard end-of-file card

The standard end-of-file card has the same function as $EOJ; $EOJ is recommended when running a job from a remote batch station, however.

$ERROR

$NOERROR

See descriptions for the corresponding batch commands (ERROR, NOERROR) in Section 2.6. Refer also to NOTE at the beginning of this section.

$EXECUTE

Function

The $EXECUTE card causes SPRINT to insert an EXECUTE system command into the control file. This card is used when the program requires no data or uses data already existing on disk.

Card Format

EXECUTE/switch

Where:

/switch	is a switch indicating whether a loader map is to be generated.

Switches

Switch	Meaning
/MAP	A loader map will be generated and printed.
/NOMAP	A loader map will not be generated. /NOMAP is the default.

Related Card

$DATA

$EXECUTE performs the same function as $DATA; however, no data deck follows $EXECUTE. Use $EXECUTE when there is no data or when the data file already exists on disk (for example, through the previous use of the $CREATE card).

The files to be placed in the EXECUTE command string are determined for $DATA in the same way as they are for $EXECUTE.

Comment

An $EXECUTE card following another $EXECUTE card in the control file without intervening $language cards causes the program executed by the first EXECUTE card to be loaded and executed again.

$FORTRAN

See the $language card description.

$GOTO

See the GOTO batch command description in Section 2.6. Refer also to NOTE at beginning of this section.

$IF

Comment

With $IF, SPRINT substitutes the appropriate system prompt character for the dollar sign ($), making a batch command out of the $IF SPRINT command. The function, characteristics, and format are identical for $IF and the IF batch command, with the exception noted below. Refer to Section 2.6, BATCH COMMANDS, for details on IF.

$IF allows an option for the statement parameter that is not available with the IF batch command. $IF permits you to specify another SPRINT command, such as $GOTO, for the statement parameter. This feature is useful for handling error and nonerror conditions on a system independent basis. (Refer to Section 5.4.2.1, $IF and System Independent Processing, for further information.)

$INCLUDE

Function

The $INCLUDE card causes a relocatable binary file (.REL) that already exists on disk to be loaded with your program.

Characteristics

A file specified on a $INCLUDE card is added to the list of filenames remembered by SPRINT and included in the EXECUTE command string generated by a $DATA or by a $EXECUTE card.

Card Format

```
$INCLUDE dev:<directory>filename.ext [/SEARCH]
```

Where:

dev:	is the name of the file structure that contains the file to be loaded. If dev: is omitted, DSK: is assumed.
filename.ext	is the name of the file to be loaded. The extension is normally .REL. If the filename is omitted, an error message is issued and the job is terminated.
<directory>	is a directory name, which may be different from that specified on the $JOB card. If omitted, the directory name on the $JOB card is used.
/SEARCH	indicates that the file will be loaded in library search mode. This switch is optional.

Restrictions

The file specified on the $INCLUDE card must be a relocatable binary file and must already exist on a mounted disk.

You can name only one file per $INCLUDE card.

$JOB

Function

The $JOB card is the first card in your card deck. The $JOB card, in conjunction with the $PASSWORD card, causes SPRINT to create a control file in your disk area into which commands are placed for the batch controller. It also causes SPRINT to create a log file in the same area of the disk.

TOPS-10 Card Format

```
$JOB name [directory] /switch(es)
```

Where:

name	is the user-assigned name for the job; if omitted, SPRINT creates the unique name JBxxxx (where x represents characters randomly chosen by SPRINT to produce a unique name). SPRINT assigns the jobname to the control and log files and appends the extensions .CTL and .LOG to them, respectively.

[directory]	is your project-programmer number. This argument is required. A space or comma can separate this argument from the job-name.
	
	You may specify a wildcard designation (#) for the programmer number in the $JOB card; for example:
	
	`$JOB FLEX[4,#] or`
	`$JOB FLEX<4,#>`
	
	SPRINT checks to determine if wildcard programmer numbers are permissible for the specified project. If they are, then SPRINT creates a unique programmer number. If they are not allowed, SPRINT issues an error message and the job is terminated.

/switch(es)	are optional switches to batch to tell it the constraints that you have placed on your job. They are described below.

TOPS-20 Card Format

```
$JOB directory/switch(es)
```

Where:

directory	is your assigned user name. This field can contain from 1 to 39 alphanumeric and special characters. This argument is required. The user-name cannot be a files-only directory; it must be a directory that can be logged into.
/switch(es)	are switches from the following group. These switches are optional.

Switches

The following is a brief description of the switches that you can use on the $JOB card. These switches are explained in detail in Appendix B.

Switch	Meaning
/ACCOUNT:name	The job's CPU time will be charged to the account specified.
/AFTER:date-time	The job cannot be run until after the specified date and time. The date and time are specified in the form dd-mm-yy hh:mm (for example, 16-APR-88 17:15). If you omit this switch, batch schedules your job based on the time required and other parameters.
/AFTER:+hh:mm	The job cannot be run until after the time the card deck is read in plus the amount of time specified. The amount of time that the job must wait after it has been entered is specified in the form +hh:mm (for example,+1:30).
/ASSISTANCE:arg	This switch specifies whether the job needs operator assistance. The argument is YES or NO.
/BATCH-LOG:arg (-20) /BATLOG:arg (-10)	The log file is written according to the arguments you specify with this switch: APPEND causes the log file to be appended to an existing log file of the same name; SPOOL causes the file to be written to the system spooling area; SUPERSEDE causes the log file to overwrite another log file of the same name.
/CARDS:nn	The job can punch a maximum of nn cards (up to 10,000 in decimal).
/CORE:nnK (-10) /CORE:nnP (-10)	The job can use a maximum of nnK words (decimal) or nn pages of memory up to the maximum allowed at your installation. The default is usually 32K or 64P.

/DEPEND:nn	This switch specifies the initial interjob dependency count (in decimal) for the job. Refer to Section 3.5, SUBMITTING RELATED JOBS, for details on this switch.
/FEET:nn	The job can punch a maximum of nn (decimal) feet of paper tape.
/JOBNAME:name	This switch assigns a 1- to 6-character name to the job. With TOPS-10, using this switch has the same effect as specifying the job name in the name field of the $JOB card. If you omit this switch, batch creates a new name for your job. The name created by batch is assigned to both your control file and your log file. Batch adds the file type .CTL to the control file and the type .LOG to the log file.
/LOCATE:name/number	This switch specifies the network node to whose line printer the job's output is to be sent. Enter the remote station name or, on TOPS-10, the station number (in octal).
/LOGDISP:arg	This switch specifies the disposition of the LOG file after the batch job has been processed. The arguments are DELETE (default), PRESERVE, and KEEP. Note that KEEP and PRESERVE are identical.
/NAME:name (-10)	This switch specifies the user's name, which can be up to 12 characters. Enclose 'name' in quotes if 'name' contains any blank characters. This switch is optional unless your installation requires the user's name in addition to the project-programmer number and password when you log in.
/OUTPUT:arg	This switch determines whether or not the log file is printed. The arguments are: ALWAYS (print the file), LOG (same as ALWAYS), NOLOG (suppress printing), and ERROR (TOPS-10)/ERRORS (TOPS-20) (print only if an error occurs). ALWAYS or LOG is the default.

/PAGES:nn	The job can print a maximum of nn pages (in decimal). The default limit is 200 pages; however, the system administrator may change this value at batch system generation time. If you need more than 200 pages for your job, you must include the /PAGES switch on the $JOB card to indicate the approximate number of pages that your job will print. If your output exceeds either the maximum that batch allows or the number that you specified in the /PAGES switch, the excess output will not be printed and the message ?LPTPLE PAGE LIMIT EXCEEDED will be written in the log file. However, even if you exceed the maximum, the first 10 pages of the log file will be printed.

NOTE

Do not arbitrarily enter a large PAGES value as this may delay execution of your batch job.

/PPN:[p,pn] (-10)	This switch specifies the TOPS-10 project-programmer number under which the job will run. System-independent jobs require this switch. (Refer to Section 5.4, SYSTEM-INDEPENDENT JOBS.)
/PRIORITY:nn	This switch assigns the job a priority value (external priority) that determines when the job will run relative to other jobs in the specific queue. The values you can specify for nn range from 1 to 20 in decimal. See Appendix B for more information on this switch.
/RESTART	This switch specifies that the job may be restarted after a system failure.
/NORESTART	This switch specifies that the job may not be restarted after a system failure.
/SEQUENCE:nn	This switch allows you to specify an identifying decimal number for the job to distinguish it from others in the input queue. If you omit this switch, the system assigns a "sequence" number to the job. The system also assigns a request id number to jobs. It is recommended that you use the request id number when modifying or killing jobs with duplicate names.

/TIME:hh:mm:ss	This switch sets specific limits upon the amount of CPU time the job can use. Normally, batch allows your job to use up to five minutes of central processor (CPU) time. CPU time is the amount of time that your job runs in memory, not the amount of time that it takes batch to process your job. If you need more than five minutes of CPU time, you must include the /TIME switch on the $JOB card to indicate the approximate amount of time that you will need. If you do not specify enough time, batch terminates your job when the time is up. However, if you specify a large amount of time, batch may hold your job in the queue until it can schedule a large amount of time for it. The value in the /TIME switch is given in the form hh:mm:ss (hours:minutes:seconds). If you specify only one number, batch assumes that you mean minutes. Two numbers separated by a colon (:) are assumed to mean hours and minutes. All three numbers, separated by colons mean hours, minutes, and seconds.
/TPLOT:mm	This switch specifies the maximum amount of plotter time that the job can use (in minutes).
/UNIQUE:arg	This switch specifies how jobs are to be protected from the effects of other batch jobs running in the same directory. An argument of 0 or NO means no protection; 1 or YES (the default) means only one batch job at a time is to be run, and 2 means that the job is to be run in a unique SFD. This last argument is valid only for TOPS-10 systems.
/USER:USERNAME(-20)	This switch specifies the TOPS-20 user directory under which the job will run. System-independent jobs require this switch. (Refer to Section 5.4, SYSTEM-INDEPENDENT JOBS.)

Comment

SPRINT assumes that the $JOB card is punched in the system standard card code (usually ASCII); however, if SPRINT cannot read the card correctly, it will then assume a card code of 026. The default mode for the entire job is based upon one of the two codes that produced a successful read of $JOB.

$LABEL

Function

The $LABEL card causes SPRINT to insert a label in the control file.

Card Format

$LABEL label

Where:

label is a 1- to 6-character label as defined in Section 2.3.

Comment

Control cards such as $BACKTO and $CHKPNT can reference the label supplied on this card.

$language

Function

The $language cards direct the input spooler to compile your programs using the specified language processor.

Characteristics

From the table below, you can see that there is a $language card for many of the language processors and interpreters supported by your operating system. Also, note the particular extensions that these translators expect of the files they process.

Card Name	Default Filename Extension/Type
$ALGOL	.ALG
$BLISS	.BLI
$COBOL	.CBL
$FORTRAN	.FOR
$MACRO	.MAC
$SIMULA	.SIM
$SNOBOL	.SNO

The following description applies to all the $language cards; you need only substitute the name of the language desired (from the above list) immediately after the dollar sign.

The $language card can be used with two different input conditions:

1. The source code immediately follows the $language card and the $language card does not specify a filename. In this case, the $language card directs the input spooler to copy the program onto disk, assign the program a unique filename of the form LN???? with the appropriate filename extension or type (shown in the above table), and insert a COMPILE system command into the control file. The cards that you may use to terminate copying of the program vary according to whether you use the /DOLLARS or the /NODOLLARS switch. (Refer to the brief switch descriptions below, or to Appendix B for a more detailed explanation of these two switches.) The source and object files will be deleted upon logout.

2. The source code of the program is already on disk and the $language card specifies a filename. In this case, the $language card directs the input spooler to insert a COMPILE system command into the control file.

NOTE

A $language card specifying a filename, followed by a deck containing the source code, will cause an error message to be issued.

The $language card does not cause automatic execution of the program after translation of the source code. Execution is initiated by an EXECUTE system command or by a $DATA or a $EXECUTE card. The $SNOBOL card is an exception. It is the only $language card that is not used in conjunction with a $DATA or $EXECUTE card to generate an EXECUTE command. The $SNOBOL card itself initiates both compilation and execution.

TOPS-10 Card Format

If the source deck follows the $language card, the format of the card is:

```
$language dev:[directory] (processor switches)/switch(es)
```

If the source code already exists on disk, the format of the card is:

```
                                                      ---      ---
                                                      ||/LIST    |
$language dev:filename.ext[directory](processor switches) ||/NOLIST  |
                                                      ---      ---
```

Where:

dev:	is the name of the file structure that contains the program to be compiled if the program is already on disk. Otherwise, this is the name of the file structure onto which the program is copied. If dev: is omitted, DSK: is assumed.
filename.ext	is the name of the file to be compiled. The filename is specified only when the file is already on disk. If a filename is specified and there is a card deck to follow the $language card, an error message will be issued.
[directory]	is a project-programmer number, which may be different from that specified on the $JOB card. Directory specifications may also be enclosed in angle brackets (< >).
(processor switches)	are the switches to be passed to the compiler. (Refer to appropriate language manual and to the COMPILE command for a description of the processor switches). They must be enclosed in parentheses and the slash (/) must not appear in connection with these switches.
/switch(es)	are the switches that control the mode of input interpretation and the listing of the compiled program. The only switch available when the program is already on disk is either the /LIST or the /NOLIST switch.

TOPS-20 Card Format

If the source deck follows the $language card the format of the card is:

```
$language/switch(es)
```

If the source code is available from another device and if only compilation is desired, the format of the card is:

```
                                   ---      ---
$language dev:<directory>name.typ  ||/LIST    |
                                   ||/NOLIST  |
                                   ---      ---
```

Where:

dev:	is the name of the device that contains the program to be compiled. If dev: is omitted, DSK: is assumed.
name.typ	is the name of the file to be compiled. Specify the name only when the file already exists. If you specify a filename when a card deck follows the $language card, the system issues an error message.
<directory>	is a directory name that may be different from that specified on the $JOB card.
/switch(es)	are the switches that control the mode of input interpretation and the listing of the compiled program. The only switch available when the program is already on disk is either the /LIST or the /NOLIST switch.

