DECSYSTEM

BATCH
Operator's Guide

Order No. DEC-20-OBOGA-A-D

20

digital

BATCH
Operator's Guide

Order No. DEC-20-OBOGA-A-D

digital equipment corporation · maynard. massachusetts

First Printing, February 1976

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECsystem-20	TYPESET-11

CONTENTS

CONTENTS (Cont.)

PREFACE

The *DECsystem-20 Batch Operator's Guide* is intended primarily for computer console operators. It tells how to run the programs that comprise the Batch system (sometimes referred to as the GALAXY system) and includes commands, error recovery procedures, and helpful hints. Among these hints are practical examples of the use of various commands, descriptions of procedures in a typical installation, and other suggestions.

Chapter 9 is intended primarily for the systems programming staff and/or the operations manager. Chapter 9 contains a list of "stop codes" with their explanations and suggestions for action to be taken if the system "crashes".

The readers of this manual should be familiar with the operation of the DECsystem-20 hardware and operating system. Operators and systems programmers should have access to a complete set of the DECsystem-20 documentation.

The information in this document reflects the software of release 1A of the GALAXY Batch system. This release includes:

PTYCON, version 1
BATCON, version 101
LPTSPL, version 101
SPRINT, version 2B
QUASAR, version 1A
QUENCH, version 1
DECsystem-20 Operating System, release 1

REFERENCES

DECsystem-20 Operator's Guide (DEC-20-OTPGA-A-D)
DECsystem-20 User's Guide (DEC-20-OUGAA-A-D)
DECsystem-20 Monitor Calls Reference Manual (DEC-20-OMRMA-A-D)
DECsystem-20 Batch Reference Manual (DEC-20-OBOGA-A-D)
DECsystem-20 EDIT User's Guide (DEC-20-UEUMA-A-D)

CHAPTER 1
INTRODUCTION TO THE BATCH SYSTEM

The Batch system operates under the control of the DECsystem-20 Operating System and increases the efficiency of the system by processing jobs that do not require human interaction. The types of jobs best suited for Batch are: jobs that are large and long-running, jobs that require large amounts of data, jobs that are frequently run for production, and jobs that require little or no interaction with the user.

Batch System Components
QUASAR, the system queue manager, is the heart of the Batch system. It is responsible for scheduling all the jobs entered into and processed from the system queues by the various programs in the Batch system. These programs are: the Batch Controller (BATCON), the input spooling processor (SPRINT), the line-printer spooler (LPTSPL), and the system queuing program (QUENCH).

This manual describes how to run only the Batch subsystem; but not the entire operating system or the computer itself. (For information about these subjects see the operator's guide and monitor installation guide for your system.)

Batch and the Operator
The Batch software handles many tasks for you (e.g., QUASAR selects which jobs to process and when, it decides when you should change forms in the line printer, and determines what type of forms to mount). However, there are still many operations for you to perform in order to keep the Batch system running efficiently. This is particularly important since the user is absent and dependent on you to see that his jobs are completed.

In the Batch operating environment, your responsibilities are to:

1. set various system parameters,
2. adjust the parameters if it becomes necessary,
3. service requests from jobs (e.g., tape mounts and dismounts),
4. take care of the peripheral units such as the line printers, and
5. in an installation where a substantial number of Batch jobs are submitted on cards, you are probably responsible for getting the cards read through the card reader.

In addition to the above tasks, you should periodically examine the queues and be able to modify the requests if necessary.

Error Recovery
Several procedures for recovering from the most common types of errors are described in this manual. In some instances the steps you should take are listed and in others we indicate that you should consult the system administrator before continuing. In each case, the procedure is intended simply as a guideline; the action you take is dependent on the practices at your installation.

Automatic System Startup
"Automatic system startup" is a phrase frequently used throughout this manual. This term refers to a mechanism that starts all the system's operator jobs automatically.[1] After you or the systems programmer answer the startup

[1] See Chapter 3 of the DECsystem-20 Operator's Guide for more information about the startup questions.

option questions, SYSJOB is automatically run. It reads a command file which starts up various operator jobs. This command file, <SYSTEM> SYSJOB.RUN, is then output. An example is reproduced below for your information.

```
RUN SYS:CHKPNT
RUN SYS:INFO
RUN SYS:MAILER
RUN SYS:QUASAR
JOB 0 /LOG OPERATOR XX 220100
ENA
↑ESET LOGINS ANY
↑ESEND * SYSTEM IN OPERATION
↑ESET OPERATOR
PTYCON
GET <SYSTEM>PTYCON.ATO
/
```

Note that the last command in the above SYSJOB.RUN file was GET<SYSTEM>PTYCON.ATO. This command causes PTYCON (see Chapter 2) to read a command file, which in turn starts up the jobs that are normally run under PTYCON.

Next, SYSJOB, running under job 0, outputs a record of the processing done by the operator job which it started. It precedes each line with SJ 0. The processing of the PTYCON.ATO file is also output. For example,

```
SJ  0:
SJ  0:  V 1.02.36, P3 SYSTEM, 27-JAN-76, TOPS-20 1(100)
SJ  0: @LOG OPERATOR 220100
SJ  0:  JOB 1 ON TTY102 29-JAN-76 09:11

[SYSTEM IN OPERATION]
SJ  0: @ENA
SJ  0: $SET LOGINS ANY
SJ  0: $SEND * SYSTEM IN OPERATION
SJ  0: $SET OPERATOR
SJ  0: $PTYCON
SJ  0: PTYCON> GET <SYSTEM>PTYCON.ATO
SJ  0:
SJ  0: PTYCON> SILENCE
SJ  0: PTYCON.LOG.1PTYCON> B-START
SJ  0: PTYCON>
SJ  0: **** L(0) 09:12:30 ****
SJ  0:  JOB 5 ON TTY103 29-JAN-76 09:12
SJ  0: @ENA
SJ  0: $LPTSPL
SJ  0: /
SJ  0: **** B(1) 09:12:40 ****
SJ  0:  JOB 6 ON TTY104 29-JAN-76 09:12
SJ  0: @ENA
SJ  0: $BATCON
SJ  0: /START
SJ  0: !
SJ  0: PTYCON> L-START
SJ  0: PTYCON>
SJ  0: **** L(0) 09:12:54 ****
SJ  Ø: START PLPTØ=LPTØ
SJ  Ø: !
SJ  Ø: PTYCON> WHAT ALL
SJ  Ø: L(Ø)     5        OPERATOR LPTSPL   TI      ØØ:ØØ:Ø1
SJ  Ø: B(1)     6        OPERATOR BATCON   TI      ØØ:ØØ:Ø1
SJ  Ø: P(2)     4        OPERATOR OPLEAS   RN      ØØ:ØØ:ØØ
SJ  Ø: OPR (3)  3        OPERATOR CHECKD   RN      ØØ:ØØ:11
SJ  Ø: PTYCON>
SJ  Ø: **** OPR(3) Ø9:13:51 ****
SJ  Ø: @
```

Once PTYCON has completed processing the PTYCON.ATO file, the system is ready to process both timesharing and Batch jobs.

CHAPTER 2 PTYCON

Function
As you read this manual, you will notice that there are several programs that the operator must run at one time; the PTYCON program provides a convenient method for you to control several jobs simultaneously from a single terminal. If you did not run PTYCON (or a similar program), it would be necessary for you to either continually attach to and detach from the various jobs you are running or use a separate terminal for each job and attempt to monitor them all as frequently as possible. With PTYCON, control for multiple jobs, called subjobs, is concentrated at one terminal. They are called subjobs because they are run under the control of another job (i.e., PTYCON) rather than being directly controlled by a user sitting at a terminal.

Method of Operation
The DECsystem-20 considers all jobs to be timesharing jobs; thus, each job must be associated with a timesharing terminal (TTY). In some situations, however, it is inconvenient to require a timesharing terminal to be associated with every job on the system (e.g., where many jobs are needed for a single application; the application in the case of PTYCON being the operation of the entire system). There is a method whereby a single job running from a single timesharing terminal can control many subjobs simultaneously. This is accomplished by implementing a "device" called a pseudo-terminal (PTY) which in effect is a software simulation of a terminal. The controlling job can send information to a PTY and receive information from the PTY. This mechanism is used both in PTYCON and in BATCON.

PTYCON starts running subjobs over pseudo-terminals (PTYs) and controls each subjob by sending appropriate information (i.e., the commands and/or data, etc., that you give it) over the PTY controlling that subjob. You can run as many subjobs as there are PTYs on the system up to a maximum of 24. PTYCON allows you to remain at one terminal and still control multiple jobs over multiple PTYs.

Operator's Responsibilities
Your responsibilities are to supervise the PTYCON subjobs that you are running, handle all requests initiated by the users, including tape and disk mounts and dismounts, load and remove card decks, line-printer listings, etc., and set various parameters to ensure that the system is running smoothly and efficiently.

Startup
PTYCON is normally started automatically on the operator's terminal when the system is brought up. At this time PTYCON reads and processes the file <SYSTEM>PTYCON.ATO (usually referred to as the "Auto File"). This processing normally consists of bringing up SPRINT, LPTSPL, BATCON, and any other operator programs that you need, such as OPLEAS. A sample of <SYSTEM>PTYCON.ATO is shown below.

```
SILENCE                          (SILENCE output to CTY)
LOG                              (Create LOG file PTYCON.LOG)
DEFINE ^$L                       (DEFINE subjob 0 as L)
DEFINE ^$B                       (DEFINE subjob 1 as B)
DEFINE ^$P                       (DEFINE subjob 2 as P)
DEFINE ^$OPR                     (DEFINE subjob 3 as OPR)
CONN OPR                         (CONNECT to subjob OPR)
LOG OPERATOR FOO 220100          (LOGIN)
^X                               (Return to PTYCON)
OPR-RUN <SYSTEM>CHECKD           (Run CHECKED under OPR)
```

```
OPR-DIRECTORY                    (command to CHECKD)
CONN P                           (CONNECT to subjob P)
LOG OPERATOR FOO 220100          (LOGIN)
OPLEAS ^$                        (Run OPLEAS)
^X                               (Return to PTYCON)
CONN L                           (CONNECT to subjob L)
LOG OPERATOR FOO 220100          (LOGIN)
ENA                              (ENABLE)
LPTSPL                           (Run LPTSPL)
^X                               (Return to PTYCON)
CONN B                           (CONNECT to subjob B)
LOG OPERATOR FOO 220100          (LOGIN)
ENA                              (ENABLE)
BATCON                           (Run BATCON)
^X                               (Return to PTYCON)
NO SILENCE                       (Allow output to CTY)
B-START                          (Command to subjob B running BATCON)
L-FREEZE                         (Command to subjob L running LPTSPL)
L-MLIMIT 500                     (Command to subjob L running LPTSPL)
L-START                          (Command to subjob L running LPTSPL)
WHAT ALL                         (PTYCON command to output status of all subjobs)
@
```

When PTYCON is started this way, it is normally left detached and you must attach to it before you can use it. First run SYSTAT to determine PTYCON's job number and then attach to PTYCON. (Since PTYCON is usually job number 1, you can just do a SYSTAT of job 1 to confirm its number.) For example,

```
@SYS 1
    1 102 PTYCON OPERATOR
@ATT OPERATOR x 1                    (the x stands for your password)
  [ATTACHED TO TTY 102, CONFIRM]
^ C
$ START
PTYCON
**** OPR (4) 13:15:37
@
PTYCON
```

If job 1 is not running PTYCON, run SYSTAT for all the jobs to see which one is running PTYCON, and then ATTACH to it. If the output from SYSTAT shows that PTYCON is not running, then you must start it yourself. To do this, type PTYCON at system command level.

```
@PTYCON
```

When PTYCON responds with its prompt, type the command to run the auto file.

```
PTYCON> GET <SYSTEM>PTYCON.ATO
```

If you do not use the auto file, and/or you wish to start up additional subjobs, you must create them and log them in yourself. This process is described below, in the section "creating subjobs".