Switches

The following is a brief description of the switches that you can use with the $language cards. These switches are explained in detail in Appendix B. Each of the $language cards can contain any of the following switches, except in the case of the /CREF switch which can be used only with the $FORTRAN and $MACRO cards.

Switch	Meaning
/026	The card deck is read as 026 card code.
/ASCII	The card deck is read as ASCII card code.
/BCD	This switch has exactly the same effect as the /026 switch; that is, the card deck is read as 026 card code.
/CREF	A cross-referenced listing is created by the CREF program. This switch is available only on the $FORTRAN and $MACRO cards.

/DOLLARS	Only $JOB, $EOD, and $EOJ are recognized as control cards. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or as user data.
/LIST	A compilation listing will be generated. /LIST is the default.
/NOLIST	Normally, the $language card tells batch to ask the compiler to generate a compilation listing of your source program. The listing is then printed as part of your job's output. If you do not want this listing, you can include the /NOLIST switch on the $language card to stop generation of the listing.
/SUPPRESS	When batch reads the cards containing your source program, it normally copies everything on the card up to column 80 or any column you may specify in the /WIDTH switch. However, if you do not want trailing spaces copied (to save space on the disk, for example), you can tell batch, by means of the /SUPPRESS switch, not to copy any trailing spaces into the disk file.
/NOSUPPRESS	Trailing blanks are not suppressed. /NOSUPPRESS is the default.
/WIDTH:nn	Only columns 1 through nn (inclusive) of each card are read. Specify nn as a decimal number.

$MACRO

See the $language card description.

$MESSAGE

Function

The $MESSAGE card causes the supplied text to be output to the system operator's terminal at the time the job is run.

Card Format

$MESSAGE/switch text

Where:

/switch	is a switch indicating whether operator response is required. The switch is optional but, if present, must immediately follow the command ($MESSAGE). Otherwise, an error message will be issued and the job will be terminated.
text	is the message to be output to the system operator.

Switches

Switch	Meaning
/WAIT	The job will wait for a response from the operator before resuming its processing.
/NOWAIT	The job will continue after typing the message without waiting for a response from the operator. This is the default.

$OPERATOR

$NOOPERATOR

See descriptions for the corresponding batch commands (OPERATOR, NOOPERATOR) in Section 2.6. Refer also to NOTE at beginning of this section.

$PASSWORD

Function

The $PASSWORD card helps ensure that an unauthorized person does not have access to the system.

Characteristics

This card contains the password associated with the project-programmer number specified on the $JOB card. If the password does not match the password stored in the system for the specified project-programmer number, SPRINT does not create any files, issues an error message to the log file, and terminates the job.

NOTE

The $PASSWORD card is not required if you are submitting a card job from the terminal and the directory you specified on the $JOB card matches the directory under which you are logged in. (Refer to Section 4.1.1 for details.)

Card Format

$PASSWORD password

Where:

password	is a 1- to 39-character password

NOTE

There must be exactly one space between the end of the card name ($PASSWORD) and the first character of the password.

Requirements

If you use the $PASSWORD card, it must immediately follow the $JOB card.

$RELOCATABLE (-10)

Function

$RELOCATABLE is a File Card that causes SPRINT to copy a relocatable binary program from cards to a file in your disk area. The cards are read in binary mode.

Characteristics

Files created by a $RELOCATABLE card are added to the list of files maintained by SPRINT to be included in the EXECUTE command generated by a $DATA or $EXECUTE card.

Card Format

$RELOCATABLE filename.ext

Where:

filename.ext	is the name of the file into which the binary data is copied. If you omit the filename, SPRINT creates the filename RL???? (where ? represents a character arbitrarily chosen by SPRINT to produce a unique filename). If you omit the extension, .REL is assumed.

Restrictions

Relocatable binary programs can be read only when the input is from cards.

Related Card

$CREATE/BINARY

The $CREATE card with the /BINARY switch is similar to the $RELOCATABLE card in that it causes SPRINT to copy the binary card deck that follows it to a disk file. However, there is an important difference between the two cards. All files created by a $RELOCATABLE card are added to the list of files maintained by SPRINT to be included in the EXECUTE command generated by a $DATA or $EXECUTE card. A file created by $CREATE with the /BINARY switch is not included unless there is a $INCLUDE card for the file.

$REQUEUE

$REVIVE

See descriptions for the corresponding batch commands (REQUEUE, REVIVE) in Section 2.6. Refer also to NOTE at the beginning of this section.

$SEQUENCE

Function

The $SEQUENCE card specifies a unique number (sequence number) for the job that helps distinguish it from other jobs within the queues. If you omit this card, the system assigns a sequence number to the job. This number is useful when modifying or killing jobs to which you have assigned duplicate names. (Give the number as an argument to the /SEQUENCE switch, and include this switch on the modify or kill command line. Refer to Chapter 3, JOB CONTROL, for details on modify and kill commands.)

NOTE

You may place the $SEQUENCE card anywhere in the deck between the $PASSWORD and $EOJ cards. **It is not allowed in the first card position.**

Also, you cannot use this card to terminate decks to be copied to disk with the /DOLLARS switch in effect.

Card Format

```
$SEQUENCE n
```

Where: n is a decimal number

Related Items

1. Request ID Number

   The system also assigns a unique request id number to every job you submit to a queue. It is recommended that you use the request id number rather than the sequence number when modifying or killing jobs with duplicate names.

2. $JOB/SEQUENCE:nn

   The $JOB card with the /SEQUENCE switch performs the same function as the $SEQUENCE card.

$SILENCE

See the SILENCE batch command description in Section 2.6. Refer also to NOTE at the beginning of this section.

$SIMULA

$SNOBOL

See the $language card description.

$TOPS

$TOPS10

$TOPS20

Function

The $TOPS, $TOPS10, and $TOPS20 cards direct the input spooler to copy all cards following them into the batch control file. The copying process is terminated by the next control card in the deck.

Characteristics

The cards following a $TOPS, $TOPS10, or $TOPS20 card are expected to contain batch, system, user, or system program commands. These cards will not be processed by SPRINT. The cards that you may use to terminate the copying of these commands vary according to whether you specify the /DOLLARS or the /NODOLLARS switch. Refer to the brief switch descriptions below, or to Appendix B, for a more detailed explanation of these two switches.

As discussed in Section 5.4, SYSTEM-INDEPENDENT JOBS, all three of these cards are valid for systems running TOPS-10 and TOPS-20. However, the commands associated with $TOPS10 and $TOPS20 are copied to the batch control file on a system dependent basis. For instance, on a system running TOPS-20, the cards following $TOPS20 are copied to the file. On a system running TOPS-10 the cards following $TOPS-20 are ignored.

The $TOPS card is a system-independent version of $TOPS10 and $TOPS20; that is, the cards following $TOPS are copied to the batch control file whether or not you are on a TOPS-10 or TOPS-20 system. Use $TOPS for commands in a system-independent job that you want to include for both TOPS-10 and TOPS-20. For example, instead of writing your card file as:

```
$TOPS10
.DAYTIME
.SYSTAT
.DIRECT
$EOD
$TOPS20
@DAYTIME
@SYSTAT
@DIRECT
$EOD
```

Write:

```
$TOPS
DAYTIME
SYSTAT
DIRECT
$EOD
```

In this way, you can use one set of commands to accomplish the same tasks on both systems.

Section 5.4 further discusses these three commands.

NOTE

A single system or batch command or a group of consecutive system and/or batch commands must be preceded by a $TOPS10 or $TOPS20 card and followed by the appropriate terminating card. If such commands are placed into the SPRINT input stream but are not properly delimited, an error message will be issued.

Card Format

```
$TOPS      /switch(es)
or
$TOPS10
or
$TOPS20
```

/switch(es)	Switches that control the modes in which the input is read and interpreted.

Switches

The following is a brief description of the switches that you can use on the $TOPS, $TOPS10, or $TOPS20 card. These switches are explained in detail in Appendix B.

Switch	Meaning
/026	The card deck is read as 026 card code.
/ASCII	The card deck is read as ASCII card code.
/BCD	This switch has exactly the same effect as the /026 switch; that is, the card deck is read as 026 card code.
/DOLLARS	Only $JOB, $EOD, and $EOJ are recognized as control cards. All other cards with a dollar sign in column 1 are treated as user data.
/NODOLLARS	If a card has a dollar sign in column 1, the contents of column 2 determine whether it is treated as a SPRINT control card or user data.
/SUPPRESS	Trailing blanks are suppressed.
/NOSUPPRESS	Trailing blanks are not suppressed.
/WIDTH:nn	Only columns 1 through nn (inclusive) of each card are read. Specify nn as a decimal number.

5.4 SYSTEM-INDEPENDENT JOBS

SPRINT can process certain card jobs under either the TOPS-10 or the TOPS-20 operating system. Such jobs are system independent.

You might want to create a system-independent job if you have access to both a TOPS-10 and a TOPS-20 system and you run a job on one system that is very similar to a job you run on the other system.

Such jobs usually vary only to compensate for operating system differences and are likely to exist where a particular application runs on both systems. For example, the collection of system accounting information is the kind of application that could be common to both systems. In these cases, SPRINT's system-independent capabilities allow you to coordinate the two jobs accomplishing the tasks for the application. That is, all the SPRINT and batch commands for both jobs can reside in one file, with those commands pertaining to a specific operating system being executed only when the job is run under that system.

The only system commands that you cannot use in a batch job are ATTACH and SET TIME-LIMIT. Batch sends these commands to the EXEC, the EXEC issues an error, and batch detects the error and terminates your job. Also, you cannot use the LOGIN command in your batch job because you will get an error that will terminate your job. Batch logs in your job in accordance with your $JOB and $PASSWORD cards.

5.4.1 Creating a System-Independent Job

You create a system-independent job much like you create any other card job. The major difference lies in the $JOB card: with system-independent jobs you must specify both the /USER: and /PPN: switches. These switches are to immediately follow $JOB on the $JOB card as follows:

```
     $JOB /USER:TUCKER/PPN:[27,13]/JOBNAME:SAMPLE
```

Observe that the /USER: switch replaces the user-name parameter you are accustomed to putting on the $JOB card for TOPS-20 system-dependent jobs. /PPN: replaces the $JOB [proj,prog] parameter you use for TOPS-10 system-dependent jobs. Also regarding TOPS-10, the $JOB "name" parameter, valid on a system-dependent job, is inappropriate for a system-independent job. Use the /JOBNAME: switch to name a system-independent job as shown above.

The password appearing on the $PASSWORD card must equal the password corresponding to both your user name under TOPS-20 and your project-programmer number under TOPS-10. This means that the passwords on the two systems must be identical.

Lastly, include in the one job both the $TOPS10 and $TOPS20 cards along with all their subsequent data cards. Or you can use $TOPS.

The following shows a sample system-independent job.

```
$JOB /USER:TEST/PPN:[27,1531]/JOBNAME:TOPSXX
$PASSWORD TEN20
$!
$! This control file can be run with either TOPS-10 or TOPS-20.
$! The file lists itself then executes several monitor commands: it
$! performs the COPY, SYSTAT, and DIRECT commands.
$! All SPRINT commands are executed with both operating systems.
$!
$NOERROR
$SILENCE
$TOPS20         !!!!!!!Start of TOPS-20 commands!!!!!!!
$!Executed under TOPS-20 only:
@COPY TOPSXX.CTL (TO) LPT:
@SYSTAT LPT
@DIRECT ,
@LPT
@
!End of TOPS-20 commands.
$EOD
$!
$TOPS10         !!!!!!!Start of TOPS-10 commands!!!!!!!
!executed under TOPS-10 only:
.COPY LPT:=TOPSXX.CTL
.SYSTAT/L
.DIRECT/L
!End of TOPS-10 commands.
$EOD
$EOJ
```

5.4.2 System-Independent Processing

As with other card jobs, SPRINT converts system-independent card jobs to batch control files.

SPRINT determines that a card job is system-independent if the first non-blank character after $JOB on the $JOB card is a slash (/). In processing system-independent jobs, SPRINT conditionally translates the cards following a $TOPS10 or $TOPS20 card on a system-dependent basis. That is, the cards between a $TOPS10 card and the terminator are translated on a TOPS-10 system; these cards are ignored on a TOPS-20 system. Similarly, cards between the $TOPS20 card and the terminator are translated only on a TOPS-20 system. By this action, any command preceded by the system prompt character is read by the appropriate system. Recall that the cards following a $TOPS10 or $TOPS20 card contain such commands.

Cards following the $TOPS card are translated on both a TOPS-10 and a TOPS-20 system.

The SPRINT commands are the same for both operating systems, thus requiring no special processing. $IF, however, although the same for both systems, is treated uniquely. The following section discusses $IF.

SPRINT chooses the directory for a system-independent job according to the /USER: and /PPN: switches you specify on the $JOB card. When the job is run under TOPS-20, SPRINT refers to the /USER: switch to determine the directory; with TOPS-10, SPRINT refers to the /PPN: switch.

5.4.2.1 $IF and System-Independent Processing

In processing a job, SPRINT examines the first character after the right parenthesis of the (condition) parameter on the $IF card. If this character is a dollar sign ($), denoting a SPRINT command as the statement parameter, SPRINT translates the dollar sign to the appropriate system prompt character, creating an executable batch command from the SPRINT command. (Recall from the NOTE at the beginning of Section 5.3.7, Control Card Descriptions, that many SPRINT commands are convertible to batch commands. In fact, all of the batch commands have their equivalents among these convertible SPRINT commands.)

This method of handling the $IF card allows you to test for error and nonerror conditions and to act upon those conditions on a system-independent basis. You can do this in those instances where you would specify a batch command for the statement parameter. For example, the card:

```
     $IF (ERROR) $GOTO END
```

causes processing to resume at the END label, whether an error occurs on a TOPS-10 system or a TOPS-20 system.

Use the IF batch command instead of $IF in those cases that require a system or user program command for the statement parameter. Also, use the IF command if you do not wish a particular IF statement to apply for runs of the job under both operating systems.

5.5 SAMPLE JOBS

The following sample jobs illustrate the versatility of the batch system. They show how to compile and run programs, list card decks on the printer, and so forth.

Unless specified, the examples apply to both TOPS-10 and TOPS-20. You may have to change the monitor prompt character or a system command to make an example fit your operating system. More important, however, the sequence of card types you need to perform various system functions is identical for the two systems. The following examples are intended to provide these sequences.

5.5.1 Listing Cards on a Line Printer

METHOD 1

The following example shows the cards you need to list cards on a line printer using method 1.

The $CREATE card creates a disk file in your directory with the specified name and filename extension/type. All following data cards are read into the file and the file is submitted to the print queue. The .LST extension/type causes the file to be deleted after it has been printed.

The $EOD card is optional.

```
$JOB
$PASSWORD
$CREATE name.LST/PRINT
data cards
$EOD
$EOJ
```

The extension .LST causes the file to be deleted after printing.

METHOD 2

This example shows the cards you need to list cards on a line printer using method 2.

The $TOPS-20 card causes the batch controller to interpret all cards up to the next control card as system commands.

The @PRINT card submits the previously created disk file to the print queue. The file is deleted after it is printed.

```
$JOB
$PASSWORD
$CREATE filename
data cards
$EOD
$TOPS20
@PRINT filename/DELETE
$EOJ
```

5.5.2 Producing a MACRO CREF Listing

The following example shows how to produce a CREF listing of a MACRO deck whether or not errors occur in the program.

```
$JOB
$PASSWORD
$MACRO/CREF
MACRO deck
     .
     .
     .
$EOJ
```

5.5.3 Using Error Processing Commands

The following example shows how to use the IF batch command to create a CREF listing if no error occurs during a MACRO compilation.

```
$JOB
$PASSWORD
$CREATE TRIAL1.MAC
MACRO deck
    .
    .
    .
$MACRO TRIAL1.MAC/CREF
$TOPS10
.IF (NOERROR) .GOTO B
.GOTO A
B::.CREF
A::!HERE TO SKIP CREF
$EOJ
```

5.5.4 Creating a Complex Card Job

The following example illustrates a MACRO assembly, two FORTRAN compilations, the loading and execution of all three programs as a single module, and shows how to enter monitor commands along with the programs and the SPRINT control cards.

```
$JOB
$PASSWORD
$MACRO/CREF
MACRO source program
    .
    .
    .
$FORTRAN
FORTRAN source program
    .
    .
    .
$FORTRAN
FORTRAN source program
    .
    .
    .
$EXECUTE
$TOPS10
.R BASIC
*NEW
BASIC source program
    .
    .
    .
*RUN
*MONITOR
$EOJ
```

5.5.5 Loading a FORTRAN Program with a System Library

The following example shows how to load a FORTRAN program with a system library. The $FORTRAN card causes the FORTRAN program to be compiled. The $DATA card generates an EXECUTE system command and both the FORTRAN program and the relevant subroutines from SYS:SSP.REL will be included in the EXECUTE command string.

```
$JOB
$PASSWORD
$FORTRAN
FORTRAN source program
     .
     .
     .
$INCLUDE SYS:SSP.REL/SEARCH
$DATA
data deck
     .
     .
     .
$EOJ
```

5.5.6 Reading from or Writing to a Disk File

The following example shows the cards you need to compile and run a program that reads from or writes to a disk file. $FORTRAN, a language card, causes the program to be assembled by the appropriate compiler or assembler. The FORTRAN source program provides the disk read/write instructions, and the $EXECUTE card causes the program to be executed.

Your program should provide the necessary disk read/write statements.

```
$JOB
$PASSWORD
$FORTRAN
FORTRAN source program
     .
     .
     .
$EXECUTE
$EOJ
```

For other languages, use $COBOL, $MACRO, $ALGOL,...

5.5.7 Reading from or Writing to a Tape File

The following example shows the cards you need to compile and run a program that reads from or writes to a tape file. Two cards introduced in this section are explained as follows:

MOUNT command	Causes a tape mount directive to be sent to the system operator. The batch job will not be processed beyond this command until the operator has mounted the appropriate tape. If you use this command, be sure to give your tape to the operator before you submit your job.

DISMOUNT command	Rewinds the magnetic tape onto one reel and deassigns the tape drive from your job. The tape drive is now available for use by other jobs.

Your program should provide the necessary tape read/write statements.

```
$JOB
$PASSWORD
$TOPS20
@MOUNT
$FORTRAN
FORTRAN source program
     .
     .
     .
$EXECUTE
$TOPS20
@DISMOUNT
$EOJ
```

For other languages, use $COBOL, $MACRO, $ALGOL,...

5.5.8 Compiling and Running a Program that Reads Data from Cards

The following example shows the cards you need to compile and run a program that reads data from cards. The $DATA card marks the beginning of the data and contains an implicit EXECUTE command. Thus, you do not have to specify a separate $EXECUTE card if you use the $DATA card.

Your program should provide the necessary read statements. For example, the following two FORTRAN statements would cause the system to read data from a card reader:

```
OPEN (UNIT=2,ACCESS='SEQIN')
READ (2,100) Variable list
```

The $EOD card is optional.

```
$JOB
$PASSWORD
$FORTRAN
FORTRAN source program
     .
     .
     .
$DATA
data for program
     .
     .
     .
$EOD
$EOJ
```

For other languages, use $COBOL, $MACRO, $ALGOL...

5.5.9 Using an Interpretive Language (APL, BASIC, CPL, or SNOBOL)

The following example shows the cards you need to use an interpretive language.

A card introduced in this section, MONITOR, provides the statement that returns control from the interpreter to the system. The format for this statement changes according to the language you use as follows:

Statement	Language
MONITOR	BASIC, CPL
)MONITOR	APL
System prompt character (use without an asterisk)	SNOBOL

```
$JOB
$PASSWORD
$TOPS20
@CPL
CPL statements
     .
     .
     .
*MONITOR
$EOJ
```

For other languages, use APL, BASIC, SNOBOL.

5.5.10 Running a System Program or Utility

The following example shows the cards you need to run a system program or utility. The @SORT card is a name card that can contain the name of any utility or system program you wish to run.

NOTE

> If a compiled and linked program (one with a .EXE filename extension/type) is stored in your disk area, you may run it by specifying:
>
> ```
> .
> @ RUN name
> ```

The command string card contains additional commands (if required) to the program. Precede program commands with an asterisk (*).

NOTE

TOPS-20:

A TOPS-20 system command (one of the commands discussed in the Commands Reference Manual, such as COPY) differs from a system program or utility in that you must precede input lines for a command with an at sign (@) rather than an asterisk. (See Section 2.2, LINE IDENTIFIERS, for details.)

```
$JOB
$PASSWORD
$TOPS20
@SORT
*command string
$EOJ
```

5.5.11 Saving Cards in a Disk File

The following example shows the cards you need to save cards in a disk file. The $CREATE card creates a disk file with the specified filename and stores all following data cards in the file. The file is created in your disk area.

The $EOD card is optional.

```
$JOB
$PASSWORD
$CREATE filename
data cards
    .
    .
    .
$EOD
$EOJ
```

5.5.12 Suppressing Printing of the Log File

If you have access to a timesharing terminal, you may desire (as a papersaving measure) to suppress printing of the log file and to save the log file in your disk area. You can then examine and delete it. You can accomplish this by placing the following switches on the $JOB card:

```
$JOB name/OUTPUT:NOLOG/LOGDISP:PRESERVE
```

The /OUTPUT:NOLOG switch suppresses printing of the log file. The /LOGDISP:PRESERVE switch saves the log file in your disk area under name.LOG where 'name' is the job name (not the user name).

Alternatively, you may wish to have the log file printed only if an error occurs during execution of the job. You can do this by placing the following switch on the $JOB card:

```
$JOB name/OUTPUT:ERRORS
```

5.5.13 "Stacking" Card Decks

Assign unique names to card decks that are "stacked" then later accessed during batch processing. (To "stack" a deck is to write its contents on the disk.) Otherwise, each succeeding card deck will overlay the preceding one before the system executes any batch or system commands; that is before the system even begins processing the control file that SPRINT created from your job deck.

Recall that all card decks and commands in the job deck are sent to separate disk files and that commands in the control file, one of the newly-created disk files, not the sequence of commands in the original job deck, govern the batch system. Thus, as in the following example, files with duplicate names are overwritten as they are sent from the card reader to disk, not as the batch system executes your job.

```
$JOB/PPN:[27,5344]/USER:SAMPLE
$PASSWORD
$CREATE DATA.TXT
     .
     .
  (card deck)
     .
     .
     .
$EOD
$TOPS
TYPE DATA.TXT
$CREATE DATA.TXT
     .
     .
  (card deck)
     .
     .
     .
$EOD
$TOPS
TYPE DATA.TXT
$EOJ
```

The job deck in this example will cause the second DATA.TXT file to overlay the first upon being "stacked" and will produce the following control file (commands are preceded by the appropriate system prompt character):

```
TYPE DATA.TXT
TYPE DATA.TXT
```

The second data file is typed twice.

CHAPTER 6

BATCH SYSTEM OUTPUT

The output from a batch job is normally in the form of printed listings containing:

1. The job's log file.

2. Printer output you requested through commands in the control file, such as compilation listings and program output. With card jobs, compilation listings are produced from the $language control card unless you specify otherwise. These listings are automatically printed. You can also include the COMPILE system command in your control file with switches to produce listings.

6.1 THE LOG FILE

NOTE

TOPS-20

As of version 5 of the EXEC, system messages are sent to your log file if a batch job is run for you between the time the message is sent and the time you log onto your terminal. These messages are not typed on the terminal as they usually are when you log in.

Like the terminal output that results from a timesharing session, the batch system transaction log shows your commands (from the control file) and the system's responses to those commands. Specifically, into the log file go control file lines as they are passed to the job; BATCON, SPRINT, and system messages; and the details of most operator actions affecting the job. In addition, any output destined for your terminal (through your use of the TYPE system command, for example) goes to the log file.

With switches available for the $JOB card and for the SUBMIT command, you can control:

1. How the log file is created.

   The /BATLOG (-10) and /BATCH-LOG (-20) switches give you control in this area. With these switches, you can append the log file to an existing one of the same name (default), replace an existing log file of the same name, or write the log file to the system spooling area (from where it is printed).

2. Whether the log file is printed.

   The /OUTPUT switch gives you the options to print the log file under all circumstances, only when errors occur during processing, or not at all.

3. What the log file's disposition will be upon completion of the job.

   With card jobs, you can use the /LOGDISP switch with $JOB to specify whether the log file is to be removed from or saved in your disk area.

   Likewise, when submitting a job from the terminal under TOPS-20, using the /LOGDISP switch (with the SUBMIT command) provides you with the same two options.

   When submitting a job from the terminal under TOPS-10, the /DELETE switch removes the log file from your disk area. The /PRESERVE switch (default) saves the log file.

The following sections further describe the log file.

6.1.1 BATCON Log-File Output

The contents of the log file appear exactly as they would have if you had been running at a timesharing terminal, with the following exception: each line begins with a time stamp to indicate the time at which the line was processed and a notation to indicate the type of the line.

The time stamp has the format hh:mm:ss to indicate the time of day on a 24-hour clock. The notation is next, followed by the line of input or output. The notation is one of the following:

BATCH	Indicates a batch command from the control file or a command issued by BATCON.
CANCEL	Indicates that the job was canceled at the user's request.
COMMENT	Indicates a comment line from the control file.
DUMP	Identifies output lines resulting from execution of the DUMP batch command.
FALSE	Indicates that the condition tested in the IF command was false. The statement parameter is listed as a comment on the same line with the IF command.
HEADER	Identifies a command line in the header section of the step or indicates the number of lines processed in the step.
IGNORE	Identifies lines that were skipped when BATCON was searching for the IF command or for one of the reserved labels. This notation also identifies blank lines in the control file.

LABEL	Identifies a label found in the control file. The rest of the control file line is listed in the log file on a separate line.
MONITR	Indicates that the output line was from the monitor or that the input line was to be processed by the monitor as a system command.
OPERAT	Identifies a line entered by the operator. Usually, the specific operator command or comment appears in the log file.
TRUE	Indicates that the condition tested in the IF command was true. The statement parameter is listed in the log file on a separate line.
USER	Indicates that the output was sent from a job at user (program) level or that the input was to be sent to a running program.

Any lines in the log file that do not appear as described above are error messages from the batch controller or comments. (Refer to Section 6.2, FATAL ERROR MESSAGES, WARNINGS, COMMENTS for more information on batch system output.)

6.1.2 SPRINT Log-File Output

If you submitted your job on cards, the log file also lists the control cards and contains a record of SPRINT's processing, of the errors SPRINT detected, and of operator intervention that occurred during SPRINT's operations. SPRINT output precedes the information entered into the log file by BATCON; BATCON writes to the log file only when the job is run (after SPRINT has processed the card deck).

SPRINT follows the same format used by BATCON for log file lines. The notation portion of a SPRINT line is taken from the following group:

STDAT	Gives the system name and SPRINT's version number.
STMSG	Identifies any SPRINT non-error message.
STERR	Identifies any SPRINT error message.
STCRD	Identifies SPRINT control cards.
STSUM	Identifies the summary message at the end of the job.
STOPR	Identifies any operator actions that occurred during SPRINT's processing.