NOTE

If you forget to ATTACH to PTYCON, you will not lose any output from subjobs since PTYCON can still output on your terminal through SYSJOB.

Communicating with PTYCON

Once you have PTYCON running and you are at PTYCON command level (i.e., PTYCON has typed its prompt, "PTYCON>"), it is ready to receive commands from you. In communicating with PTYCON you can use the standard TOPS-20 features, including CTRL/C, CTRL/T, CTRL/R, CTRL/U, recognition input (except for single-line subjob communication), rubout, EDIT, languages, and system commands.

There are several modes by which you can communicate with PTYCON.

1. You can issue PTYCON commands (explained at the end of the chapter) directly to PTYCON,
2. You can issue system commands to subjobs through single-line commands,
3. You can directly connect to a subjob, or
4. When there are no subjobs available, you can temporarily leave PTYCON and go to system command level by issuing the PTYCON command PUSH.

Creating Subjobs

There are two methods of creating PTYCON subjobs. You can use either the CONNECT or the DEFINE command. When you issue either command, followed by a number which has not yet been assigned to a subjob, you create a subjob with that number. (Valid subjobs range from 0 to 23.)

When you use the DEFINE command, you can type ESCape instead of a number and PTYCON will create a subjob and give it the next free subjob number. In addition, the DEFINE command allows you to associate a name with the subjob.

You must log in each subjob you create whether you use CONNECT or DEFINE. If you log in subjobs under the same name that PTYCON is logged in under, you can use a fictitious password. Two examples of creating and logging in subjobs are given below.

1. Creating and logging in a subjob with CONNECT

```
PTYCON> CO 3
[CONNECTED TO SUBJOB 3]

 V 1.02.36, P3 SUPPORT TEST SYSTEM, 27-JAN-76. TOPS-20 1(100)
@LOG OPERATOR X  10300   (the x is your fictitious password)
 JOB 26 ON TTY111 29-JAN-76 12:41
@
```

At this point, you can give commands directly to the job. When you want to return to PTYCON command level, type CTRL/X. (Hold down the control key and type X.)

2. Creating a subjob with DEFINE, giving it the name OPR, and logging it in.

```
PTYCON> DEF 3 OPR
PTYCON> OPR-LOG OPERATOR X 10300   (the x is your fictitious password)
PTYCON>
**** OPR(3) 12:43:18 ****


 V 1.02.36, P3 SUPPORT TEST SYSTEM, 27-JAN-76. TOPS-20 1(100)
@LOG OPERATOR  10300
```

```
 JOB 26 ON TTY111 29-JAN-76 12:43
@
PTYCON>
```

Subjob Communication
Once a subjob exists, you can communicate with it by giving it single-line commands or by being directly connected to it.

Single-line Commands
The format of the single-line command is the subjob number or the name, followed by a hyphen and then the command or data to be sent to the subjob.

Single-line commands should be issued only after the PTYCON prompt.

You also must observe what state the subjob is in; i.e., whether it is awaiting program input or whether it is at system command level. The last character output by the subjob indicates its current state. If the character is "@" or "$", the subjob is at system command level and if the character is "*", it is awaiting program input. However, regardless of the state it is in, you always get returned to PTYCON command level. For example,

```
PTYCON> OPR-FILCOM
PTYCON>
**** OPR(3) 13:13:16 ****
FILCOM

*
PTYCON> OPR-=ABC,DEF
PTYCON>
**** OPR(3) 13:13:30 ****
=ABC,DEF

No differences encountered

*
PTYCON> OPR-^C
PTYCON>
**** OPR(3) 13:13:40 ****
^C
@
@
PTYCON>
```

Direct Connection
To communicate directly to a subjob, use the CONNECT command. First type CONNECT followed by a subjob number. When the system responds with the message that you are connected, you can type to the connected subjob just as if it were a normal timesharing job that did not go through PTYCON. When you have finished giving commands to the subjob, type CTRL/X to return to PTYCON command level. Note that the example below accomplishes the same task as the single-line command example in the previous section.

```
PTYCON> CO 3
[CONNECTED TO SUBJOB 3]
FILCOM

*=ABC,DEF
```

```
No differences encountered

*^C↵
@^X↵
PTYCON>
```

Note that when giving single-line commands to several subjobs, the output from these jobs (with its identifying header) will be intermixed on the typescript unless you prevent this from happening. If any non-connected subjob is likely to produce a lot of output on your terminal and you would prefer not to see it (because you are involved with other subjobs), you should suppress the output that you do not want to see. (Refer to the ACCEPT, DISCARD, LOG, and REFUSE commands for information about the various methods of suppressing output.)

The disadvantages of the single-line communication method are the amount of extra typing required, the inability to use recognition input, and the fact that what you type is different from what you type to a normal timesharing job.

The advantage of the CONNECT command is the ability to communicate directly (PTYCON being transparent) each time you issue a command to a subjob.

The demands of the individual subjobs will determine which method you choose to communicate with them at a particular time.

Helpful Hints

1. Open subjobs – You will notice that in the example of the PTYCON.ATO file shown previously, a job defined as OPR was logged in. This is generally an open job, under which you can run various tasks that might be required during the day. Creating and logging in the job when PTYCON is started simply saves time later and guarantees an available subjob for when you might need it. If you do not wish to tie up a PTY with the OPR job, simply log it out or edit the auto file to omit logging it in.
2. Naming Subjobs – The DEFINE command allows you to use names of up to five alphanumeric characters as mnemonics for subjobs. However, it is recommended that you use short names to cut down on your typing. A list of the subjob names commonly used is given here for your information.

   B for BATCON
   L for LPTSPL
   P for OPLEAS
   SP for SPRINT
   OPR for the open job.

Commands

The commands to PTYCON, with their guide words in parentheses are:

ACCEPT (OUTPUT FROM SUBJOBS)
BELL (WHEN OUTPUT WAITING)
CONNECT (TO SUBJOB)
DEFINE (SUBJOB #)
DISCARD (OUTPUT FROM SUBJOB)
EXIT (FROM PTYCON)
GET (COMMANDS FROM FILE)
HELP MESSAGE
KILL (SUBJOB)
LOG (OUTPUT TO FILE)
PUSH (EXEC LEVEL)
REFUSE (OUTPUT FROM SUBJOBS)
SILENCE (ALL OUTPUT TO TERMINAL)
WHAT (IS STATE OF SUBJOB)

The ESCAPE CHARACTER TO RETURN TO COMMAND LEVEL IS: ^X

These commands are discussed in detail below.

Command	Explanation
ACCEPT *n,. . . ,n*	Allows you to receive output on your terminal from the specified subjobs. This is the normal mode. When ACCEPT is issued after a NO ACCEPT or REFUSE command, it causes any available output from the specified subjobs to be typed out on your terminal immediately and it continues to allow information from the subjobs to be output on your terminal as it becomes available. If you omit *n,* ALL is assumed.
NO ACCEPT *n,. . . ,n*	Suppresses the output from the specified subjobs on your terminal. NO ACCEPT is equivalent to REFUSE. If you omit *n,* ALL is assumed.
BELL	Causes the terminal bell to ring or beep every 10 seconds when a subjob with REFUSE in effect has output waiting. BELL is the normal mode. Thus, you need to issue this command only when you want to cancel a NO BELL command that you previously issued.
NO BELL	Suppresses the ring or beep of the terminal bell that indicates a REFUSEd subjob has output. Consequently, when NO BELL is in effect, you are not warned that there is output waiting.
CONNECT *n*	Connects your terminal to subjob *n.* The effect of this command is that the connected subjob will appear to be a normal timesharing job that is not running under PTYCON. All commands that you issue to subjob *n* are passed directly to it. As soon as you connect to a subjob, you receive a message confirming that the connection was successful. Then, any output that was being buffered for that subjob by the REFUSE or NO ACCEPT command is output on your terminal. If you omit the subjob number, PTYCON connects your terminal either to the last connected subjob or to the last subjob defined, whichever was done most recently. If you specify a valid subjob number that you have not previously defined, a new subjob will be created for you. However, you may not specify a subjob name that has not previously been defined.
CTRL/X	Returns a connected subjob to PTYCON command level, i.e., it disCONNECTs a CONNECTed subjob.
DEFINE *n name*	Defines, i.e., creates, a new subjob *n* with an optional name or, associates the already existing subjob *n* with a name. The name may consist of from 1-5 alphanumeric characters. If you type ESCape for the subjob number, PTYCON assigns the next free subjob number. You may define a new name for a subjob that already has a name associated with it. If you define a subjob with a name already given to another subjob, you get the message %NAME ALREADY IN USE, REASSIGNED TO THIS SUBJOB

Command	Explanation
DEFINE *n name* (cont.)	The name is given to the newly defined subjob and the association between the name and the old subjob is broken. Also, defined names take precedence over subjob numbers in cases where you define a subjob to be another number (e.g., DEFINE (SUBJOB #) 0 (AS) 1). Therefore, whenever you use 1 in a PTYCON command, it refers only to subjob 0 which is named 1.
DISCARD *n, . . . ,n*	Eliminates output from the specified subjob (unless you are connected to it). The output is in effect, thrown away as far as your terminal is concerned. However, if you have specified the PYTCON LOG command, then the output does get written into the LOG file. The DISCARD command differs from REFUSE in that REFUSE saves the output for ACCEPTance on your terminal at a later time. However, DISCARD overrides the REFUSE command. That is, if you DISCARD a subjob's output while REFUSE is in effect, none of the output buffered by REFUSE prior to the DISCARD command is typed on your terminal. However, any accumulated output is entered in the LOG file (provided that the LOG command is in effect). You must specify a subjob number; *n* may not be omitted. The word ALL may be substituted for *n*.
NO DISCARD *n, . . . ,n*	Allows output to be entered on your terminal. This is the normal mode.
EXIT	Exits from PTYCON and returns you to system command level. You should use EXIT only when you are finished using PTYCON. If you want to return to system command level temporarily, while subjobs are still running, use the PUSH command. If you type EXIT and subjobs are still active PTYCON gives you the message CAUTION: EXITING MAY LOG OUT THE STILL ACTIVE SUBJOBS CONFIRM: (TYPE CONTROL-X TO GET BACK TO PTYCON) At this point, you can type CTRL/X to return to PTYCON immediately. If you type carriage return, you exit from PTYCON and may lose subjobs. **NOTE** If by accident you do type EXIT and then respond to CONFIRM with a carriage return, you can recover if you *immediately* type CONTINUE. This will return you to PTYCON and preserve any subjobs you had running. However, you must do this within five minutes or the subjobs will be logged out.
GET *filespec*	Reads and executes the commands in the specified file. If you omit the filespec, the file PTYCON.ATO in your directory is assumed. Certain characters in the command file must be typed differently from the way you would enter them directly on a timesharing terminal. Do not use the CTRL key to type control characters in the command file. CTRL*x* (where *x* represents an alphabetic character) must be typed as ^*x* (circumflex-*x*). ESCape must be typed as ^$ (circumflex-dollar sign). A circumflex must be typed as ^^ (two circumflexes). A sample PTYCON.ATO file is reproduced at the beginning of this Chapter.

Command	Explanation
HELP	Types out a list of the PTYCON commands and their associated guide words.
KILL *n*, . . . ,*n*	Kills the specified subjobs (i.e., logs them out) and deassigns the subjob numbers. This command will work only if the specified subjobs are logged in under the same user name as PTYCON or, if they are not, then PTYCON must be running with OPERATOR or WHEEL privileges enabled. If one of these conditions is not met, you will receive the message, COULD NOT KILL SUBJOB *n* In order to log jobs out in such instances you must CONNECT to the subjob, LOGOUT, return to PTYCON, and issue the KILL command. The last step is necessary to deassign a subjob number. Simply logging out a subjob does not deassign the number. **NOTE** "ALL" may be substituted for *n*. You must type ALL in its entirety.
LOG *filespec*	Causes all interactions with PTYCON to be recorded in the specified file. If you omit the filespec, PTYCON.LOG is assumed. If you omit only the file type, .LOG is assumed. Log files of the same name are appended to and not superseded. (Refer to the *DECsystem-20 Operator's Guide* for more information about the contents of this file.)
NOLOG	Stops output to the LOG file. This is the only PTYCON command that stops output to the log file. NOLOG is the normal mode.
PUSH	Exits to system command level without affecting the subjobs. This command is useful when you want to do something at system command level temporarily; you do not have to use another subjob. To return to PTYCON command level when you have finished your task, type POP. When you do a PUSH command, output from the subjobs is suspended. Also, as with REFUSE, PUSH causes the terminal bell to ring or beep (if the BELL command is in effect) when output is waiting. When you return to PTYCON all waiting output is typed on your terminal.
REFUSE *n*, . . . ,*n*	Inhibits output from the specified unconnected subjob to your terminal. However, any output that exists will be buffered so you can access it later via the ACCEPT or NO REFUSE command, or by CONNECTing to a subjob. At the same time you access the output, it also is written in the LOG file (if it exists). The terminal bell will sound when a REFUSEd subjob has output waiting provided that the BELL command is in effect. The word "ALL" may be substituted for *n*.

Command	Explanation
NO REFUSE *n*, . . . ,*n*	Reverses the effect of the REFUSE command. NO REFUSE is the normal mode and is equivalent to ACCEPT.
SILENCE	Discards any output that normally would be output on your terminal while PTYCON processes the "auto" file (i.e., the file specified in the GET command). Output will still get printed in the log file if the LOG command is in effect. The SILENCE command is effective only when it is issued in an "auto" file. If you type it when you are running subjobs, PTYCON ignores it. **NOTE** If you have a SILENCE command in an "auto" file, it is recommended that you also have a NO SILENCE. If you do not, you will not see the PTYCON prompt indicating that it has processed the file and is ready to accept commands.
NO SILENCE	Allows output on your terminal while PTYCON is processing the "auto" file (i.e., the file specified in a GET command). NO SILENCE is the normal mode.
WHAT *n*	Types out the status of the specified subjob. If you omit *n*, PTYCON assumes ALL. The status information includes the subjob name (if any), the subjob number, the job number, the user logged in under the subjob, the program running under the subjob, and the subjob's current state. (See Chapter 3 of the *DECsystem-20 Operator's Guide* for details about output from the WHAT command.

CHAPTER 3
BATCON

Function
The Batch Controller, BATCON, is a system program that initiates and controls the processing of Batch jobs. QUASAR, the Batch system queue manager, selects jobs from the Batch input queue (INP:) and gives them to BATCON for processing.

BATCON is capable of handling up to 14 jobs (called subjobs) at one time. It communicates with the subjobs over pseudo-teletypes (PTY's) and in this manner supplies information from each of the control files to the appropriate jobs and obtains information from the job and enters it into the LOG file.

BATCON receives Batch jobs in the form of user disk files (Batch control files) containing system commands, program commands, user programs, data normally entered by a timesharing user from a terminal, and special Batch commands. (For information about the actual format of these control files, refer to the *DECsystem-20 Batch Reference Manual.*)

Operator's Responsibilities
Your primary responsibilities are to set and adjust parameters in BATCON to ensure optimum system performance, and to respond to requests issued by BATCON subjobs. QUASAR uses these BATCON parameters to schedule jobs for BATCON. You set these parameters both for individual jobs and for all the Batch jobs running on the system. Occasionally it is desirable to change the various limits for the values that you initially set. The intended effect of resetting the parameters is to prevent BATCON from overloading the system during peak times and to defer large or long-running jobs to the slack periods of the day. The commands that set these parameters are described in the section on commands at the end of this chapter.

Once you have set these parameters, you need do little more to manage BATCON. However, as individual jobs make operator requests, e.g., tape mounts or dismounts and PLEASE messages, you are expected to handle them. You should also check on BATCON and its subjobs periodically to ensure that all jobs are running and are not hung in a wait or CTRL/C state. The commands that display information about the various jobs and the current values of the parameters are described in the section on commands at the end of this chapter.

Under certain conditions it may become necessary for you to take over control of a BATCON subjob and inform the user of your action. The commands that accomplish these tasks are listed in the commands section at the end of this chapter.

Startup
BATCON is normally initialized automatically under PTYCON (via the auto file) at system load time and requires no action from you to start it. The PTYCON commands to start up BATCON manually are given below for use under circumstances where it is desirable to start BATCON manually.

```
@PTYCON
PTYCON> DEFINE (SUBJOB #) 0 (AS) B
PTYCON> B-LOG OPERATOR X 10300
PTYCON>
**** B(0) 10:12:43 ****
```

```
 V 1.02.36, 1031 DEVELOPMENT SYS., 30-JAN-76. TOPS-20 1(100)
@LOG OPERATOR 10300
 JOB 13 ON TTY107 10-FEB-76 10:12
@
PTYCON> B-BATCON
PTYCON>
**** B(0) 10:13:01 ****
BATCON
/
PTYCON>
```

At this point, you generally set the various parameters. For example,

```
PTYCON> B-MJOB 5
PTYCON>
**** B(0) 10:13:27 ****
MJOB 5
/
PTYCON>
```

BATCON is ready to start processing once you have started it as follows:

```
PTYCON> B-START
PTYCON>
**** B(0) 10:13:44 ****
START
/
PTYCON>
```

Command Format
The general format of a command string is:

x–n–COMMAND argument

x	The x stands for the subjob number or the defined subjob name for BATCON under PTYCON. You must precede all commands issued to BATCON with this character. See the chapter on PTYCON for more information about naming subjobs.
n	This indicates the BATCON subjob number. If you omit the n in your command string, BATCON assumes the command is intended for the last subjob you specified in a previous command. You may substitute the word ALL for n. In this case, all active subjobs are the subjects of the command. Some BATCON commands are general (i.e., do not apply to a specific subjob) and hence do not require the subjob number. In these cases you should omit both the subjob number and its associated hyphen. The format of the command then becomes x–COMMAND
COMMAND	This is one of the commands described in the commands section. Commands may be abbreviated to as few letters as are necessary to indicate a unique command.
argument	The argument varies for each command and is detailed in the individual command descriptions. It may be a number, a string of characters, a key word, or it may not be required.

NOTE
When you are CONNECTed to BATCON you must omit the "x–" part of the command string. Thus, the format is simplified to "n–COMMAND argument".

If a command requires an argument, you must specify one; otherwise, BATCON will issue an error message. The default value for an argument will be assumed only when the command itself is not given.

Helpful Hints

1. Responding to subjob requests – When a subjob requires a response from you BATCON types out the message

```
OPR–Respond BL–$
Waiting . . .
```

and rings the bell on your terminal. However, BATCON does *not* expect you to respond with "BL–$". Instead, BATCON expects a response of the form x–n–ESC↵ when you are at PTYCON command level and a response of the form n–ESC when you are CONNECTed to BATCON.

x	represents the name defined for BATCON under PTYCON (usually B) or the PTYCON subjob number.
-	is a hyphen and is required.
n	represents the BATCON subjob number.
-	is a hyphen and is required.
ESC	represents the ESCape character (which is echoed as a dollar sign).
↵	represents a carriage-return.

If you do not respond, BATCON will type out information about the job, the message "WAITING FOR RESPONSE", and then ring the terminal bell several times. This prompting is repeated at intervals until you respond.

2. Under certain exceptional conditions, there may be no LOG file for BATCON to write (e.g., after an attempt to automatically LOGOUT a job). When this is the case, any output sent from the subjob is typed (with a heading) on your terminal. Although this can sometimes be annoying, there is a good chance that the output will contain error information that you will need in diagnosing the problem that occurred.

Commands

The commands available to you fall into three categories and are grouped accordingly. The three types are:

1. Commands to set selection criteria for Batch jobs,
2. Commands to display the current parameter settings, and
3. Commands to control Batch subjobs.

Commands to Set Selection Criteria for Batch Jobs

Command	Explanation
EXIT	Ceases selection of new jobs, waits for currently active jobs to be completed, and then returns to TOPS-20 command level.
MJOB n	Limits the number of Batch jobs that can be run concurrently to n, which represents a decimal number in the range zero to the value of the JOBMAX assembly parameter.
MTIME n	Limits each individual Batch job to n seconds of CPU time, where n is a decimal number in the range 0 to 262,143 seconds. Any Batch jobs in the input queue that have estimates larger than n will not be selected for execution by BATCON. The default is no limit.
NEXT n	Selects the job with the queue sequence number n as the next job to be processed.
RESET	Stops scheduling new jobs, waits for all currently active jobs to be completed, and returns to operator command wait. At this point, all parameters are RESET to their default settings.
START	Begins Batch processing. You can also use this command to cancel the effect of a RESET or EXIT command.
TIME n	Limits the total amount of CPU time available to all Batch jobs to n seconds, where n represents a number in the range 0 to 262,143 seconds. The default is no limit.

Commands to Display the Current Parameter Settings

CURRENT	Types out the current values of the parameters set by the commands above. Also, types out the number of the current default subjob, the number of active job streams and any exceptional conditions that affect the Batch system.
HELP	Lists all the valid BATCON commands. Only the commands themselves will be listed; detailed descriptions are given in this section of the *Batch Operator's Guide*.
MONJOB	Types out Batch subjob numbers and the corresponding system job numbers.
WHAT	Types out a description of the current job, including the subjob number, the system job number, the job name, the user name, the program running, its state and the amount of CPU time used. In addition, this command displays the last line of text entered into the specified job's LOG file.

Commands to Control Batch Subjobs

EXAMINE n	Displays the next n lines of the control file of the specified subjob on your terminal. Only the lines contained in the current core buffers can be displayed.

Commands to Control Batch Subjobs (Cont.)