The first entry in the log file for a card job always contains the identifier STDAT and a message giving the current version of SPRINT and the system name, for example:

```
14:54:15 STDAT  SPRINT VERSION 104(4107)    RZ123A KL #1026/1042
```

Each control card is written into the log on an STCRD line. These lines, beginning with the $JOB card, follow the STDAT entry. For security reasons, the password you specify on the $PASSWORD card is never written. If one of the control cards is a File Card, the line immediately following contains a message of the form:

TOPS-10:

```
14:17:40 STMSG file.ext Created - nn Cards read - nn Blocks
written
```

TOPS-20:

```
14:17:40 STMSG file.typ Created - nn Cards read
```

When the $EOJ card is read, SPRINT prints a summary message indicating the status of the job (whether it was completed or aborted) and giving the number of cards read, the number of files and blocks written, and the number of each type of error that occurred. The summary is also placed in the system accounting file. Two examples of job summaries are given below.

```
1.  14:17:42  STSUM End of Job Encountered
    14:17:42  STSUM 423 Cards Read
    14:17:42  STSUM 4 Hollerith Errors
    14:17:43  STSUM Batch Input Request Created

2.  16:18:22  STSUM Job Aborted due to Fatal Error
    16:18:22  STSUM 10 Cards Read
    16:18:23  STSUM 10 Hollerith Errors
```

Refer to Section 6.2.4, SPRINT Messages, for additional information on SPRINT output.

6.2 FATAL ERROR MESSAGES, WARNINGS, COMMENTS

The following conventions are used in describing the batch system messages that appear in your log files:

$cardname	Control card name
dev	Device name
file.ext	Filename and extension (-10)
file.typ	Filename and file type (-20)
n	Decimal number
p,pn	Project-programmer number (-10)
x	Alphabetic character
/switch	Switch name

Most messages fall into one of three categories. These categories are determined by the beginning character of the message.

? at the start of the message indicates a fatal error message. No further processing will be done and the job will be terminated. For example, an unrecognizable control card (with a card job) would generate a fatal error message.

% at the start of the message represents an advisory or warning message. Processing will continue in this case unless you specified otherwise with the ERROR batch command. For example, an unrecognizable switch that can be ignored would generate a warning message.

[at the start of the message indicates a comment line. It is for your information only and does not affect the running of the job.

In addition, each message issued has a six-letter prefix. The first three letters indicate the program that issued the message (for example, BAT for BATCON). The last three letters represent an abbreviation of the text of the message. For example:

```
%SPTNAU NO AUXACC ENTRY
```

Error messages generated by a batch job will appear in your log file. Error messages generated when you submit invalid commands to the system via a terminal will be displayed at the terminal.

6.2.1 BATCON Messages

BATATX [Time limit exceeded; allowing HH:MM:SS extra time]

The job used up its alloted time (as determined by the /TIME switch or the system default) and has been granted the specified grace period.

BATBJC ? Job has been canceled - LOGIN failure

The system would not allow this job to log in. The reason is specified in the output from the LOGIN program.

BATBJC [Batch job has been canceled]

The reason the job was canceled is given in an accompanying message.

BATBLA [Beginning processing at label <label name>]

The job is starting at the indicated label. This is an advisory message that occurs either (1) after a system reload when you have specified a CHKPNT label and the CHKPNT has been taken, or (2) at the beginning of the job in accordance with the /TAG switch.

BATBLI [Beginning processing at line <line number>]

The job is starting at the line specified with the /BEGIN switch.

BATCFE ? Control file error for <control file spec> - <error text>

This message occurs for a variety of I/O errors. The reason is given in a self-explanatory message in the format above. The specification for the file involved is given with the message.

BATCST ? Cannot set time limit; job canceled

The system is unable to grant your job extra time because of some system error.

BATECF ? End of control file while searching for label <label name>

The end of the control file was reached while the batch controller was searching for %xxxx, where xxxx is one of the error recovery labels (CERR, ERR, TERR, or FIN).

Or, the end of the control file was reached while the batch controller was searching for the label specified in a GOTO or BACKTO command. A common cause of a GOTO error is:

```
GO TO label
```

The space between GO and TO causes the system to take GO as the GOTO command and TO as its label. GO is a valid abbreviation for GOTO.

Another example of this error is:

```
BACKTO ABCD

     .

     .

     .

ABCD::  command
```

The label specified in a BACKTO command must precede the command in the control file.

BATECF ? End of control file while searching for line <line number>

The value specified for the /BEGIN switch exceeded the number of lines in the control file.

BATEOF ? End of file during step header processing

The $ENDHDR command is missing at the end of the step header for the job.

BATEPL ? BACKTO command has entered a possible loop

You may have created a loop in your control file through improper use of the BACKTO command. For example, A:: BACKTO A is a loop and would generate this message.

BATFFS [Found %FIN while searching for <label name> - proceeding from %FIN]

A %FIN was encountered during a search initiated by a GOTO command or an implicit GOTO (searches for %CERR, %ERR, or %TERR). %FIN takes precedence during searches for reserved labels or for a label specified in a GOTO command.

BATICS ? Illegal character specified for <command name> command

You specified a character that is an illegal signal character for either the ERROR or OPERATOR command. The signal character for these commands must not be a control character, an exclamation point (!), or a semicolon (;)..

BATIIC ? Illegal IF command argument or syntax error

The operand of the IF command could not be recognized, or the format of the IF command is invalid.

BATILL % Incomplete last line in control file

A carriage-return/line-feed is missing at the end of the control file.

BATISL ? Illegal $STEP label

The $STEP label must conform to the standards established for all other unreserved labels. Refer to Section 2.3, LABELS.

BATJNR ? Job canceled after a restart - it is not restartable

The job has a restart parameter of NO, and it has been reselected after a system reload. In accordance with your /RESTART specification, the job is being canceled.

```
BATJRQ  [Job requeued by  BATCON ]
                          user
```

Your batch job has been requeued. Other messages accompany this message to indicate the reason.

BATMNS [MDA not supported - line ignored]

The Mountable Device Allocation software (MDA) is not installed on your system; however, one of the commands or switches on your command line requires MDA. Therefore, the batch controller bypasses this control file line.

BATMOS ? More than one job step encountered - job canceled

The current version of this software allows only one job step per control file.

BATMSL ? Missing $STEP label

A $STEP label is required if any step header commands are present (for example, $MOUNT).

BATNBC % Job nnn no longer under BATCON's control; job canceled

Another user or the operator is attached to the job and now controls it.

BATNLS ? No label specified or illegal syntax

The label was not specified for a command that requires one, or a punctuation error was made.

BATOIN ? Operator intervention not allowed in this stream - job canceled

The /ASSISTANCE switch indicates that no operator assistance is needed for the job. However, a command in your control file (for example, MOUNT) calls for operator intervention.

BATSSE ? Step header syntax error - <error text>

You did not follow rules for entering commands in the step header section of the control file. The particular violation is given in a self-explanatory message in the format above.

BATTLE ? Time limit exceeded after allowing extra time; job canceled

The grace period has expired. Refer to error message BATATX for details.

BATTLF [A temporary log file <file name> will be created, printed, and deleted]

This error is most likely to occur if you do not have write privileges for this directory.

6.2.2 LPTSPL Messages

```
LPEND   SUMMARY:  xxx PAGES OF OUTPUT
                  yyy DISK BLOCKS READ
```

Where:

xxx is the number of pages printed.
yyy is the number of disk blocks processed.

LPERR ?CAN'T ACCESS FILE xxx, yyy

Where:

xxx is the file to be printed.
yyy is the reason it cannot be printed.

LPERR ?ERROR READING INPUT FILE - xxx

Where:

xxx is the reason the file cannot be read.

LPERR ?INVALID ACCOUNT STRING SPECIFIED (xxx)

Where:

xxx is the specified account string.

LPERR ?LPT I/O ERROR OCCURRED DURING xxx, COPY:yyy,PAGE:zzz;STATUS IS:foo

Where:

xxx is the file being printed.
yyy is the copy of the file.
zzz is the page of the file.
foo stands for the octal printer status bits.

LPERR ?PAGE LIMIT EXCEEDED

LPMSG BACKSPACED TO BEGINNING OF xxx

Where:

xxx is the file to which you BACKSPACEd.

LPMSG FILE xxx BACKSPACED yyy COPIES

Where:

xxx is the file being BACKSPACEd.
yyy is the number of copies being BACKSPACEd.

LPMSG FILE xxx BACKSPACED yyy PAGES

Where:

xxx is the file being BACKSPACEd.
yyy is the number of pages being BACKSPACEd.

LPMSG FILE xxx FORWARD SPACED yyy COPIES

Where:

xxx is the file being forward spaced.
yyy is the number of copies being skipped.

LPMSG FILE xxx FORWARD SPACED yyy PAGES

Where:

xxx is the file being forward spaced.
yyy is the number of pages being skipped.

LPMSG FILE xxx SKIPPED BY OPERATOR

Where:

xxx is the file being skipped.

LPMSG FINISHED FILE xxx

Where:

xxx is the file that was printed.

LPMSG JOB BEING RESTARTED AFTER xxx

Where:

xxx represents 'system failure' or 'requeue by operator'.

LPMSG JOB CANCELED BY USER xxx

Where:

xxx represents the user who cancelled the job.

LPMSG STARTING FILE xxx

Where:

xxx is the file that is being printed.

```
LPOPR  JOB CANCELED BY THE OPERATOR
               REASON:  xxx
```

Where:

xxx is the reason the operator canceled the job.

LPOPR JOB REQUEUED BY THE OPERATOR

LPOPR JOB WILL RESTART AT THE BEGINNING OF THE xxx

Where:

xxx is the current copy or current file.

LPOPR JOB WILL RESTART AT THE CURRENT POSITION

LPOPR OPERATOR STOPPED CARRIAGE CONTROL SUPPRESSION

LPOPR OPERATOR SUPPRESSED CARRIAGE CONTROL FOR REST OF xxx

Where:

xxx is this file or this job.

LPOPR REQUEUE REASON IS: xxx

Where:

xxx is the reason the job was requeued.

6.2.3 QUEUE Messages -- TOPS-10 Only

%QUECLG CORE LIMIT OF nnP IS GREATER THAN CORMAX OF nnP

You specified a core limit (for a batch job) of nnP with the /CORE switch. This amount is more than is available to any job on the system. The job will be put in the queue but will not be scheduled until the value of CORMAX is raised by the operator.

%QUECLR CORE LIMIT OF nnP RAISED TO MINIMUM OF nnP

The core limit of nnP that you specified with the /CORE switch has been raised by the system to nnP.

?QUECMN CANNOT MODIFY /NOTIFY SWITCH

It is illegal to modify a job in order to set or reset the /NOTIFY switch.

%QUECPB CAN'T PRINT BINARY filespec

You attempted to print the binary file (filespec) with the switch, /PRINT:ASCII, specified (either explicitly or implicitly). Note that /PRINT:ASCII is the default.

?QUEDMI /DEFER REQUEST MUST INCLUDE /CREATE OR /KILL

The /DEFER switch must be used in conjunction with either the /CREATE or the /KILL switch.

?QUEDND INPUT DEVICE NOT A DISK dev:

An input device (dev:) specified in the QUEUE command is not a disk. The QUEUE program can handle files only from disk.

?QUEDRF /DISPOSE:RENAME FAILED

The monitor rejected the request to rename the file to the spool area.

?QUEFNI FILE NAME ILLEGAL IN DEFAULT PATH

There is a syntax error in the /PATH switch specification.

?QUEFRI FILENAME REQUIRED FOR INPUT QUEUE

You must specify a filename for your control file.

?QUEICC IMPROPER COMMAND CODE

This implies a problem with the QUEUE program that is not your fault. You should notify the system administrator.

?QUEIDT INVALID DEVICE TYPE SPECIFIED
(Same as QUEIQN)

?QUEIFT ILLEGAL FORMAT IN TIME SPECIFICATION

You have given more than three values for the /TIME switch. For example, /TIME:1:0:0 is legal and specifies a time of 1 hour, but /TIME:1:0:0:0 is illegal and would result in the display of the above message.

?QUEINF INCORRECT NUMBER OF FILES IN INPUT QUEUE REQUEST

Only two files may be specified in a SUBMIT (or QUEUE INP:) command. The two file specifications are for the control and log files.

?QUEIQN ILLEGAL QUEUE NAME name

The name of the queue specified in the QUEUE command is not a valid queue name. Valid queue names are: INP:,LPT:,LL:,LU:, CDP:,PTP:, and PLT:.

?QUEIUR ILLEGAL USE OF /DISPOSE:RENAME

You cannot modify or kill a job using this switch.

?QUEIUS INVALID UNIT NUMBER SPECIFIED

A unit number not in the range of 0-7 was specified.

?QUELFS LOGIN PLEASE TO USE SWITCH xxx

You must be logged in to run the QUEUE program with the xxx functions where xxx represents CREATE, MODIFY, or KILL.

?QUELQU LISTED QUEUE USER CANNOT INCLUDE SFDS

You cannot ask for a list of queues by SFDs. For example, the command PRINT [30,54,A]/L is illegal and would generate this message.

?QUEMDC MOUNTABLE DEVICE COMMANDS NOT IMPLEMENTED

The version of QUEUE that is currently running is unable to handle mountable device commands.

?QUENAF NOT ALL REQUESTED FILES EXIST

Some of the files in the QUEUE request do not exist and you specified the /ALLFILES switch.

?QUENFI NO FILES IN REQUEST

You did not specify any files in your QUEUE request or none of the files you specified exists.

?QUENQS NO QUEUE SPECIFIED IN /KILL OR /MODIFY

You must specify which queue contains the request you want to kill or modify.

?QUENSD INPUT DEVICE DOES NOT EXIST dev:

You specified an input device (dev:) that does not exist. The device may have been misspelled or it might not be on line.

?QUENSR JOBNAME OR /SEQUENCE OR/REQUESTID REQUIRED

You must specify either the jobname or sequence number (or both) of the request(s) you want to modify or kill.

?QUEOEI OUTPUT EXTENSION IS ILLEGAL

The jobname must not have an extension.

%QUEOTE INPUT QUEUE REQUEST USES ONLY TWO ENTRIES

Only two files may be specified in a SUBMIT (or QUEUE INP:) command. The two file specifications are for the control and log files.