Command	Explanation
GO message	Continues the specified subjob that has been halted by the STOP command. The message is optional, but if present, it will be placed in the LOG file of the specified subjob and be indicated as a comment.
KILL argument!message	Cancels the specified subjob. The argument can be one of the following: 1. FLUSH – cancels without error recovery. 2. ERROR – cancels with normal user error recovery. 3. NERROR – cancels without error recovery. If the argument is not specified, KILL ERROR is assumed. The message is optional, but if present, it must be preceded by an exclamation point. It will appear as a comment in the LOG file of the specified subjob. **NOTE** If the argument to this command is missing, any message must be preceded by an exclamation point (!). If no message is present, the exclamation point is not necessary.
OPERATOR line	Sends the supplied line to the specified subjob. This command is the response to dialogue mode and has no effect unless the user has issued the Batch command @OPERATOR within the control file. Dialogue mode is the operator-program communication facility provided by BATCON (refer to the *DECsystem-20 Batch Reference Manual*).
REQUEUE hh:mm	Returns the specified subjob to the system input queue for processing later. The argument hh:mm is used to give the subjob an AFTER parameter (e.g., 1–REQUEUE 1:00 requeues subjob 1 and provides an AFTER parameter of one hour). If the argument is missing, the assembly parameter REQTIM is used. Consult the system adminstrator to determine the value of the assembly parameters. **NOTE** When you specify an AFTER time, if you type only one set of numbers BATCON assumes you mean minutes. It is only when you specify two sets of numbers, separated by a colon that BATCON assumes both hours and minutes. Some examples are given below. 1–REQUEUE 20 requeues subjob 1 and provides an AFTER parameter of 20 minutes. 11–REQUEUE 1:30 requeues subjob 11 and provides an AFTER parameter of 1 hour and 30 minutes.

Commands to Control Batch Subjobs (Cont.)

Command	Explanation
	NOTE (Cont.) 3–REQUEUE 100 requeues subjob 3 and provides an AFTER parameter of 100 minutes (i.e., 1 hour and 40 minutes).
STOP message	Halts the specified subjob (i.e., puts the job in a CTRL/C state). The job will not be resumed until you issue a GO command. The message is optional but, if present, it will appear as a comment in the LOG file of the specified subjob.
n–ESCape	Allows BATCON subjob n (which is in operator wait) to proceed. Normally, this is the response to the Batch command @PLEASE, but you can also use it for other exceptional conditions in which operator intervention is required.

CHAPTER 4
LPTSPL

Function
LPTSPL, the line-printer spooler, is a program that handles all the print requests for the system: both those submitted through the QUENCH PRINT command and spooled requests generated by user programs. QUASAR, the system queue manager, selects jobs to be printed according to a variety of parameters. You may set and dynamically change some of these parameters depending on the requirements of the system.

Operator's Responsibilities
Your responsibilities are to check the print queue periodically (see the section on QUENCH for a description of how to check the contents of the system queues) in order to determine whether or not jobs are being processed. If they are not, you must find out why and take the appropriate action. For example, a backlog of large jobs might be building up because you have set MLIMIT too low.

In addition to monitoring the queue, you must be prepared to stop runaway jobs, change the paper when necessary (the software tells you when to change the forms type and what forms to use), be sure the paper does not jam, and see that other mechanical difficulties are corrected. The commands that enable you to perform these tasks are described at the end of this chapter.

In some installations, it is also the operator's responsibility to remove listings from the printer and to distribute them.

Startup
Normally LPTSPL is started up automatically under PTYCON when the system is brought up. You can use the commands below if it becomes necessary to start the line-printer spooler manually.

```
@PTYCON↵
PTYCON DEF$ L↵                    (The dollar sign is echoed when you type ESCape.)
PTYCON CONN L↵
LOG OPERATOR X 10300↵
ENA↵
LPTSPL↵
FREEZE↵
MLIMIT 500↵
START↵
^ X
```

Error Procedures

1. Runaway Jobs – If you notice that a print job is "throwing" a lot of paper (i.e., ejecting paper extremely rapidly), and you feel that it is printing nothing of importance, then you should probably stop the job with either the KILL or SKPFILE command.
2. File Access Errors – File Access errors for a user's file can occur fairly frequently. These errors are not your fault but can happen if the user's file is in some way illegible to the system. For example, the file might have been deleted accidentally. When problems of this nature arise, LPTSPL prints a message on the user's output to inform him of the error, outputs the message, ?LPTCAF CAN'T ACCESS FILE, on your terminal, and proceeds to the next request.

Normally, there is nothing you can do to recover from File Access errors, but if they become a recurrent problem you should investigate.

Command Format
There is no special syntax for giving commands to LPTSPL. You type the commands, which can be abbreviated to as few characters as are necessary to make them unique, directly to LPTSPL. The individual command descriptions specify whether or not an argument is required.

Special Forms Handling
The special forms handler in LPTSPL provides a means by which you can automatically have various forms parameters set on the basis of forms names. In order to accomplish this, you must set up a file on SYS: called LPFORM.INI which contains a list of forms names and switch settings for those forms.

Each line in LPFORM.INI is of the form:

formname/sw_1/sw_2/sw_3 . . .

formname	is a 1- to 6-character form name. You or the system administrator should specify form names descriptive of the types of forms used at your installation.
/sw_1/sw_2/sw_3 . . .	are one or more switches available to you to describe the manner in which the forms are to be used.

Switch	Meaning
/BANNER:nn	The value of nn specifies the number of banner pages (job header pages) desired.
/CHAIN:xxx or /DRUM:xxx	The xxx represents a 1- to 6-character ASCII string that specifies the chain or drum to be used on the line printer. LPTSPL types the switch name and argument on your terminal when the forms are scheduled. If you specify both the /CHAIN and /DRUM switches for the same entry in the LPFORM.INI file, the spooler will type only the last one it encountered.
/HEADER:nn	The value of nn specifies the number of file header pages desired.
/LINES:nn	The value of nn specifies the number of lines to be printed on the banner, trailer, and header pages. The argument is used for internal calculations and affects only the banner, header, and trailer pages; not the printing of the file itself.
/NOTE:xxx	The xxx represents a note of up to 50 characters which will be typed on your terminal at the time the forms are scheduled.
/PAUSE	Causes LPTSPL to pause before each job that uses the type of form specified. This switch overrides the UNLOCK operator command.
/RIBBON:xxx	The xxx represents a 1- to 6-character ASCII string that specifies the type of ribbon to be used on the line printer. LPTSPL types the switch name and argument on your terminal at the time that the forms are scheduled.
/TAPE:xxx	The xxx represents a 1- to 6-character ASCII string which specifies which vertical forms unit control tape or carriage-control tape should be used. You will be notified by a message from LPTSPL to use that tape on the line printer.

Switch	Meaning
/TRAILER:nn	The value of nn specifies the number of job trailer pages desired.
/WHAT	Causes LPTSPL to type a short WHAT message on your terminal for each job printed. This message consists of the job name, the user name, and the job sequence number.
/WIDTH:nnn	The value of nnn is used for internal calculations to determine a width class. The width class specifies the number of characters per line to be printed on the banner, header, and trailer pages. There are three width classes. 1. Width class 1 (nnn=0–60) specifies that up to 66 characters per line will be printed on the banner, header, and trailer pages. 2. Width class 2 (nnn=61–100) specifies that up to 90 characters per line will be printed on the banner, header, and trailer pages. 3. Width class 3 (nnn=101–132) specifies that up to 129 characters per line will be printed on the banner, header, and trailer pages.

LPTSPL requests that you change the forms only if the first four[1] characters of the new form name differ from the first four characters of the old form name. This allows you to enter lines in LPFORM.INI of the form:

```
NARROW/HEADER:1/BANNER:1/TRAILER:1
NARR01/HEADER:0/BANNER:1/TRAILER:1
NARR02/HEADER:0/BANNER:0/TRAILER:0/WHAT
```

LPTSPL will schedule any of the three types of forms, NARROW, NARR01, or NARR02, without asking you to change the forms (provided that one of the specified forms is already mounted).

Forms Alignment
For information on the installation and alignment of paper on your line printers, and for a description of carriage format tapes, see the instructions that come with your particular line printer.

Helpful Hints

1. Changing MLIMIT – The MLIMIT command provides a useful mechanism for maintaining a high level of line printer throughput. By lowering the value of the command argument (which limits the number of pages for each job) during prime time, you can have LPTSPL automatically defer large jobs to the slack periods of the day. At this time you can raise the value of MLIMIT and complete the jobs that were too large to be run earlier.

   If your installation has more than one line printer and one of them is in greater demand (e.g., it prints both upper and lower case characters) than the other(s), many more people will specifically queue jobs to it and so generate throughput problems. Therefore, setting MLIMIT even lower for that particular printer will help prevent backlog.
2. Freezing Forms – QUASAR's algorithms for scheduling forms are designed to make sensible decisions about forms changes for the majority of computing environments. However, since each installation is different, these decisions may not be optimal for you. For example, it might be the practice in certain installations to have one line printer reserved specifically for a single type of forms during the very busy periods of the day or even at all times. If this is the case at your installation, you can use the FREEZE command to LPTSPL to indicate that you do not want QUASAR to schedule a different type of forms for a specific printer. The UNFREEZE command enables you to reverse the effects of the FREEZE command.

[1] The value of this parameter is determined at Batch system generation time; the default is four.

The FREEZE command is particularly advantageous when you have LPTSPL "STARTed" for a device other than a line printer. It is suggested that you group the output according to forms type to facilitate the printing when it is eventually done. For example, if you are writing requests queued to LPT1 onto a magnetic tape, you might put jobs requiring different forms types onto different tapes. When you eventually print the jobs, you will know that only each tape will require a different forms type and you will not have to change forms continually. The procedure below is one method of accomplishing this. Connect to the subjob under which LPTSPL is running and type the following:

FREEZE↵	(tells QUASAR to schedule jobs only for the type of forms currently mounted)
START↵	(starts LPTSPL processing)

When the queue is empty of requests for the frozen forms, type

STOP↵	(stops LPTSPL)
FORMS "new forms"↵	(informs LPTSPL that forms type "new forms" is mounted so that QUASAR will schedule requests only for those forms even though it does not matter which forms, if any, actually are mounted)

Change the magnetic tape in order to separate the output requiring different forms types. Then type

GO↵	(resumes processing)

and so on until all the requests have been processed. Note that you do not have to type FREEZE again. This is because once forms are frozen, subsequent forms will also be frozen provided that you do not issue a RESET command.

3. Selecting a Specific Job – The NEXT command makes it easy for you to select a specific job, from any one of the queues, to be processed next. This command is very useful in situations where you must run a certain job next but the size of the job (i.e., number of pages) is greater than the value of MLIMIT. You do not want to raise the value of MLIMIT because other large jobs might also get printed. So you should use the NEXT command which allows you to pick the specific job you want printed.

   The NEXT command is also handy in situations where you have forms frozen, you must print a job that requires forms different from those currently mounted, and you do not want to process any other jobs requiring the different forms. In this case, once the job you picked (with the NEXT command) is scheduled, LPTSPL requests a forms change and the new form is now frozen. So, to print only the special job and return to using the old forms, you could use the procedure given below. Type to the PTYCON subjob running LPTSPL,

NEXT n↵	(selects the job with the queue sequence number n for processing next)

LPTSPL types a message for you to change forms

```
MOUNT xxx FORMS THEN TYPE GO
```

You should change the forms and type

GO↵	(resumes processing)
PAUSE↵	(causes LPTSPL to stop when job n is finished)

Finally, when LPTSPL pauses, mount the old forms again and type

FORMS “oldforms”↲	(tells LPTSPL that “oldforms” are now mounted and still frozen)
GO↲	(resumes processing)

Commands
The following commands may be entered to LPTSPL to control its operation.

Command	Explanation
BACKSPACE *n*	Backspaces the file *n* pages and begins printing there.
CHKPNT	Takes a checkpoint immediately.
CURRENT	Prints out the current value of MLIMIT.
EXIT	Causes LPTSPL to return to the monitor. This command takes effect at the end of the current job.
FORMS	Causes LPTSPL to type out the name of the forms currently mounted.
FORMS *formsname*	Informs LPTSPL that form *formsname* is now mounted.
FORWARD *n*	Forward spaces the file *n* pages and begins printing there.
FREEZE	Causes the currently mounted forms to be frozen (i.e., will not schedule a different type of form for the printer).
UNFREEZE	Removes the effect of the FREEZE command.
GO	Continues processing after a LOCK, PAUSE, or STOP has been issued.
HELP	Types out a list of all the LPTSPL commands.
KILL	Aborts the current job and goes to the next job.
LIMIT *n*	Changes the limit for the current job to *n* pages.
LOCK	PAUSEs automatically after each job.
UNLOCK	Removes the effect of the LOCK command so that LPTSPL will continue automatically after each job.
MLIMIT *n*	Does not run any job which outputs more than *n* pages. This command has no effect on the job currently being processed. (See Helpful Hint 1.)
NEXT *n*	Selects the job with the queue sequence number *n* as the next job to be processed. (See Helpful Hint 3.)
PAUSE	Finishes processing the current job and stops.

Command	Explanation
REPRINT	Stops processing the current copy of the current file immediately and starts it over.
REQUEUE/switches	Stops writing immediately and places the job back into the queue for later processing. The switches are optional and if omitted, the /C switch is assumed.

Switch	Meaning
/A:n	Starts reprocessing after n minutes.
/B:n	Starts reprocessing n pages back in the file.
/C	Starts reprocessing at the current position in the file. This is the default.
/F:n	Starts reprocessing n pages forward in the file.
/H	"Holds" the request, i.e., starts reprocessing after 12 hours.
/T	Starts reprocessing at the "top", i.e., the beginning of the job.

RESET	Reinitializes LPTSPL and resets all parameters to their initial values. The effect of this command is delayed until the end of the current job.
SKPCOPY	Stops processing the current copy of the current file immediately and begins the next copy. If the last or only copy is being processed at the time you issue it, this command starts the next file in the request.
SKPFILE	Stops processing the current copy of the current file immediately and begins processing the first copy of the next file in the request.
START	Immediately starts processing files queued to the line printer. If you issue this command while LPTSPL is already started and there are no pending commands (i.e., you have not issued a RESET, EXIT, or PAUSE) you will receive the message, %LPTLAS LPTSPL ALREADY STARTED. If there are pending commands when you type the START command, LPTSPL issues the message [LPTCPC CLEARING PENDING COMMANDS] and then continues.
STOP	Stops the line printer immediately. **NOTE** ST is an acceptable abbreviation for the START command. Therefore, if STOP is desired, at least STO must be typed.
SUPPRESS /switch	Causes LPTSPL to ignore all vertical paper motion characters except for line feed. The switch specification is optional and if omitted, /F is assumed.

Switch	Meaning
/F	The SUPPRESS command applies only to the current file. This is the default.
/J	The SUPPRESS command applies to the entire print job.

NOSUPPRESS	Removes the effect of the SUPPRESS command. NOSUPPRESS is in effect at the time the spooler is initialized.
WHAT	Types out on your terminal the present status of LPTSPL. This information consists of the job name, the user name, the job's sequence number, and information about the file being printed.

CHAPTER 5
SPRINT

Function
SPRINT is a system program that reads a sequential input stream from the card reader, prepares the input for later processing by the Batch Controller, and enters the jobs into the Batch input queue. In addition to the above processing, SPRINT creates a LOG file for each job and enters a report of its processing into the file. This LOG is appended to BATCON when the job is actually run.

Operator's Responsibilities
Your main responsibility for SPRINT is to place the user's card decks in the card reader, ensure that they are read in properly, report any difficulties in the user's LOG file, remove the decks from the output hopper, and set and adjust various parameters for the system. The SPRINT commands available to you are described at the end of this chapter.

Startup
SPRINT is normally started automatically (via the auto file) under PTYCON when the system is brought up. The following procedures can be used if it becomes necessary to start up SPRINT manually.

```
@PTYCON
PTYCON> DEFINE (SUBJOB #) 0 (AS) SP
PTYCON> SP-LOG OPERATOR X 10300
PTYCON>
**** SP(0) 16:26:48 ****

 V 1.02.36, 1031 Development Sys., 11-FEB-76. TOPS-20 1(100)
@LOG OPERATOR X 10300
 JOB 11 ON TTY107 11-FEB-76 16:26
 END OF LOGIN.CMD.1
@
PTYCON> SP-SPRINT
PTYCON>
**** SP(0) 16:27:05 ****
SPRINT
PTYCON>
**** SP(0) 16:27:08 ****
/
PTYCON> SP-START
PTYCON>
**** SP(0) 16:27:15 ****
START
**** SP(0) 16:27:21 ****
!
PTYCON>
```

Error Procedures

1. Hopper Empty – SPRINT informs you that the hopper is empty by typing the message %SPTHPE HOPPER EMPTY OR STACKER FULL on your terminal (see Chapter 8 for more detailed descriptions of the SPRINT messages). The "hopper empty" condition means that SPRINT ran out of cards to read but was given no indication that the end of file was reached (i.e., SPRINT thinks that it is still in the middle of a job and is awaiting more input). There are two common causes of this condition; they are:
   a. The individual job contains more cards than fit into the hopper and you have not fed in the remainder of the deck as the cards are being read through the reader. To correct this, first remove the cards that have already been read (i.e., those in the output hopper), and simply put the unread portion of the deck in the input hopper and press the RESET button.
   b. The user forgot a $EOJ card and and there is no new job behind the deck to terminate this one. To continue processing you can
      1) place a $EOJ (or an end-of-file) card in the reader, then press the RESET button,
      2) press the EOF button if you card reader has one, or
      3) put more jobs in the input hopper (since a new job is one signal to SPRINT that the previous deck is terminated) and press the RESET button.
2. Pick Failure – A pick failure can occur when the edge of a card is damaged in such a way that the card reader cannot "pick"; i.e., draw the card through the reading mechanism. To correct this condition, remove the problem card from the hopper and smooth the damaged edge on a hard, even surface with the side of your thumbnail. Then put the card in front of the unread portion of the deck, joggle (or riffle) the deck, put it back into the input hopper, and press the RESET button. If you still get a pick failure message, try duplicating or repunching the card and reading it in again.
3. Read Errors – Sometimes a card goes through the card reading mechanism but does not get read correctly because of some (hopefully intermittent) difficulty either with the card reader or the card itself. When such an error occurs, SPRINT stops reading, the stop light on the card reader comes on, and SPRINT issues one of the following messages: %SPTCME CARD-MOTION ERROR, %SPTDTM DATA MISSED ERROR, %SPTDVE INPUT DEVICE ERROR, or %SPTRCK READ CHECK. Each of these messages is followed by the message, [SPTRLC RESET THE LAST CARD AND PRESS THE RESET BUTTON]. SPRINT expects you to "reset" the last card. To reset, take the last card that was read through the card reader (i.e., the last card in the output hopper), place it back in the input hopper as the first card, and press the RESET button. If you still get read errors, examine the card with a card gauge to see if it is off-punched. If it is, duplicate or repunch it and try reading it through again. If the problem is not off-punching, check to see if there is punching to the left of column 1 and to the right of column 80 (i.e., the card is offset and there is an extra column). If this is the case, return the deck to the user, indicate the bad card, and continue with the next job.
4. Returns to System Command Level – If SPRINT returns to system command level it is generally an indication of some internal problem in SPRINT and is not an error which you could have avoided. If SPRINT returns to system command level, simply restart SPRINT. Type SPRINT to the PTYCON subjob under which SPRINT was running and then give commands to start processing as described in the Startup procedures.

   If the same type of error recurs, it is more than likely a SPRINT problem (e.g., version skew) or hardware problem and you should consult your system administrator before trying to run SPRINT again.

Command Format

The format of a command to SPRINT is simply the command name followed by any argument as specified in the description of the individual commands. You may continue a command on a subsequent line (or lines) by typing the standard continuation character, the hyphen (-), as the last character that is not a TAB, space, or comment character on each line you want to continue. SPRINT responds to a continuation character by typing a number sign (#) on the next line. The following sequence of commands illustrates the use of the continuation character:

```
/TELL THIS IS THE BEGINNING - ↲
#  OF THE MESSAGE ↲
```

SPRINT prompts you with two different characters. It types a slash when it is awaiting input from you and it types an exclamation point when it is processing a job but still able to accept commands from you.

SPRINT checks for commands from you before each card is read, and if you have typed a command, SPRINT processes it immediately (except as noted in the descriptions of the command).

Commands
The commands available to you for controlling SPRINT are described below.

Command	Explanation
EXIT[1]	Exits to the system. This command takes effect at the end of the current job.
GO	Continues after a PAUSE or a STOP command.
HELP	Types a list of SPRINT operator commands and their actions.
KILL[1]	Kills the current job and flushes the input stream until the next job is encountered.
MSGLVL abc	Controls the size of the messages that SPRINT types on your terminal.

a=0	Types short messages
1	Types long messages
b=0	Suppresses user error cards
1	Types out the first fatal error card of the job
2	Types each error card
c=0	Does not type any cards
1	Types each $JOB card
2	Types each $ card
3	Types all cards

The default MSGLVL is 100 which means SPRINT uses long messages but does not type any user errors or cards.

Command	Explanation
PAUSE	Pauses at the end of the current job.
RESET[1]	Reinitializes SPRINT. This has the same effect as CTRL/C CTRL/C START (^C^C START). The effect of this command is delayed to the end of the current job.
START	Starts SPRINT processing. If you type this command once SPRINT has begun processing, you will get a warning message unless you type RESET first and no jobs are active.
STOP	Stops reading immediately and waits for further commands. The GO command will cause processing to continue with no ill effects.
TELL msg	Places the message (msg) from you into the current job's LOG file.
WHAT	Types out the current status of the job, including the job number, job name, user name, the job's sequence number, and information about the current file.

[1] If you issue the command while SPRINT is stopped in the middle of a job, it will cause an automatic GO.

CHAPTER 6
QUASAR

Function
QUASAR is the "heart" of the Batch system. All user queued and spooled requests are sent directly to QUASAR for processing. In addition, QUASAR directs BATCON and LPTSPL (i.e., schedules jobs for them and tells them when to run the selected jobs).