?QUEOWI OUTPUT WILDCARDS ARE ILLEGAL

The jobname specification on a QUEUE/CREATE command and the output specification on a QUEUE/LIST command cannot have wildcard constructions.

%QUEPII /PROTECT:777 IS ILLEGAL, CHANGED TO 677

A protection code of 777 for a queue request is not permissible. The protection was changed to 677.

?QUESJN SPECIFY JOBNAME LEFT OF EQUAL SIGN

The jobname must be to the left of the equal sign.

?QUESVT SWITCH VALUE TOO LARGE n

You specified a value on the /DEPEND switch that is too large.

?QUETCT TIME COMPONENT TOO LARGE n

The amount of time specified on the /AFTER or the /TIME switch is too large. The value of hh:mm:ss for the /AFTER switch cannot be greater than 23:59:59. The value for the /TIME switch cannot be greater than 63:59:59.

?QUETTL TIME TOO LARGE n

A time field contains a value that is larger than the maximum permissible.

?QUEUCI LISTED QUEUE USER CANNOT INCLUDE SFDS

SFDs may not be included as part of the jobname specification.

?QUEWDI WILDCARD ILLEGAL IN INPUT QUEUE FILE DIRECTORY

A wildcard construction is not permissible in an input queue file specification. For example, the command SUBMIT JOB.CTL[40,*] is illegal and would generate this message.

?QUEWID WILDCARD ILLEGAL IN DEFAULT PATH

A wildcard specification may not be used with the /PATH switch.

?QUEWIE WILDCARD ILLEGAL IN INPUT QUEUE FILE EXTENSION name

You must not specify a wildcard extension in your SUBMIT (or QUEUE INP:) command. The command SUBMIT JOB.* is illegal.

?QUEWIN WILDCARDS ILLEGAL WITH/NEW

The use of /NEW implies that the files have not yet been created. Therefore, it is not possible to perform a wildcard search for the file.

?QUEWIQ WILDCARD ILLEGAL IN INPUT QUEUE FILE NAME filename

You must not specify a wildcard filename in your SUBMIT (or QUEUE INP:) command. The command SUBMIT *.CTL is illegal.

6.2.4 SPRINT Messages

STERR ?SPTBFP Bad format for Project-Programmer Number on $JOB Card

STERR ?SPTCCF Error creating file xxx, yyy

Where:

xxx is the file specification.
yyy is the system error that resulted.

STERR ?SPTCNF Control Card not found when expected -Card #nnn

Where:

nnn is a number (in decimal) indicating the card's position in the input stream.

STERR ?SPTECC Error creating BATCH control (CTL) file, xxx

Where:

xxx is the system error that resulted.

STERR ?SPTECL Error creating BATCH log file, xxx

Where:

xxx is the system error that resulted.

STERR ?SPTERI Error reading input file

STERR ?SPTEWC Error writing BATCH control (CTL) file, xxx

Where:

xxx is the system error that resulted.

STERR ?SPTEWF Error writing file xxx, yyy

Where:

xxx is the file specification.
yyy is the system error that resulted.

STERR ?SPTFSE File specification error on $ccc card

Where:

ccc is the named control card.

STERR ?SPTIAS Illegal account string "aaa"

Where:

aaa is the user account string.

STERR ?SPTICC Illegal control card - card #nnn

Where:

nnn is a number (in decimal) indicating the card's position in the input stream.

STERR ?SPTIFJ Improperly formatted $JOB card

STERR ?SPTIMP Incorrect or missing password card

STERR ?SPTIMP Specified password incorrect for user

STERR ?SPTIPN Invalid directory specification - uuu on job card

Where:

uuu is the user identification.

STERR ?SPTIUI Improper use of $INCLUDE

STERR ?SPTIUN Illegal or missing user name

STERR ?SPTJNF $JOB card not found

STERR ?SPTMPP Missing Project-Programmer Number on $JOB Card

STERR ?SPTPNB Specified PPN may not run BATCH jobs.

STERR ?SPTRCI Improper use of $RELOC

STERR ?SPTTMB Too many illegal binary cards

The number of illegal binary cards has exceeded the limit specified with the /ERROR switch on the $JOB card.

STERR ?SPTTMC Too many binary checksum errors

The number of binary checksum errors has exceeded the limit specified with the /ERROR switch on the $JOB card.

STERR ?SPTTMH Too many Hollerith errors

The number of Hollerith errors has exceeded the limit specified with the /ERROR switch on the $JOB card.

STERR ?SPTUUN Unrecognized User Name "uuu" on the $JOB card

Where:

uuu is the user identification.

APPENDIX A

COMMANDS

Appendix A contains an overview of the BATCON and SPRINT commands and of commonly used batch-related system commands.

A.1 BATCH COMMANDS

Use the following commands to control BATCON, the batch controller. These commands, discussed in detail in Chapter 2, may appear in both terminal and card jobs. Precede each command with the system prompt character (. for TOPS-10, @ for TOPS-20).

```
BACKTO
CHKPNT
DUMP
ERROR
GOTO
IF
MESSAGE
NOERROR
NOOPERATOR
OPERATOR
PLEASE
REQUEUE
REVIVE
SILENCE
```

A.2 SPRINT COMMANDS

The following commands may be used only on control cards for SPRINT. These commands are described in detail in Chapter 5.

Command	Applicable Switches
$ALGOL	see $language
$BACKTO	None
$BLISS	See $language
$CHKPNT	None
$COBOL	See $language

$CREATE	/026 /ASCII /BCD /BINARY (-10) /CPUNCH /DOLLARS /NODOLLARS /IMAGE /PLOT (-10) /PRINT /PROTECT: /SUPPRESS /NOSUPPRESS /TPUNCH /WIDTH:
$DATA	/026 /ASCII /BCD /BINARY (-10) /DOLLARS /NODOLLARS /IMAGE /MAP /NOMAP /SUPPRESS /NOSUPPRESS /WIDTH:
$DUMP	None
$EOD	None
$EOJ	None
$ERROR	None
$EXECUTE	/MAP /NOMAP
$FORTRAN	See $language
$GOTO	None
$IF	None
$INCLUDE	/SEARCH
$JOB	/ACCOUNT: /AFTER: /ASSISTANCE: /BATCH-LOG:(-20) /BATLOG:(-10) /CARDS: /CORE:(-10) /DEPEND: /FEET: /JOBNAME: /LOCATE: /LOGDISP: /NAME:(-10) /OUTPUT: /PAGES: /PPN:(-10)

	/PRIORITY:
	/RESTART
	/NORESTART
	/SEQUENCE:
	/TIME:
	/TPLOT:
	/UNIQUE:
	/USER:(-20)
$LABEL	None
$language	

Substitute the following for 'language': ALGOL, BLISS, COBOL, FORTRAN, MACRO, SIMULA, SNOBOL

	/026
	/ASCII
	/BCD
	/CREF
	/DOLLARS
	/NODOLLARS
	/LIST
	/NOLIST
	/SUPPRESS
	/NOSUPPRESS
	/WIDTH:
$MACRO	See $language
$MESSAGE	/WAIT
	/NOWAIT
$NOERROR	None
$NOOPERATOR	None
$OPERATOR	None
$PASSWORD	None
$RELOCATABLE	None
$REQUEUE	None
$REVIVE	None
$SEQUENCE	None
$SILENCE	None
$SIMULA	See $language
$SNOBOL	See $language
$TOPS	/026
$TOPS10	/ASCII
$TOPS20	/BCD
	/DOLLARS
	/NODOLLARS
	/SUPPRESS
	/NOSUPPRESS
	/WIDTH:

A.3 SYSTEM COMMANDS

The system commands listed below are those that you as a batch system user might commonly use. They allow you to examine your job, manipulate it in various ways, and provide you with access to a number of input/output devices.

A.3.1 Job Control Commands

Use the following job control commands to submit, modify, and cancel jobs in the output or batch input queues. Chapters 3 and 4 discuss these commands in detail.

NOTE

TOPS-20

These commands are EXEC commands that do not destroy your core image. You can CTRL/C out of a program, use one of these commands, and then return to your program.

A.3.1.1 SUBMIT Command -

TOPS-10 Format:

```
SUBMIT jobname/switch(es)=/switch(es) control file/switch(es),
log file/switch(es)
```

TOPS-20 Format:

```
SUBMIT (BATCH JOB) /switch(es) control file/switch(es),...
```

Applicable Switches:

```
/ACCOUNT:
/AFTER:
/ASSISTANCE:
/BATCH-LOG: (-20)
/BATLOG: (-10)
/BATOPT: (-10)
/BEGIN:
/CARDS:
/CHECK (-10)
/CONNECTED-DIRECTORY: (-20)
/CORE: (-10)
/DELETE
/DEPEND:
/DESTINATION: (-10)
/DESTINATION-NODE: (-20)
/DISTRIBUTION: (-10)
/ERPROTECTION (-10)
/FAST (-10)
/FEET:
/JOBNAME:
/KILL (-10)
/LIST: (-10)
/LOGDISPOSITION: (-20)
/LOGNAME: (-20)
/NONOTIFY (-10)
/NORESTART (-10)
/NOTIFY:
/OKPROTECTION (-10)
/OPTION: (-10)
/OUTPUT:
/PAGES:
/PATH: (-10)
/PHYSICAL (-10)
/PRESERVE
/PRIORITY:
/PROCESSING: (-10)
/PROCESSING-NODE: (-20)
/PROTECTION: (-10)
/READER
/REQUESTID: (-10); with /KILL
 or /MODIFY only)
/RESTART: (-10)
/RESTARTABLE: (-20)
/SEQUENCE:
/SITGO (-10)
/STREAM: (-10)
/TAG:
/TIME:
```

```
/METERS: (-10)                     /TPLOT:
/MODIFY (-10)                      /UNIQUE:
/NEW (-10)                         /USER: (-20)
/NONEW (-10)                       /USERNAME: (-10)
```

A.3.1.2 Modify Commands -

TOPS-10 Format:

```
SUBMIT
PRINT
PLOT        jobname=/MODIFY/switches(es)
CPUNCH
TPUNCH
```

TOPS-20 Format:

```
                        | BATCH
                        | CARDS                (ID) jobname or
MODIFY (REQUEST TYPE)   | PAPER-TAPE            request id/switch(es)
                        | PLOT
                        | PRINT
```

Applicable switches for batch requests:

```
/AFTER:
/BEGIN:
/CARDS:
/CORE: (-10)
/DEPEND:
/DESTINATION: (-10)
/DESTINATION-NODE: (-20)
/FEET:
/JOBNAME:
/METERS: (-10)
/OUTPUT:
/PAGES:
/PRESERVE
/PRIORITY:
/PROTECTION: (-10)
/RESTART:
/SEQUENCE:
/TAG:
/TIME:
/TPLOT:
/UNIQUE:
/USER: (-20)
(enabled wheel or enabled oper. only)
```

A.3.1.3 Cancel Commands -

Refer to your operating system commands manual for additional information on cancelling jobs.

TOPS-10 Format:

```
          | BATCH-REQUEST
          | CARD-PUNCH-REQUEST          request id, jobname, or
   CANCEL | MOUNT-REQUEST               wild card job name
          | PAPER-TAPE-REQUEST
          | PLOTTER-REQUEST
          | PRINTER-REQUEST
```

TOPS-20 Format:

```
                         | BATCH
                         | ARCHIVE
                         | CARDS
                         | MOUNT         (ID) jobname or
   CANCEL (REQUEST TYPE) | PAPER-TAPE    request id/switch
                         | PLOT
                         | PRINT
                         | RETRIEVE
```

Applicable Switches:
/SEQUENCE:
/USER: (-20; enabled wheel or enabled operator only)

To cancel all jobs under your name, specify * as the jobname.

A.3.1.4 PRINT Command -

TOPS-10 Format:

```
PRINT jobname/switch(es)=/switch(es) filespec/switch(es),...
```

TOPS-20 Format:

```
PRINT (FILES) /switch(es) filespec/switch(es),...
```

Refer to your system commands manual for a list of PRINT command switches.

A.3.2 Information Commands

TOPS-10 Format:

```
SUBMIT
PRINT
PLOT     /switch
CPUNCH
TPUNCH
```

In all cases, you can enter the command without switches to obtain information on the queue.

Applicable switches for the QUEUE-class information commands listed above:

```
/LIST:
/FAST
/CHECK
/STREAM:(with SUBMIT only)
/DESTINATION:
/PROCESSING:(with SUBMIT only)
```

or

```
SHOW | ALLOCATION
     | QUEUES/switch
```

Applicable switches for the SHOW QUEUES command:

```
/ALL
/BRIEF
/FULL
/USER:[p,pn]
```

TOPS-20 Format:

```
INFORMATION (ABOUT) | BATCH-REQUESTS       /switch
                    | OUTPUT-REQUESTS
```

Applicable Switches for the TOPS-20 INFORMATION Command:

```
/ALL
/FAST
/PROCESSING-NODE:   (with BATCH-REQUESTS only)
/user:
```

Refer to the TOPS-20 Commands Reference Manual for a complete list of arguments (other than BATCH- or OUTPUT-REQUESTS) you can specify with the INFORMATION command.

A.3.3 Tape Control Command Formats

Refer to your system commands manual and Tape Processing Manual for descriptions of the tape control commands.

APPENDIX B

SWITCHES

B.1 SWITCH DEFAULTS

If you commonly use the same switches and switch values with the SUBMIT or output queue commands, you may wish to set your own default values for the switches you use:

TOPS-10:

For the SUBMIT command, enter a line into the file DSK:SWITCH.INI [p,pn] consisting of the following:

```
SUBMIT/switch(es)
```

Every time you submit a batch job, these switches and the values you specify for them take effect. They override the defaults set at batch system installation.

You can set defaults for the output queue commands by following the same format as above, substituting such commands as PRINT or PLOT for SUBMIT.

Refer to the end of this section for the list of switches you can put into DSK:SWITCH.INI [p,pn].