There is really no direct interaction between you and QUASAR; operator-QUASAR communication is effected through either BATCON or LPTSPL. When you set or change scheduling parameters for these programs (e.g., NEXT or MLIMIT for LPTSPL, or MJOB for BATCON), what actually happens is that the particular program sends the information to QUASAR and it is QUASAR that keeps track of the values of the various parameters and makes use of them during scheduling. On the very rare occasions when QUASAR must communicate with you, it passes the information through BATCON or LPTSPL and gets them to issue the messages. These messages can be identified by the prefix CTQ which stands for component-to-QUASAR. (See Chapter 8 for more information about these messages.)

Startup
QUASAR is normally started up automatically under SYSJOB when the system is brought up. However, the procedure to start up QUASAR manually is given below for your information.

```
^ESPEAK↵                    (Refer to the DECsystem-20 Operator's Guide for an ex-
RUN SYS:QUASAR↵             planation of the ^ESPEAK command)
^Z
```

Error Procedures

1. QUASAR Stops Running – When QUASAR stops running it is due to one of two conditions: a QUASAR error which results in QUASAR issuing a stop code and halting, or a system-detected error which causes QUASAR to exit to the system. In either case, you might want to have the situation looked at by the systems programmers. However, you can restart QUASAR by following the startup procedures described above.

   This "new" QUASAR that you just started up does not know about the existence of BATCON and any LPTSPLs you may have running so it is necessary to put them into contact with the "new" QUASAR. The first thing to do is to kill the current ones so that they can be restarted. For example, you would type,

   ```
   B-KILL↵           (kills BATCON)
   L0-KILL↵          (kills LPTSPL 0)
   L1-KILL↵          (kills LPTSPL 1)
   ```

   Each of the KILL commands causes any processing to stop immediately and sends a message to QUASAR. But since these jobs are unknown to the "new" QUASAR, it sends a message to you through them. The message is

   ```
   ?CTQMFQ MESSAGE FROM QUASAR – NOT A KNOWN COMPONENT
   ```

   This means that although you have stopped the actual processing of the jobs, the original requests for them still exist in the queues. They will be processed in the normal way when the "new" QUASAR is put in contact with BATCON and LPTSPL which occurs as soon as the programs have been "STARTed".

CHAPTER 7
QUENCH

Function
The QUENCH program is the normal mechanism by which the timesharing user communicates with the Batch and spooling system.

The QUENCH program enables the user to add, list, modify, and remove the queue entries in the various system queues. QUEUE (the command that runs the QUENCH program) is primarily a user command rather than an operator command, but you should be aware of the functions of the program because you are likely to want to examine and modify the queues. For a complete discussion of the QUEUE command see the *User's Guide*.

The queue requests submitted by both timesharing and Batch users are acted upon by the programs appropriate to each request. The following is a list of the system queues.

Queue Mnemonic	**Contents of the Queue**
LPT:	Line printer requests
INP:	Batch job requests

QUASAR is responsible for scheduling (i.e., selecting) jobs to be run and for giving them to the appropriate program. QUASAR bases its choices on various parameters and constraints that are set up by you, by the system administrator, and by the user who submitted the job. The types of limits you and the system administrator can place on the jobs are explained in the descriptions of the individual programs (LPTSPL and BATCON). There are a variety of constraints that a user can place on his own job. For example, he can specify that his job be run at a particular time, use a special form, or require a specific line printer.

Operator's Responsibilities
It is good practice to examine the queues periodically to determine how much of a backlog of jobs is built up and to see if any jobs are stuck in a queue because of some problem. One such problem may be that the appropriate program is not running. You may also have occasion to remove a request from a queue or alter an AFTER or DEADLINE parameter because of your operations schedule.

Commands
The following are some of the commands which you can use to examine the various queues (in order to keep track of what is happening in the system) and to modify or remove requests. For more detailed specifications about these commands, see the explanation of the individual commands in the *User's Guide*.

The most commonly used function is the facility to list the contents of all the queues. In addition, you may obtain listings of the contents of specific queues. The commands to accomplish this are listed below.

Command	**Explanation**
QUEUE ↵	Lists the contents of all the queues.
PRINT ↵	Lists the contents of the line-printer queue.
SUBMIT ↵	Lists the contents of the Batch input queue.

Note that you may also obtain listings of the individual queues with the appropriate QUEUE command. For example, PRINT is equivalent to QUEUE LPT:/L, SUBMIT is equivalent to QUEUE INP:/L.

Examples
Examples of examining, modifying, and removing QUEUE requests are given below.

1. Examining the Queues.

```
@Q↵
INPUT QUEUE:
STS        USER                                   JOB      SEQ  PRIO   TIME      AFTER
RUN    EIBEN                                     STATUS  2958   10   00:20:00
AFT    BATCH-USER                               BATJOB  4730   10   00:05:00   +19:5
8
DEP    EXERCISER                                KUL07   4992   10   00:05:00
          Dep=1
DEP    EXERCISER                                KUL07   5000   10   00:05:00
          Dep=1
DEP    EXERCISER                                KUL07   5007   10   00:05:00
          Dep=1

OUTPUT QUEUE:
DEV              USER                              JOB      SEQ  PRIO  LIMIT    AFTER
LPT       BATCH-USER                             EXAMPL  4731   10      21   +19:58

TOTALS: INP:    5 Jobs;   00:40:00 Sec. Runtime
          LPT:    1 Job;   21 Pases
```

The above is an example of a listing of the contents of all the queues. This was accomplished by typing Q↵. The circles around job sequence numbers 4730 and 4731 are to indicate which queue requests will be discussed in the following examples.

2. Modifying Requests.

```
@PRI EXAMPL[*,*]=/MOD/PRIO:50↵
[1 Job Modified]
```

The second example shows how you would modify the priority of a queue request (in this case a job in the line printer queue). The contents of the queues listed below show the modified request. A circle indicates the request with the new priority number.

```
@PRI↵

OUTPUT QUEUE:
DEV              USER                              JOB      SEQ  PRIO  LIMIT    AFTER
PLPTO * KOHN                                   DUMPER  4732   10      23
LPT       BATCH-USER                             EXAMPL  4731   50      21   +19:56
* Job being output now

TOTAL:  LPT:    2 Jobs;   44 Pases

@SU BATJOB[*,*]=/MOD/AFT:20↵
[1 Job Modified]
```

The above SUBMIT command shows how you would modify the value of the AFTER parameter. Below is a listing of the input queue with the AFTER parameter successfully changed.

```
@SU
INPUT QUEUE:
STS         USER                                       JOB     SEQ  PRIO   TIME     AFTER
RUN     EIBEN                                          STATUS  2958  10  00:20:00
AFT     BATCH-USER                                     BATJOB  4730  10  00:05:00  +03:4
7
DEP     EXERCISER                                      KUL07   4992  10  00:05:00
          Dep=1
DEP     EXERCISER                                      KUL07   5000  10  00:05:00
          Dep=1
DEP     EXERCISER                                      KUL07   5007  10  00:05:00
          Dep=1

TOTAL:  INP:    5 Jobs;   00:40:00 Sec. Runtime
```

3. Removing Requests.

```
@PRI EXAMPL[*,*]=/KILL
[1 Job Killed]
```

The above command deletes a request from the print queue. An examination of the print queue shows that the request no longer exists.

```
@PRI

OUTPUT QUEUE:
DEV           USER                                    JOB     SEQ  PRIO LIMIT    AFTER
PLPTO * FRANCIS                                       ANALYS 4733   10      35
* Job being output now

TOTAL:  LPT:    1 Job;   35 Pages
```

4. Removing Specific Requests
It is important to specify the job sequence number when you want to modify or kill a single request and multiple requests of the same name exist. In the example below, there are two requests with the name EXAMPL.

```
@PRI

OUTPUT QUEUE:
DEV           USER                                    JOB     SEQ  PRIO LIMIT    AFTER
LPT      BATCH-USER                                   EXAMPL 4734   10      21  +03:44
LPT      BATCH-USER                                   EXAMPL 4735   10      21  +03:44

TOTAL:  LPT:    2 Jobs;   42 Pages
```

The PRINT command below causes both of the requests to print EXAMPL to be killed.

```
@PRI EXAMPL[*,*]=/KILL
[2 Jobs Killed]
@PRI
The queue is empty
@
```

Again, there are two requests for EXAMPL in the queue.

```
@PRI

OUTPUT QUEUE:
DEV           USER                                         JOB      SEQ  PRIO LIMIT   AFTER
LPT        BATCH-USER                                    EXAMPL  4736   10       21   +03:41
LPT        BATCH-USER                                    EXAMPL  4737   10       21   +03:41

TOTAL:     LPT:     2 Jobs;    42 Pages
```

The print command below includes the job sequence number (4737) which causes only that one request to be killed.

```
@PRI EXAMPL[*,*]=/SEQ:4737/KILL
[1 Job Killed]
@PRI

OUTPUT QUEUE:
DEV           USER                                         JOB      SEQ  PRIO LIMIT   AFTER
PLPTO  *  KOHN                                           DUMPER  4738   10       23
LPT        BATCH-USER                                    EXAMPL  4736   10       21   +03:40
* Job being output now

TOTAL:     LPT:     2 Jobs;    44 Pages
```

CHAPTER 8

ERROR MESSAGES

Most of the messages returned to the operator fall in one of three categories. These categories are determined by the first character of the message.

? at the start of the message generally indicates a fatal error message
% at the start of the message represents an advisory or warning message
[at the beginning of a message indicates a comment line.

In addition, some of the messages have a 6-letter prefix. The first three letters indicate the program that issued the message (e.g., LPT for LPTSPL). The last three letters represent an abbreviation of the text of the message. For example:

```
?LPTPLE   PAGE LIMIT EXCEEDED
```

BATCON Operator Messages
The following messages from BATCON can appear on the operator's console.

ARGUMENT IS OUT OF RANGE
The numeric argument specified is greater than or lesser than the limits permitted for that command.

BATCON UNABLE TO KJOB
The job could not be KJOBed. This message is issued after BATCON has made two attempts to send a KJOB command. Further messages give alternative actions.

COMMAND IS AMBIGUOUS
The operator has entered an abbreviation that does not specify a unique command.

ILLEGAL NUMBER FORMAT
Illegal characters appear in a field that must contain only numerics.

ILLEGAL SUBJOB SPECIFIED
The operator specified a subjob number that was less than 1 or beyond the assembly parameter JOBMAX.

JOB CANNOT BE PUT IN MONITOR MODE
A subjob did not return to monitor mode after it sent a ↑C. Further messages give alternative actions.

MISSING ARGUMENT OR ILLEGAL DELIMITER
A command that requires an argument was entered without one or with a punctuation error.

NO SUBJOBS ARE ACTIVE
A subjob command has been issued, but no subjobs are currently running.

RESPOND BL-OPR line
A subjob is requesting a response to dialogue mode.

RESPOND BL-$
A subjob is waiting for a response. This message usually appears after a PLEASE command. Refer to the "Helpful Hints" section in Chapter 3 for an explanation of subjob response.

SUBJOB IS NOT ACTIVE
The subjob specified is not currently running.

SUBJOB #n JOB #n
This is a general heading to clarify the source of more messages.

UNKNOWN BATCON COMMAND
The command entered cannot be recognized by BATCON.