TOPS-20:

Set switch defaults by giving the SET DEFAULT command with an argument of SUBMIT or one of the output queue commands, for example:

1. SET DEFAULT (FOR) SUBMIT/switch(es)

2. SET DEFAULT (FOR) PRINT/switch(es)

You can place the SET DEFAULT command in three different command files, which are read by the system at various times:

1. COMAND.CMD is read when you issue the LOGIN, PUSH, or SUBMIT commands. This is the only command file that is read when you PUSH.

2. LOGIN.CMD is read only when you log in. This file is read after COMAND.CMD.

3. BATCH.CMD is read only when you submit a batch job. This file is also read after COMAND.CMD.

When the system reads a command file, the switches you specified therein take effect. Thus, you can override the options in COMAND.CMD with those in LOGIN.CMD or BATCH.CMD.

If you have been resetting switch defaults manually and want to return to the options in one of the command files, issue the TAKE system command as follows:

```
@TAKE LOGIN.CMD
```

This will execute the commands in LOGIN.CMD, resetting your switch defaults to their original values.

SUBMIT Switches Available for Defaulting with TOPS-10 and TOPS-20:

```
/ACCOUNT:
/AFTER:
/ASSISTANCE:
/BATCH-LOG: (-20)
/BATLOG: (-10)
/BATOPT: (-10)
/BEGIN:
/CARDS:
/CONNECTED-DIRECTORY: (-20)
/CORE: (-10)
/DELETE
/DEPEND: (-10)
/DEPENDENCY-COUNT: (-20)
/DESTINATION: (-10)
/DESTINATION-NODE: (-20)
/DISPOSE:
/DISTRIBUTION: (-10)
/ERPROTECTION (-10)
/FEET:
/JOBNAME:
/LOGNAME: (-20)
/NEW (-10)
/NONEW (-10)
/NOTIFY:
/OUTPUT:
/PAGES:
/PATH: (-10)
/PRESERVE
/PRIORITY:
/PROCESSING: (-10)
/PROCESSING-NODE: (-20)
/READER
/RESTART: (-10)
/RESTARTABLE: (-20)
/TAG:
/TIME:
/UNIQUE:
/USER: (-20)
  (enabled wheel or enabled operator only)
/USERNAME: (-10)
```

B.2 TIME AND DATE SWITCHES

The following rules apply to all the switches for the **$JOB** card and the **TOPS-10 SUBMIT command** that require a time and/or date to be specified. See NOTE at the end of this section for important differences with the TOPS-20 SUBMIT command. Refer to your system commands manual for complete information on time and date formats for your particular system.

When specifying a time of day (hh:mm:ss)

1. The colon (:) or colons must be included.

2. Times will be right aligned before they are interpreted. That is, if all fields are not present, the rightmost field is interpreted as the number of seconds in the case where an argument of hh:mm:ss is required, and the rightmost field is interpreted as the number of minutes when an argument of hh:mm is required. Some examples are given below:

/TIME:30	means	30 seconds (see last NOTE in this section)
/TIME:45:00	means	45 minutes (see last NOTE in this section)
/TIME:1:15:00	means	1 hour and 15 minutes

When specifying a date, the format is dd-mmm-yy and:

1. The hyphen (-) must be included.

2. At least the day and the month are required. (See last NOTE in this section.)

NOTE

The month, "mmm," must be a 3-letter month abbreviation (such as JAN for January). Other date formats (such as 07 for July) are accepted but are not recommended because they could be ambiguous to the system.

3. If the year is omitted, the date (and its associated time, if present) will be interpreted to mean the next occurrence of that date (and time).

4. If the time argument is omitted from a date specification, the time is assumed to be midnight on the specified date, that is, at the **beginning** of that date.

   In the examples below assume that the current date is 18-OCT-88.

/AFTER:19-OCT-88	means midnight on October 19, 1988
/AFTER:17-OCT	means midnight on October 17, 1988

NOTE

With the TOPS-20 SUBMIT command, /TIME:30 means 30 minutes and /TIME:45:00 means 45 hours. Also, the date format requires the year.

If you specify NO to this switch, and the system encounters a request to the operator in your control file, it terminates the job immediately.

(SUBMIT,$JOB)

/BATCH-LOG:arg (-20)
/BATLOG:arg (-10)

Directs placement of the log file on the disk. You may use one of the following three arguments:

APPEND	The log file is appended to an existing one of the same name.
SUPERSEDE	The log file replaces an existing one of the same name.
SPOOL	The log file is written to the system spooling area rather than to your disk area, thereby saving space in your directory. The log file is automatically printed from the spooling area. With SPOOL, the log file is always deleted, even if you specify /PRESERVE (-10) or /LOGDISP:KEEP(-20).

Examples:

1. If you submit a batch job with the command SUBMIT FILA/BATCH-LOG: SUPERSEDE, then FILA.LOG will be written as a new log file. Any old copy of FILA.LOG will be deleted.

2. If you submit a batch job with the command SUBMIT ZFIL/BATCH-LOG: SPOOL, then the log file will be written to the system spooling area.

(SUBMIT,$JOB)

/BATOPT:arg (-10)

Specifies a LOGIN option line to read for LOGIN switches to apply to the batch job. The option name that you specify with the /BATOPT switch must match a line in the SWITCH.INI file that appears as:

LOGIN:option-name/switches

(SUBMIT)

/BCD

Causes the card deck to be read as 026 card format.

($ALGOL,$BLISS,$COBOL,
$CREATE,$DATA,
$FORTRAN,$MACRO,$SIMULA,
$SNOBOL,$TOPS,
$TOPS10,$TOPS20)

/BEGIN:n

When used with SUBMIT, causes processing to begin on the nth line of the control file. When used with PRINT, causes printing to begin on the nth page.

Use this switch for a control file or a print file that can fit different applications depending on where processing begins. (See also the /TAG: switch.)

(SUBMIT,PRINT,MODIFY)

/BINARY (-10)

Causes the card deck to be read as checksummed-binary card format.

($CREATE,$DATA)

/CARDS:nn

Specifies the maximum number of cards (up to 10,000) that can be punched by the job (in decimal). If you omit this switch, no cards will be punched.

($JOB,SUBMIT,MODIFY)

/CHECK (-10)

Lists all the jobs in the specified queue that are under your project-programmer number.

(SUBMIT, all output queue commands)

/CONNECTED-DIRECTORY:<directory> (-20)

Allows an enabled wheel or enabled operator to run a job under a directory different from the directory under which the job is submitted.

The following example illustrates the difference between using this switch and using the /USER switch:

Assume that you are an enabled user logged in as OPERATOR and that the system you are using contains a files-only directory, LIBRARY.

1. If you submit a batch job with the command SUBMIT FOO.CTL/ USER:SMITH, then the batch job is logged in to user SMITH and connected to directory SMITH. Accounting charges are made to user SMITH.

2. If you submit a batch job with the command SUBMIT FOO.CTL/ CONNECTED-DIRECTORY: LIBRARY, then the batch job is logged in to user OPERATOR and is connected to directory LIBRARY. Accounting charges are made to user OPERATOR.

This switch is equivalent to the /PATH: switch on TOPS-10.

(SUBMIT)

/CORE: (-10) nnK,nnP, nnB, or nnW	Specifies the maximum amount of core (in decimal) that can be used by the job up to the maximum allowed at your installation. You have the option of specifying in either words, pages, blocks, or K words. The default usually is 32K words. The value is ignored if core limits are not enforced at your installation. ($JOB,SUBMIT)
/CPUNCH	Causes SPRINT to place the disk file created with $CREATE into the card punch output queue. ($CREATE)
/CREF	Creates a file that the CREF program processes to produce a cross-referenced listing for your program. The default is that no CREF listing is created. ($FORTRAN,$MACRO)
/DELETE	Causes the file to be deleted after the job has completed execution. If you use this switch to delete the log file under TOPS-10, the file will not be deleted until it has been printed. For example, `.SUBMIT JOB=CONTROL, LOG/DELETE` causes the log file LOG to be deleted immediately after being printed. (SUBMIT, MODIFY, all output queue commands)
/DEPEND:nn	Allows you to coordinate the selection of interdependent jobs for processing. Specifically, this switch sets or modifies the job's dependency count. The job will not run unless the dependency count is set to zero. If you do not use this switch, no dependency is assumed and the job is selected to run based only upon such factors as priority value, output quantities, etc. (See section 3.5, SUBMITTING RELATED JOBS, for more information on this switch.) ($JOB,SUBMIT,MODIFY)
/DEPENDENCY-COUNT:nn (-20)	Performs the same function as the /DEPEND switch (above). (SUBMIT,MODIFY)

/DESTINATION: (-10) /DESTINATION-NODE: (-20)	Specifies the DECNET network node or IBM remote job entry station to whose line printer the log file and all spooled output are to be sent. The node name must be of six or fewer characters. (On TOPS-20 systems, this switch is a no-op if the node name specified is a DECnet node name.) On TOPS-10 systems, you can use this switch to obtain queue listings for the specified node. (SUBMIT, all output queue commands)
/DISTRIBUTION:arg (-10)	Specifies text to place in the distribution field on the banner page of output listings. For batch input requests, the distribution text is printed on the banner page of the log file listing. You can use this field to include mailing information, or the location where the operator should leave the listing. The text field may be up to 39 alphanumeric characters, including punctuation and spaces if the text is placed in quotation marks.
/DOLLARS	Normally, a dollar sign ($) in column 1 of a punched card indicates that the card is a SPRINT control card and that a SPRINT command immediately follows the dollar sign. The /DOLLARS switch allows a card with a dollar sign in column 1 to be treated as program data (and not a SPRINT control card), provided that the card is not one of the following: 1. $JOB 2. $EOD 3. $EOJ NOTE Any deck for which the /DOLLARS switch is specified must terminate with one of the three cards mentioned above. ($ALGOL,$BLISS,$COBOL, $CREATE,$DATA, $FORTRAN,$MACRO,$SIMULA, $SNOBOL,$TOPS,$TOPS10, $TOPS20)
/ERPROTECTION (-10)	Returns an error message if processing the request requires a violation of protection codes. The system defaults to this action. (SUBMIT, all output queue commands)
/FAST (-10)	Lists the batch input queue entries in an abbreviated format. (SUBMIT, all output queue commands)

SWITCHES

/FEET:nn	Indicates the maximum number of feet (decimal) of paper tape that will be punched by the job. If you omit this switch, no paper tape will be punched. ($JOB,SUBMIT,MODIFY)
/IMAGE:nn	Causes the card deck to be read in image mode. The switch must be followed by a decimal number in the range 2 thru 80.

This switch causes following cards to be read in image mode until either the end-of-file is reached or a card is read that contains punches in all rows of column 1 and in all rows of column nn and blank otherwise. Neither $EOD nor any other control card is recognized until the system finds this card.

If you do not give a value for nn, the system assumes 2.

A previously specified /WIDTH switch is not acknowledged when cards are read in image mode.

A typical control card, such as $CREATE TEST.IMG/IMAGE, causes the system to read all cards following this card and write them to the disk file TEST.IMG until it encounters a card that is fully punched in columns one and two and blank otherwise.

A control card written as $CREATE FILX/IMAGE:5 causes the system to read and write the cards following this card until it encounters a card which is fully punched in columns one and five and blank otherwise. Specify an argument for the /IMAGE switch, as in this case, if for some reason you wish to consider as data a card punched in columns one and two.

The newly-created disk file looks like this:

1. One 80-column card occupies 27 words on disk (3 12-bit bytes per word - 1 indicates punch; 0 indicates no punch).
2. Each card column occupies one byte with the 12-punch as the high-order bit (leftmost bit of the byte).
3. Byte number 81 is always zero (that is, bits 24-35 of every 27th word are zero).

Figure B-1 at the end of Appendix B illustrates these three points.

($CREATE,$DATA)

/JOBNAME:name	Assigns a 1- to 6-character name to the associated job. If you omit this switch from the $JOB card, then SPRINT creates a name of the form JBxxxx (where x represents a random character chosen by SPRINT) for the control and log files. If you omit this switch from the SUBMIT command, the job name will be taken from the log file name on TOPS-10 and from the control file name on TOPS-20. For the PRINT command, the job name is taken from the first six characters of the first file in the request. ($JOB, SUBMIT, MODIFY, CANCEL (-20), all output queue commands)
/KILL (-10)	Allows you to cancel or kill the running of a job by removing the specified entry from the specified queue. You can use this switch even if the submitted request has been started. Refer to Chapter 3, JOB CONTROL, for examples. (SUBMIT, all output queue commands)
/LIST	Causes a program compilation listing to be generated. The system defaults to this switch. ($ALGOL,$BLISS,$COBOL, $FORTRAN,$MACRO,$SIMULA, $SNOBOL)
/LIST:arg (-10)	Lists the entries for the specified queue. The arguments are ALL, FAST, and JOBS. (SUBMIT, all output queue commands)
/LOCATE:name/number	Specifies the network node to whose line printer the job's output is to be sent. With TOPS-10, /LOCATE accepts the node number (in octal) or the node name. With TOPS-20, only the node name is accepted. ($JOB)
/LOGDISP:arg	Specifies the disposition of the LOG file after the batch job has been processed. You may specify one of the following three arguments: 1. DELETE - deletes the log file after printing it. This is the default for card jobs. 2. PRESERVE - saves the log file after printing it. This is the default for jobs submitted with the SUBMIT command.

3. KEEP - same as PRESERVE. Use this argument instead of PRESERVE when submitting jobs from a timesharing terminal under TOPS-20. You can use KEEP or PRESERVE with card jobs.

($JOB,SUBMIT (-20))

/LOGNAME:name (-20)	Normally, the log file takes the name of the control file and adds the extension .LOG. This switch allows you to specify your own logname (up to 39 characters) and your own file extension (up to 39 characters). If you specify a name but do not specify an extension, the extension will default to .LOG. Note that use of the /READER switch nullifies the effects of /LOGNAME. This switch provides the only way for you to specify a log filename. The command, SUBMIT FIL1.CTL, FIL2.LOG, does not specify that the log file for FIL1.CTL will be FIL2.LOG. Instead, it submits two batch jobs, one with a .CTL extension and one with an .LOG extension. (SUBMIT)
/MAP	Causes a loader map to be generated and printed. The default is /NOMAP. ($DATA,$EXECUTE)
/METERS:n (-10)	Specifies the number of meters of paper tape that can be punched by the job. (SUBMIT)
/MODIFY (-10)	Allows you to modify a job's parameters. Refer to Chapter 3, JOB CONTROL, for examples that use this switch. (SUBMIT, all output queue commands)
/NAME:name (-10)	Specifies a user's name that can be up to 12 characters. Enclose 'name' in quotes if 'name' contains any blank characters. This switch is optional unless your installation requires the user's name in addition to the project-programmer number and password when you log in. If this is the case, then the name supplied here must match the name listed in the accounting file for the specified project-programmer number and password. If the name does not match, an error message will be issued and the job will not be run.

If your installation does not require a name and one is specified, then the name (if different) overrides the name in the accounting file and will appear on the output as specified on the $JOB card.

($JOB)

/NEW (-10)

Allows the request to be accepted even if the file does not yet exist. For example, you would use this switch when the disk structure containing the file is not yet mounted.

(SUBMIT, all output queue commands)

/NODOLLARS

If this switch is in effect and a card beginning with a dollar sign ($) is encountered, SPRINT examines column 2 of the card. If column 2 contains a dollar sign, then the first dollar sign is ignored and all information from column 2 to the end of the card is treated as data. This feature is useful if you want to include a dollar sign as the first character in your data.

If column 2 contains any nonalphabetic character, then the entire card (including column 1) is treated as data.

If column 2 contains an alphabetic character, then the card is treated as a SPRINT control card.

For example:

1. $CREATE - interpreted as SPRINT control card '$CREATE'.
2. $$CREATE - interpreted as data - '$CREATE'. (The first dollar sign is ignored.)
3. $3CREATE - interpreted as data - '$3CREATE'.

/NODOLLARS is the default.

Any deck for which the /NODOLLARS switch is in effect (either explicitly or by default) can be terminated with any SPRINT control card.

($ALGOL,$BLISS,$COBOL,
$CREATE,$DATA,
$FORTRAN,$MACRO,$SIMULA,
$SNOBOL,$TOPS,$TOPS10,
$TOPS20)

/NOLIST

Prevents a program compilation listing from being generated. /LIST is the default.

($ALGOL,$BLISS,$COBOL, $FORTRAN,$MACRO,$SIMULA, $SNOBOL)

/NOMAP

Prevents a loader map from being generated. /NOMAP is the default.

($DATA,$EXECUTE)

/NONEW (-10)

Does not allow the request to be accepted if the file does not exist. Opposite of /NEW.

(SUBMIT, all output queue commands)

/NONOTIFY (-10)

Same as /NOTIFY:NO

/NORESTART

Specifies that the job should not be restarted after the system has crashed and been restored.

($JOB,SUBMIT (-10))

/NOSUPPRESS

Prohibits suppression of trailing blanks. /NOSUPPRESS is the default.

($ALGOL,$BLISS,$COBOL, $CREATE,$DATA, $FORTRAN,$MACRO,$SIMULA, $SNOBOL,$TOPS, $TOPS10,$TOPS20)

/NOTIFY:arg

Forces the system to send a short message to your terminal when the job ends. For example, if you type

```
@SUBMIT TEST/NOTIFY
```

the system will respond as follows:

```
[Job TEST Queued, Request-ID 135, Limit 0:05:00]
@
[FROM SYSTEM: Job TEST request #135 finished executing at 09:57:49]
```

Arguments:YES
NO

The default is YES if you use the switch but omit a value. The default is NO if you do not use the switch.

(SUBMIT, all output queue commands)

/NOWAIT

Causes the job to continue after typing a message without waiting for a response from the operator. /NOWAIT is the default.

($MESSAGE)

/OKPROTECTION (-10)	Suppresses issuance of an error message when a protection error occurs. (SUBMIT, all output queue commands)
/OPTION:option (-10)	For batch requests, executes the line beginning with 'SUBMIT:option' in the SWITCH.INI file, where 'option' is the identifier you choose for this line. Substitute one of the output queue-class commands for SUBMIT when applicable. Section B.1, Switch Defaults, discusses SWITCH.INI. (SUBMIT, all output queue commands.)
/OUTPUT:arg	Determines whether the log file should be printed. You may specify one of the following arguments: 1. ALWAYS - print the log file. 2. NOLOG - suppress printing of the log file. 3. ERROR - print the log file only if an error occurs. 4. LOG - print the log file. If you omit this switch, the log file will be printed. LOG and ALWAYS are equivalent. However, with SUBMIT, LOG applies to TOPS-10 only, and ALWAYS applies to TOPS-20. ($JOB,SUBMIT,MODIFY)
/PAGES:n	Specifies the maximum number of pages (in decimal) to be printed by the job, including the log file and compilation listings. The default is 200 pages. The number of pages is a function of the number of disk blocks (-10) or disk pages (-20) to be printed and of the number of copies of files to be printed. ($JOB,SUBMIT,MODIFY)
/PATH:[dir] (-10)	Runs the job from the specified directory. This switch is equivalent to the TOPS-20 /CONNECTED-DIRECTORY: switch. (SUBMIT)
/PHYSICAL (-10)	Suppresses logical device names for the specified file. The system defaults to this switch after you give the KJOB command. (SUBMIT, all output queue commands)

/PLOT (-10)

Causes SPRINT to place the disk file created by the $CREATE card into the plotter output queue.

($CREATE)

/PPN:[p,pn] (-10)

Specifies a TOPS-10 user directory other than the one from which you submitted your job. The job's password must correspond to the directory you specify with this switch.

System-independent jobs require the /PPN: and /USER: switches.

($JOB)

/PRESERVE

Saves the file after completion of the job. With the SUBMIT command, /PRESERVE saves the control file, or, on a TOPS-10 system, can save the log file. This switch is the default for all files except those with the .LST extension.

The /PRESERVE switch has two uses. It:

1. Prevents files with the .LST extension from being deleted upon completion of the job.

2. Prevents a particular file from being deleted when it is specified with a command that has a global /DELETE switch. For example:

```
PRINT/DELETE FILE1,FILE2/PRESERVE,FILE3
```

   The command shown above will delete FILE1 and FILE3 after printing, but it will not delete FILE2.

(SUBMIT, MODIFY, all output queue commands)

/PRINT

Causes SPRINT to place the disk file created by the $CREATE card into the line printer output queue. If you omit this switch the file is not printed.

($CREATE)

/PRIORITY:nn	Assigns the job a priority value (normally 1 to 20 in decimal for unprivileged users; 1-63 for privileged) which determines when the job will run relative to other jobs in the specific queue. The system always selects the job with the highest priority value. The highest value you can specify is usually 20; only the operator or an enabled wheel can exceed this value to a maximum of 63. The system manager can change your limit, which is set at batch system installation time (through the GALGEN program). The default, also modifiable by the operator, is usually 10 if you do not use this switch. If you use the switch but omit a value, the default is 20 on TOPS-10 systems; however, you get an error message on TOPS-20 systems. ($JOB,SUBMIT,MODIFY, all output queue commands)
/PROCESSING: (-10) /PROCESSING-NODE: (-20)	Specifies the IBM host system on whose CPU the JCL batch job is to be run. The node name must be of six or fewer characters. If the local node name is specified, then the batch job is processed locally. However, if any other DECnet node name is specified, then the batch job is queued by QUASAR but never executed. You can also use this switch to obtain a batch queue listing for the specified node. (SUBMIT, INFORMATION BATCH-REQUESTS (-20))
/PROTECTION:nnn (-10)	Specifies a protection code for the log file or for queue listing files that are written to your disk area. Refer to the <u>TOPS-10 Operating System Commands Manual</u> for a discussion of protection codes. With $CREATE, this switch protects the newly created file according to the specified code. ($CREATE, SUBMIT, all output queue commands)
/READER	Causes a disk-resident card job to be processed as though it had been punched on cards and submitted through the card reader. Create this file of card images using one of the on-line editors supported by your system. (SUBMIT)

/REQUESTID:n (-10)	Specifies the request id number of the job you wish to modify or kill. The request id number, assigned to your job by the system, distinguishes your job from others in the queue. The sequence number (see /SEQUENCE:) also identifies your job, but you can assign the sequence number; whereas the request id number is assigned exclusively by the system. It is recommended that you use the request id number rather than the sequence number. (with /KILL or /MODIFY - SUBMIT, all output queue commands)
/RESTART	Specifies that the job may be restarted after the system has crashed and been restored. ($JOB)
/RESTARTABLE:arg	Specifies whether the job should be restarted after the system has crashed and been restored. Only the first six characters of this switch are valid for TOPS-10 systems. Arguments:YES NO Default: NO (SUBMIT,MODIFY)
/SEARCH	Causes the file to be loaded in library search mode. ($INCLUDE)
/SEQUENCE:nn	Specifies the job's "sequence number" (decimal) within the particular queue. The system creates a unique sequence number for the job if you do not use this switch. When used with one of the modify or cancel commands, this switch allows you to specify the sequence number of the job you wish to modify or kill. This is useful to distinguish between duplicate jobnames. However, it is recommended that you use the request id number for these purposes. ($JOB, SUBMIT, MODIFY, CANCEL (-20), all output queue commands)
/SITGO (-10)	Sends your job to the SITGO compiler rather than to BATCON for processing. You can use this switch only if your system supports SITGO. (SUBMIT)

/STREAM: (-10)	Specifies the batch stream whose queue entries should be displayed. Refer to the TOPS-10 Operator's Command Language Reference Manual for a discussion of batch streams. (SUBMIT)
/SUPPRESS	Suppresses trailing blanks. /NOSUPPRESS is the default. ($ALGOL,$BLISS,$COBOL, $CREATE,$DATA, $FORTRAN,$MACRO,$SIMULA, $SNOBOL,$TOPS, $TOPS10,$TOPS20)
/TAG:label	Specifies a label (1 to 6 alphanumeric characters) for a line in the control file. Processing of the control file will begin at the line with the specified label. For example, if the fifth line in your control file was START::@EXECUTE TEST, you could specify /TAG:START to force execution of the control file to begin on that line. Use this switch for a control file that can fit different applications depending on where processing begins. (See also the /BEGIN: switch.) (SUBMIT)
/TIME:hh:mm:ss	Specifies the maximum allowable CPU time for the associated job. If you do not use the switch, the default time is 5 minutes. If you use the switch with SUBMIT and you omit a value, the default time is 1 hour. The $JOB card requires an argument to the /TIME switch. If your job exceeds the specified or default time limit, it will be terminated immediately unless you have made proper use of the %TERR or %FIN label in your control file. (See Section 2.5.3, Reserved Labels.) Refer to Section B.2 for additional information on time and date switches. ($JOB,SUBMIT,MODIFY)
/TPLOT:mm	Specifies the maximum amount of plotter time that the job can use (in minutes). If you omit this switch, no plotter time is allotted. If you use the switch with SUBMIT and you omit a value, ten minutes is allowed. The $JOB card requires an argument to the /TPLOT: switch. ($JOB,SUBMIT,MODIFY)

SWITCHES

/TPUNCH

Causes SPRINT to place the disk file created by the $CREATE card into the paper-tape punch output queue. If you omit this switch, the file is not punched.

($CREATE)

/UNIQUE:n

Specifies how jobs are to be protected from the effects of other jobs running in the same directory.

n=0 Means no protection; that is, more than one batch job may be run concurrently using the directory that you specified on either the $JOB card or with the SUBMIT command.

n=1 Means only one batch job at a time can be run using the specified directory.

n=2 Means that the job is run in a unique SFD. This argument is valid for the $JOB card on TOPS-10 systems only.

n=NO Means the same as n=0.

n=YES Means the same as n=1.

The default value is 1 or YES. Use this switch to prevent (or allow) two or more jobs from updating the same file at the same time.

($JOB,SUBMIT,MODIFY)

/USER:name (-20)

Unless this switch is specified with the $JOB card, it may be used by enabled wheels or enabled operators only. It has two uses:

1. When used with the $JOB card or with the PRINT or SUBMIT command, it allows a job to be run under or printed for a user name different from the user name under which the job was submitted.

2. When used with the MODIFY or CANCEL command, it allows these commands to be directed towards a particular user's jobs.

The argument to this switch (name) must be a valid user name. Also, the password for this job must correspond to the specified user name.

System independent jobs require the /USER: and /PPN: switches on the $JOB card.

To see the difference between using this switch and using the /CONNECTED-DIRECTORY switch, see the example given under the /CONNECTED-DIRECTORY switch.

($JOB,SUBMIT,MODIFY,CANCEL, all output queue commands)

/USERNAME:arg (-10)

Specifies the user name field for the banner page of output listings. For batch input requests, the user name is printed on the banner page for the log file listing. This field can contain up to 39 alphanumeric characters, and may include punctuation and spaces if the name is placed in quotation marks.

The following are alternative ways for users running under [1,2] to submit a batch job on behalf of another user:

```
.SUBMIT jobname[P,PN]=control file/switches,log file/switches

                          or

        .SUBMIT [P,PN]=control file/switches,log file/switches
```

where: [P,PN] is the project-programmer number of the desired user.

(SUBMIT)

/WAIT

Causes the job to stop and wait for a response from the operator before continuing. The operator may respond in such a way as to either continue the job or terminate the job.

($MESSAGE)

/WIDTH:nn

Causes columns 1 through nn (inclusive) of each card to be read. The remaining columns are ignored. If you do not use this switch or if you use it without specifying an argument, SPRINT will assume a value of 80.

If you do not specify an argument, a warning message will be issued, but SPRINT will still assume a value of 80 for the /WIDTH switch.