Component-to-QUASAR Messages
The following messages are system error messages issued by the interface module which BATCON and the spoolers use to communicate with QUASAR. Most of these messages indicate serious system problems and they should be brought to the attention of the system programmer or system administrator.

CTQASE	ADDRESSING SPACE EXHAUSTED There is no room left in the addressing space to receive an IPCF message from QUASAR. Consult the system administrator.
CTQCAP	CANNOT ACQUIRE A PID The program (i.e., BATCON or LPTSPL) does not have privileges to acquire a system name from IPCC. You must have WHEEL or OPERATOR privileges enabled to run BATCON or LPTSPL.
CTQCDQ	CANNOT DETERMINE PID OF SYSTEM QUASAR BATCON or LPTSPL could not find the PID of system QUASAR. Consult the system administrator.
CTQCGI	CANNOT GET IPCF DATA A request to the system for IPCF information (e.g., the maximum packet size) failed.
CTQFCI	FAILURE TO CONNECT TO THE INTERRUPT SYSTEM A failure occurred when BATCON or LPTSPL attempted to enable a PSI system monitor call for IPCF interrupts. Consult the system programmer.
CTQIRF	IPCF RECEIVE FAILURE The MRECV monitor call failed when the program (i.e., BATCON or LPTSPL) tried to receive an IPCF message.
CTQMFQ	MESSAGE FROM QUASAR – xxx The xxx stands for one of the following messages:
	ILLEGALLY FORMATTED DEVICE NAME The operator specified a device name in a format that is unacceptable to the program (i.e., BATCON or LPTSPL).
	ILLEGAL MESSAGE TYPE The component (i.e., BATCON or LPTSPL) sent a message to QUASAR in an invalid format. This indicates a problem with the program that sent the message. Consult the system programmer.
	INSUFFICIENT PRIVILEGES ENABLED An attempt was made to run BATCON or LPTSPL by a user who does not have sufficient privileges to run these programs. They can be run only with WHEEL or OPERATOR privileges enabled.

NOT A KNOWN COMPONENT
QUASAR is not in contact with the component (i.e., BATCON or LPTSPL) that tried to send a message. This condition generally indicates one of two problems. Either the component itself is defective or QUASAR crashed and when it was run again, the operator did not put the components back into contact with it. See Chapter 6 for more information and/or consult the system programmer.

SPECIFIED REQUEST IS NOT YOURS
QUASAR received a message that does not apply to the request it is currently processing. This condition indicates a difficulty with the program that sent the message. Consult the system programmer.

UNKNOWN QUEUE SPECIFIED
QUASAR received a message to process a queue request but the message specified a queue that is not known to QUASAR. This condition indicates a difficulty with the program that sent the message. Consult the system programmer.

WRONG VERSION NUMBER
The version number of the program (i.e., BATCON or LPTSPL) is incorrect for the version of QUASAR that you are running.

CTQNER NOT ENOUGH ROOM FOR POSSIBLE MESSAGES
The maximum size for a non-page mode IPCF message is greater than the preallocated space in the CSPQSR module. Consult the system administrator about reassembling the module or about reducing the maximum packet size.

CTQSQF SEND TO QUASAR FAILED
The program (i.e., BATCON or LPTSPL) tried to send an IPCF message to QUASAR and the MSEND JSYS failed. This is indicative of an IPCF problem. Consult the system programmer.

LPTSPL Operator Messages
The following messages from LPTSPL can appear on the operator's console.

?LPTCAF CAN'T ACCESS FILE xxx, CODE nnn – xxx
LPTSPL encountered an error while looking up the file it was trying to print. The n stands for the number and xxx for the text of one of the error codes listed in Appendix B of the *DECsystem-20 Batch Reference Manual.*

%LPTCCF CAN'T CHANGE FORMS IN THE MIDDLE OF A JOB
The operator issued a FORMS command while a job was being processed. Type PAUSE and wait for the current job to be completed.

%LPTCCL CCL ENTRY IS NOT SUPPORTED
LPTSPL does not support CCL entry (i.e., RUN with an increment of 1). It is treated as a standard RUN.

[LPTCPC CLEARING PENDING EXIT, RESET, AND PAUSE COMMANDS]
The operator issued a "START" command while LPTSPL was already started and there were other commands pending. LPTSPL clears the command(s) and continues processing.

?LPTDDE DEVICE xxx DOES NOT EXIST
The device (xxx) specified in the START command does not exist; either because it was typed incorrectly or because the device was not assigned to this job.

?LPTDIS	DEVICE xxx IS SPOOLED The device xxx specified in the START command is spooled. LPTSPL can write only on physical devices.
[LPTFAF	xxx FORMS ARE FROZEN] Informs the operator that xxx forms are now frozen.
?LPTICA	ILLEGAL COMMAND ARGUMENT nnn The argument nnn is either misspelled or is not a valid argument for this command.
%LPTLAS	LPTSPL IS ALREADY START'ED ON dev The operator issued a START command while LPTSPL was already processing.
[LPTLIR	LPTSPL IS RESET ON xxx] The copy of LPTSPL that was processing on another device has been RESET on device xxx by the operator and is awaiting a new START command.
[LPTLWP	LPTSPL WILL PAUSE AT END OF JOB] The operator issued a PAUSE command to LPTSPL while it was in the middle of a job. This message occurs in response to a WHAT command.
[LPTSIS	SPOOLER IS STOP'ED OR PAUSE'ING] This message is output in response to the WHAT operator command.
%LPTURC	xxx IS AN UNRECOGNIZED COMMAND The operator issued a command that is unknown to LPTSPL.
[LPTWFF	WAITING FOR xxx FORMS TO BE MOUNTED] The spooler is waiting for the operator to mount forms xxx. The operator must change the forms and type GO. This message is output in response to the WHAT operator command.
[LPTWFS	WAITING FOR A START COMMAND] The operator must issue a START command to cause LPTSPL to begin processing. This message is output in response to the WHAT operator command.

PTYCON Operator Messages
The following messages from PTYCON can appear on the operator's terminal.

? DOING A "GET" WITHIN A "GET" IS ILLEGAL
You are not allowed to process a GET from a file upon which you have done a GET.

? ILLEGAL SUBJOB DESIGNATOR
You referenced a subjob name or number that did not exist, or you tried to assign to a subjob a number that is larger than the number of PTY's on the system.

% NAME ALREADY IN USE, REASSIGNED TO THIS SUBJOB
You defined a subjob with a name that was already assigned to another subjob. PTYCON has terminated the association of this name with that other subjob and has assigned it to the one you just defined. However, you can still reference the other subjob by its subjob number.

? NO EXEC
You issued a PUSH command but there was no EXEC to run. Notify the system administrator.

? NO LOWER FORKS AVAILABLE

You issued a PUSH command but there were no free processes available.

? NO MORE PTY'S AVAILABLE!

You tried to create another subjob but there were no free PTY's to run the job.

? SUBJOBS ACTIVE, USE "PUSH" COMMAND!

You typed a CTRL/C to PTYCON and there were subjobs active. If you want to do a task at system level, use the PUSH command. If you really want to exit from PTYCON, use the EXIT command.

? TOO FEW ARGUMENTS!

You did not give the proper arguments for the command you typed. Because of the severity of the consequences of the DISCARD and KILL commands, you must supply either a subjob name or number or you must type the word ALL in its entirety. In addition, the DISCARD and KILL commands do not accept ESCape for the argument.

? TYPE "EXIT" TO EXIT FROM PTYCON!

You typed a CTRL/C to PTYCON and there were no subjobs active. However, you must still type EXIT to exit from PTYCON.

? UNEXPECTED PTYCON ERROR: CANNOT ENABLE FOR CONTROL-C INTERCEPT

You cannot run PTYCON if you have done a SET NO CONTROL-C-CAPABILITY. Do a SET CONTROL-C-CAPABILITY and then run PTYCON.

? UNEXPECTED PTYCON ERROR: COULDN'T GET HANDLE ON TTY FOR BINARY CHANNEL

This is an unexpected error and is not a result of any error on your part. Notify your system software specialist or system manager. You can try START immediately after the message and check the status of the subjobs. If that does not work, run PTYCON again, and within five minutes attach to the subjobs that became detached jobs.

? UNEXPECTED PTYCON ERROR: COULDN'T OPEN THE TTY IN BINARY FOR PTY COMMUNICATION

This is an unexpected error and is not a result of any error on your part. Notify your system software specialist or system manager. You can try START immediately after the message and check the status of the subjobs. If that does not work, run PTYCON again, and within five minutes attach to the subjobs that became detached jobs.

? UNEXPECTED PTYCON ERROR: DIDN'T GET LINE FEED AFTER RETURN FROM TERMINAL INPUT

This is an unexpected error and is not the result of any error on your part. Notify your system software specialist or system manager. PTYCON will continue to run.

? UNEXPECTED PTYCON ERROR: PANIC LEVEL INTERRUPT OCCURRED!

This is an unexpected error and is not the result of any error on your part. Notify your system software specialist or system manager. PTYCON will continue to run.

? UNEXPECTED PTYCON ERROR: RDTXT JSYS FAILED

This is an unexpected error and is not a result of any error on your part. Notify your system software specialist or system manager. You can try START immediately after the message and check the status of the subjobs. If that does not work, run PTYCON again, and within five minutes attach to the subjobs that became detached jobs.

? UNEXPECTED PTYCON ERROR: TABLK2: TABLE NOT IN PROPER FORMAT

This is an unexpected error and is not the result of any error on your part. Notify your system software specialist or system manager. PTYCON will continue to run.

? UNRECOGNIZED PTYCON COMMAND

You typed something to PTYCON which was not a PTYCON command. Type HELP to PTYCON for a list of PTYCON commands and their guide words.

SPRINT Operator Messages

The following messages from SPRINT can appear on the operator's console.

?SPTCCC	CAN'T CREATE CTL OR LOG FILE – xxx SPRINT is unable to create either the control or LOG file. The xxx represents one of the messages explained in Appendix B of the *Batch Reference Manual.*
%SPTCCL	CCL ENTRY IS NOT SUPPORTED SPRINT does not support CCL entry (i.e., RUN with an increment of 1). It is treated as a standard RUN.
%SPTCDI	DEVICE *dev* CAN'T DO INPUT Input cannot be performed on the device *dev* specified in the START command. For example, input may have been requested for a device that can do only output, such as the line printer (e.g., START LPT:). SPRINT can read only from the card reader.
?STPCER	COMMAND ERROR – RETYPE LINE There is a syntax error in the command string.
%SPTCME	CARD-MOTION ERROR ON CDR*n* A card was not read correctly. Reset the last card and press the RESET button.
?SPTDDI	DOUBLE DIRECTORY ILLEGAL Two directory names cannot appear without an intervening filename.
?SPTDDV	DOUBLE DEVICE ILLEGAL Two names appeared in a row without an intervening filename (e.g., LPT:MTA:) or, two colons appeared in a row (e.g., DSK::MYFILE).
?SPTDEX	DOUBLE EXTENSION ILLEGAL Two extensions (i.e., file types) cannot appear in a row without an intervening filename or comma.
?SPTDFN	DOUBLE FILENAME ILLEGAL Two filenames appeared in a row (e.g., $COBOL TEST1 TEST2) or, two periods appeared in a row (e.g., TEST..CBL).
[SPTDNR	INPUT DEVICE *dev* NOT READY] The specified input device requires operator action to prepare it for reading.
%SPTDTM	DATA MISSED ERROR ON CDR*n* A card was not read correctly. Reset the last card and press the RESET button.
%SPTDVE	INPUT DEVICE ERROR ON CDR*n* A card was not read correctly. Reset the last card and press the RESET button.