($ALGOL,$BLISS,$COBOL,
$CREATE,$DATA,
$FORTRAN,$MACRO,$SIMULA,
$SNOBOL,$TOPS,
$TOPS10,$TOPS20)

```
CARD:
-----
                 ----------------------------------|
  HIGH          /|                                 |
 ORDER-------->/ |                                 |
  BIT         /| |                                 |
             / | |                                 |
             | | |                                 |
    LOW      | | |                                 |
   ORDER---> | | |                                 |
    BIT      ---------------------------------------
              ^ ^
              | |
       COLUMNS 1 AND 2
             OF CARD

CARDS ON DISK:
--------------
               HIGH          HIGH       HIGH
               ORDER         ORDER      ORDER
                BIT           BIT        BIT
                 |      LOW    |    LOW   |     LOW
                 |     ORDER   |   ORDER  |    ORDER
                 |      BIT    |    BIT   |     BIT
                 |       |     |     |    |      |
                 v       v     v     v    v      v
                 0       11 12       23 24       35
                |-------------------------------------|
                |  COL 1     |   COL2     |   COL3    |
                |-------------------------------------|
                |  COL4      |   COL5     |   COL6    |
                |-------------------------------------|
                |                                     |
                |-------------------------------------|
                |                 .                   |
                |                 .                   |
                |                 .                   |
                |-------------------------------------|
                |  COL79     |   COL 80   |    0      |<- CONTENTS OF
BEGINNING OF    |-------------------------------------|   BYTE 81 IS
NEXT CARD IN -> |  COL 1     |   COL 2    |   COL 3   |   ZERO
   FILE         |-------------------------------------|
                |                                     |
                |-------------------------------------|
                |                 .                   |
                |                 .                   |
```

Figure B-1. Layout of Card Data That is Transferred to Disk Using the /IMAGE Switch with $CREATE or $DATA

APPENDIX C

CARD CODES

ASCII AND DEC-026 CARD CODES

Character	Octal Code	Card Column Punches ASCII	DEC-026
NULL	00	12-0-9-8-1	12-0-9-8-1
CTRL-A	01	12-9-1	12-9-1
CTRL-B	02	12-9-2	12-9-2
CTRL-C	03	12-9-3	12-9-3
CTRL-D	04	9-7	9-7
CTRL-E	05	0-9-8-5	0-9-8-5
CTRL-F	06	0-9-8-6	0-9-8-6
CTRL-G	07	0-9-8-7	0-9-8-7
CTRL-H	10	11-9-6	11-9-6
TAB	11	12-9-5	12-9-5
LF	12	0-9-5	0-9-5
VT	13	12-9-8-3	12-9-8-3
FF	14	12-9-8-4	12-9-8-4
CR	15	12-9-8-5	12-9-8-5
CTRL-N	16	12-9-8-6	12-9-8-6
CTRL-O	17	12-9-8-7	12-9-8-7
CTRL-P	20	12-11-9-8-1	12-11-9-8-1
CTRL-Q	21	11-9-1	11-9-1
CTRL-R	22	11-9-2	11-9-2
CTRL-S	23	11-9-3	11-9-3
CTRL-T	24	9-8-4	9-8-4
CTRL-U	25	9-8-5	9-8-5
CTRL-V	26	9-2	9-2
CTRL-W	27	0-96	0-9-6
CTRL-X	30	11-9-8	11-9-8
CTRL-Y	31	11-9-8-1	11-9-8-1
CTRL-Z	32	9-8-7	9-8-7
ESCAPE	33	0-9-7	0-9-7
CTRL-\	34	11-9-8-4	11-9-8-4
CTRL-]	35	11-9-8-5	11-9-8-5
CTRL-^	36	11-9-8-6	11-9-8-6
CTRL-	37	11-9-8-7	11-9-8-7
SPACE	40		
!	41	12-8-7	12-8-7
′	42	8-7	0-8-5
#	43	8-3	0-8-6
$	44	11-8-3	11-8-3
%	45	0-8-4	0-8-7
&	46	12	11-8-7
′	47	8-5	8-6

CARD CODES

(	50	12-8-5	0-8-4
)	51	11-8-5	12-8-4
*	52	11-8-4	11-8-4
+	53	12-8-6	12
,	54	0-8-3	0-8-3
-	55	11	11
.	56	12-8-3	12-8-3
/	57	0-1	0-1
0	60	0	0
1	61	1	1
2	62	2	2
3	63	3	3
4	64	4	4
5	65	5	5
6	66	6	6
7	67	7	7
8	70	8	8
9	71	9	9
:	72	8-2	11-8-2/11-0
;	73	11-8-6	0-8-2
<	74	12-8-4	12-8-6
=	75	8-6	8-3
>	76	0-8-6	11-8-6
?	77	0-8-7	12-8-2/12-0
	100	8-4	8-4
A	101	12-1	12-1
B	102	12-2	12-2
C	103	12-3	12-3
D	104	12-4	12-4
E	105	12-5	12-5
F	106	12-6	12-6
G	107	12-7	12-7
H	110	12-8	12-8
I	111	12-9	12-9
J	112	11-1	11-1
K	113	11-2	11-2
L	114	11-3	11-3
M	115	11-4	11-4
N	116	11-5	11-5
O	117	11-6	11-6
P	120	11-7	11-7
Q	121	11-8	11-8
R	122	11-9	11-9
S	123	0-2	0-2
T	124	0-3	0-3
U	125	0-4	0-4
V	126	0-5	0-5
W	127	0-6	0-6
X	130	0-7	0-7
Y	131	0-8	0-8
Z	132	0-9	0-9
[	133	12-8-2	11-8-5
\	134	0-8-2	8-7
]	135	11-8-2	12-8-5
^	136	11-8-7	8-5
-	137	0-8-5	8-2
,	140	8-1	8-1
a	141	12-0-1	12-0-1
b	142	12-0-2	12-0-2
c	143	12-0-3	12-0-3
d	144	12-0-4	12-0-4
e	145	12-0-5	12-0-5
f	146	12-0-6	12-0-6
g	147	12-0-7	12-0-7

CARD CODES

h	150	12-0-8	12-0-8
i	151	12-0-9	12-0-9
j	152	12-11-1	12-11-1
k	153	12-11-2	12-11-2
l	154	12-11-3	12-11-3
m	155	12-11-4	12-11-4
n	156	12-11-5	12-11-5
o	157	12-11-6	12-11-6
p	160	12-11-7	12-11-7
q	161	12-11-8	12-11-8
r	162	12-11-9	12-11-9
s	163	11-0-2	11-0-2
t	164	11-0-3	11-0-3
u	165	11-0-4	11-0-4
v	166	11-0-5	11-0-5
w	167	11-0-6	11-0-6
x	170	11-0-7	11-0-7
y	171	11-0-8	11-0-8
z	172	11-0-9	11-0-9
	173	12-0	12-0
	174	12-11	12-11
	175	11-0	11-0
	176	11-0-1	11-0-1
DEL	177	12-9-7	12-9-7

APPENDIX D

BATCH COMPONENTS

The following sections describe the batch-system components.

The term 'spooling' is mentioned several times in the descriptions. Spooling is the process whereby data goes to a holding area before being sent to the intended destination. Section D.7, 'LPTSPL, SPROUT' discusses spooling as it relates to the slow-speed output devices.

D.1 CDRIVE

CDRIVE reads cards from the physical card reader and places the information on disk. CDRIVE provides the facility for reading ASCII and image cards. (Refer to Appendix C for a table of card codes.)

D.2 BATCON

BATCON, the BATch CONtroller, is the heart of the batch system. It initiates and controls all batch jobs. It accomplishes this by:

- o Reading batch control files, created by users or by SPRINT.
- o Passing data and system program commands back to the job.

BATCON records its processing of the control file and the job in a log file in your disk area.

BATCON is generally synonymous with the batch system; often the term "batch processing" refers to BATCON's operations.

D.3 SPRINT

SPRINT is the Spooling PRocessor for INpuT; it performs the following functions.

1. It reads a sequential input stream that either CDRIVE has processed or that you have created on disk. (See the description of the /READER switch in Appendix B.)
2. It separates the input by placing it in files according to commands on the SPRINT control cards contained in the input stream.

3. It creates the job's log file and enters a report of its processing.

4. It sends the job to the batch input queue (if the job requires it) for subsequent processing by BATCON.

Chapter 5 discusses SPRINT more thoroughly.

D.4 EXEC (TOPS-20)

EXEC is the TOPS-20 command processor; it allows you access to the system queues. Through such EXEC commands as SUBMIT and MODIFY, you can enter a job into the batch input queue and modify the parameters for a job entered into a queue.

See Chapter 3, JOB CONTROL, for a description of the batch system-related EXEC commands.

D.5 GLXLIB

GLXLIB is a library of common GALAXY routines. All of the batch-system components except EXEC, MOUNTR, and QUEUE call upon GLXLIB's routines.

D.6 QUASAR

QUASAR, the queue manager, is the program that builds and maintains the system queues of tasks to be processed by BATCON and the output spoolers. When you submit a batch job from the terminal, or when SPRINT completes processing a batch job in card image, QUASAR makes an entry in the batch input queue. QUASAR schedules the jobs in the batch input queue to be run by BATCON.

Scheduling consists of computing and dynamically revising priorities for the job, according to the job's parameters and the priorities established by the system. While the job is running, the queue entry is flagged to show it is in use. When the system logs the job out, it usually makes an output queue entry and deletes the entry in the input queue. QUASAR schedules jobs in the output queues to be processed by LPTSPL and SPROUT, the output spoolers. The job's output queue entry is deleted only when the output is completely finished.

D.7 LPTSPL, SPROUT

LPTSPL and SPROUT, the output spoolers, are programs that drive the four slow-speed output devices - line printer, card punch, paper-tape punch, and plotter. (Your system may not have all of these devices.) These programs process a queue (or list) of requests that QUASAR maintains. Instead of sending output directly to a slow-speed device, the system sends output, requested by you or a program, to the disk, and the appropriate output spooler later transfers the information to the slow-speed output device.

One advantage of this automatic procedure is that you need not know or be concerned if an output device is unavailable (for example, because another program is using it); your program simply sends its output to the disk. Another advantage is that since the disk is a high-speed device, its acceptance of output intended for slow-speed devices expedites a program's execution. That is, a program can avoid the delay associated with writing to a slow-speed device.

LPTSPL can also drive printers connected to Local Area Transport (LAT) servers.

On TOPS-20 systems, LPTSPL can spool print requests to remote nodes in a Common File System (CFS) cluster and to VMS nodes.

D.8 OPR, ORION

OPR parses commands from the operator and sends them to ORION. ORION routes these commands to the appropriate component and receives all messages intended for the operator. These messages are in turn relayed to OPR.

D.9 QUEUE (TOPS-10)

The QUEUE component is the TOPS-10 user interface to the system queues. With the QUEUE command or any of the queue-class commands (SUBMIT, PRINT, CPUNCH, TPUNCH, PLOT), you can enter jobs into the batch input queue or the output queues. You can also modify parameters for jobs entered into these queues and cancel the running of these jobs.

Chapter 3, JOB CONTROL, discusses the QUEUE command and the queue-class commands for the batch input and output queues.

D.10 PULSAR (TOPS-10)

PULSAR is responsible for 'recognizing' newly-mounted disk packs and magnetic tapes by reading the home blocks (for disks) or the volume labels (for tapes) either automatically or upon operator command. Home blocks and volume labels contain identifying information for disks and tape reels. PULSAR also writes tape labels. Refer to the TOPS-10 Tape Processing Manual for additional information on PULSAR.

D.11 MOUNTR (TOPS-20)

MOUNTR controls mountable devices (disk and tape drives) on TOPS-20 systems. It permits and facilitates user access to these devices and performs Automatic Volume Recognition for labelled tapes. Refer to the TOPS-20 Tape Processing Manual for additional information on MOUNTR.

D.12 NEBULA

On TOPS-20 systems, NEBULA is the CFS cluster GALAXY message router. It routes operator messages to remote nodes in the cluster and receives responses from those nodes.

On TOPS-10 systems, NEBULA is the output spooler for the Distributed Queue Service (DQS). It handles requests for print jobs destined for printers connected to remote systems.

D.13 CATLOG (TOPS-10)

CATLOG, the System Catalog Manager, maintains a database of DECtapes, ANSI-labelled magtape volume sets, and disk structures. QUASAR consults the database when processing user mount requests. Management of the database is performed through commands to OPR.

INDEX

TOPS-10/TOPS-20
Batch Reference Manual
AA-H374B-TK

READER'S COMMENTS

Your comments and suggestions help us to improve the quality of our publications.

For which tasks did you use this manual? (Circle your responses.)

(a) Installation	(c) Maintenance	(e) Training
(b) Operation/use	(d) Programming	(f) Other (Please specify.) ____________

Did the manual meet your needs? Yes ☐ No ☐ Why? ____________

____

Please rate the manual in the following categories. (Circle your responses.)

	Excellent	Good	Fair	Poor	Unacceptable
Accuracy (product works as described)	5	4	3	2	1
Clarity (easy to understand)	5	4	3	2	1
Completeness (enough information)	5	4	3	2	1
Organization (structure of subject matter)	5	4	3	2	1
Table of Contents, Index (ability to find topic)	5	4	3	2	1
Illustrations, examples (useful)	5	4	3	2	1
Overall ease of use	5	4	3	2	1
Page Layout (easy to find information)	5	4	3	2	1
Print Quality (easy to read)	5	4	3	2	1

What things did you like *most* about this manual? ____________

____

____

What things did you like *least* about this manual? ____________

____

____

Please list and describe any errors you found in the manual.

Page	Description/Location of Error

Additional comments or suggestions for improving this manual: ____________

____

____

Name ____________	Job Title ____________
Street ____________	Company ____________
City ____________	Department ____________
State/Country ____________	Telephone Number ____________
Postal (ZIP) Code ____________	Date ____________

Fold Here and Tape

Affix
Stamp
Here

DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
200 FOREST STREET MRO1-3/L12
MARLBOROUGH, MA 01752-9101

Fold Here