%SPTHPE	HOPPER EMPTY OR STACKER FULL ON CDR*n* One of the following two conditions exist: 1. the input hopper is empty and SPRINT is expecting more cards (because the end of job has not been reached), or 2. the output stacker is full and SPRINT cannot read any more cards until room is made for them. In either case, correct the condition and press the RESET button.
?SPTIDS	ILLEGAL DIRECTORY SPECIFICATION The format of the directory information enclosed within the square ([ ]) or angle (<>) brackets is incorrect.
?SPTIOE	I/O ERROR WRITING LOG OR CTL FILE An I/O error occurred while SPRINT was writing the LOG or the CTL file. The job must be rerun.
?SPTLKE	LOOKUP/ENTER ERROR *n* – *xxx* SPRINT encountered an error while looking up a file specified by the operator in the START command. The *n* represents the number and *xxx* represents the text of one of the messages explained in the Appendix B of the *DECsystem-20 Batch Reference Manual.*
?SPTNDV	NULL DEVICE ILLEGAL A colon has been found without a preceding device name.
?SPTNJA	NO JOBS ACTIVE A command was given by the operator to perform an action (KILL, TELL, etc.) for the current job, but SPRINT was not processing any jobs at the time the command was issued.
?SPTNSD	NO SUCH DEVICE *dev* The device name does not exist, or it was not assigned to this job.
%SPTNST	SPRINT IS NOT STARTED The operator must type START (refer to Chapter 5) to cause SPRINT to begin processing.
[SPTOIR	OPERATOR INTERACTION REQUIRED] This is the response to the WHAT command if operator interaction is required to continue processing.
%SPTPKF	PICK FAILURE ON CDR*n* The card about to be read cannot be drawn through the card reader. Check the card (i.e., the first one in the input hopper) and take the appropriate action. (See the Helpful Hints in Chapter 5 for more information.) To continue reading, put the card back in the input hopper and press the RESET button.
%SPTRCK	READ CHECK ON CDR*n* A card was not read correctly. Reset the last card and press the RESET button.
[SPTRLC	RESET THE LAST CARD AND PRESS THE RESET BUTTON] A card went through the card reader but was not read correctly. This message is always preceded by one of the following four messages: 1. %SPTCME 2. %SPTDTM 3. %SPTDVE 4. %SPTRCK

%SPTSPL	DEVICE *dev* IS SPOOLED The input device *dev* specified in the START command is spooled. SPRINT should read only from physical devices.
%SPTSTD	SPRINT IS ALREADY START'ED ON *dev* A START command was issued when SPRINT was already processing.
%SPTUIE	UNRECOVERABLE INPUT DEVICE ERROR – STATUS *nnn* SPRINT received an input error indication from which it could not recover. The I/O status was *nnn*.

CHAPTER 9
CRASH PROCEDURES AND STOP CODES

Crash Procedures

When QUASAR encounters a problem (e.g., a JSYS failure or an inconsistent data base), it stops immediately. It does this rather than risking the possibility of compounding the error, rendering it impossible to recover, and/or obliterating clues as to what went wrong. When QUASAR stops, it types out a message of the form:

```
QUASAR STOP CODE – xxx
```

The xxx represents the 3-letter stop code.

In addition to typing out the stop code message, QUASAR stores information pertinent to the crash. This information is stored in a global block called G$CRAC.

Locations G$CRAC through G$CRAC+17_8 (inclusive) contain the contents of the accumulators at the time of the crash. G$CRAC+$20_8$ contains the address of QUASAR's internal page table. G$CRAC+$21_8$ left half has the number of queues and right half, the address of "TBLHDR", the list of queue headers. G$CRAC+$22_8$ contains the address of the bottom of the pushdown stack and G$CRAC+$23_8$, the stop code in left-justified SIXBIT.

In the explanations that follow, references to accumulators mean the saved ones in the G$CRAC block; they do not refer to the current contents.

Stop Codes

Code	Module	Explanation
ASE	QSRMEM	ADDRESSING SPACE EXHAUSTED QUASAR cannot get another page of address space in which to build its queues.
BDN	QSRQUE	BAD DEVICE NAME INTERNALLY GENERATED The device name that the monitor sent to QUASAR in a spool message was illegally formatted.
BPN	QSRMEM	BAD PAGE NUMBER An illegal page has been passed to a subroutine that does page manipulation. The number of the bad page is stored in AC13.
BRS	QSRFSS	BAD REQUEST SIZE A request must not exceed the length of a page. In this case, a call was made to failsoft a request whose length was less than or equal to zero or was greater than 512 (decimal). The actual request size is stored in AC1.
BSD	QSRT20	BAD SPOOL DATA The spool message sent to QUASAR by the system is in an illegal format (e.g., the filename could not be parsed). This is a system error. Consult the system administrator.

Code	Module	Explanation
CAP	QSRT20	CANNOT ACQUIRE A PID FOR QUASAR A failure (MUTIL JSYS) occurred when QUASAR requested that IPCC give it the name QUASAR. This condition could be caused by a program named QUASAR that is already running. The error code is stored in AC1.
CFN	QSRMEM	CANNOT FIND n CONTIGUOUS PAGES BATCON or LPTSPL requested n contiguous pages of memory but memory is so fragmented that n contiguous pages do not exist.
CGP	QSRT20	CAN'T GET PACKET SIZE QUASAR is unable to determine the maximum size of a non-page mode IPCF message. The error code is stored in AC1.
COP	QSRT20	CAN'T OPEN PRIME QUEUE QUASAR received an error while it was trying to open the master queue file. The error code returned by the system is stored in AC1. Note that one special case of this type of error is handled by the PQI stop code.
CRD	QSRQUE	CREATE REJECTED DEFER DATA A deferred spooled request was found to be invalid when QUASAR tried to process it.
CRF	QSRFSS	CREATE REJECTED FAILSOFT DATA After a crash, the internal queues which had been saved on disk were found to be bad.
CRL	QSRQUE	CREATE REJECTED LOGOUT DATA Deferred spooled requests were found to be invalid at LOGOUT time. In such cases, where QUASAR itself has created the requests, a severe problem is indicated.
CRM	QSRQUE	CREATE REJECTED MODIFIED REQUEST QUASAR's attempt to place a modified queue request back into the proper queue failed.
CRS	QSRQUE	CREATE REJECTED SPOOL DATA Spooled requests (deferred) were found to be invalid at LOGOUT time. In such cases, where QUASAR itself has created the requests, a severe problem is indicated.
CSQ	QSRT20	CANNOT SET IPCF QUOTAS QUASAR's attempt to set IPCF send and receive quotas failed.
CUD	QSRFSS	CLEARING UNUSED DPA QUASAR tried to release space in the failsoft file system. However, the space QUASAR was trying to release had never been allocated.
CUF	QSRT20	CANT UPDATE FILE A failure (UFPGS JSYS) occurred while QUASAR was attempting to update the failsoft system master file. The error code is in AC1.
CUI	QSRT20	CANT UPDATE INDEX A failure (UFPGS JSYS) occurred while QUASAR was attempting to update the failsoft system index. The error code is stored in AC1.

Code	Module	Explanation
DIF	QSRT20	DEBRK OF INTERRUPT FAILED An attempt by QUASAR to dismiss an interrupt with the DEBRK JSYS failed.
DTL	QSRFSS	DPA TOO LARGE QUASAR was trying to allocate space in the failsoft system. The space it was trying to allocate was beyond the limits of the file and the attempt failed.
DTS	QSRFSS	DPA TOO SMALL QUASAR was trying to allocate space in the failsoft file system. The attempt was made to do this in a reserved portion of the file and so it failed.
FCE	QSRMEM	FREE COUNT EXCEEDS INITIAL VALUE The internal count of the number of pages available has become larger than the initial value. This type of error is generally caused by calling a page manipulation subroutine twice with the same argument.
FCN	QSRMEM	FREE COUNT OF PAGES IS NEGATIVE The internal count of the number of pages has become negative. This type of error is generally caused by calling a page manipulation subroutine twice with the same argument.
FSP	QSRT20	FAILURE TO SET SYSTEM PID TABLE An attempt by QUASAR to fill in the system's special PID table failed. The error code is stored in AC2.
FUD	QSRFSS	FOUND UNUSED DPA QUASAR was trying to read an unused area of the failsoft file.
IAO	QSRSCH	ILLEGAL AFTER QUEUE OPERATION The only valid operation allowed to be performed on the after queue by the scheduling module is "ordering". An attempt was made to perform some other operation.
MRF	QSRT20	MESSAGE RECEIVE FAILURE An unrecoverable error occurred when QUASAR tried to receive an IPCF message. The error code is stored in AC1.
NBR	QSRQUE	NEXTJOB'ING BAD REQUEST QUASAR did a consistency check of the data in a request before sending it to a known component. The consistency check failed.
NMF	QSRFSS	NO MORE FILESPACE The master queue file is full.
NSD	QSRT20	NO SPOOLING DIRECTORY QUASAR is unable to determine the directory number for the name <SPOOL> (STDIR JSYS failure). The error code is in AC1.
PIC	QSRT20	PID TO INTERRUPT CONNECT FAILED The PID for system QUASAR could not be connected to the interrupt system (MUTIL JSYS failure). The error code is in AC1.

Code	**Module**	**Explanation**
PLL	QSRMEM	PAGE LINK LOST The chain of pointers linking pages is defective. This was detected by observing an incorrect calling sequence in the free space management routine.
PQI	QSRT20	PRIME QUEUE INTERLOCKED The prime queue could not be opened because it was interlocked by another job. This condition generally indicates that another copy of QUASAR is running.
QNR	QUASAR	QUASAR NOT RESTARTABLE QUASAR cannot be restarted. That is, you cannot just type START after a CTRL/C or stop code; QUASAR must be run again.
QSZ	QSRMEM	QUEUE SIZE ZERO QUASAR was attempting to build a queue but the size of the free space call for the queue was zero. This could be caused by an incorrect calling sequence; specifically, the M$GFRE routine. AC15 contains the address of the queue header whose size is zero.
RCN	QSRFSS	REQUEST COUNT NEGATIVE QUASAR's request count was decremented below zero.
RCW	QSRFSS	REBUILD COUNT WRONG At the time that QUASAR was rebuilding the master queues from disk it found that the number of requests which the file index indicated were in the file was different from the number actually in the file.
RFP	QSRMEM	RELEASING A FREE PAGE QUASAR tried to deallocate an already deallocated page of memory.
TMS	QSRFSS	TOO MANY SECTIONS The failsoft system file contains more sections than QUASAR is assembled to handle.
WQV	QSRFSS	WRONG QUEUE VERSION The master queue file contains a format version number that this version of QUASAR is not prepared to handle.

INDEX

INDEX (Cont.)

DECsystem-20 BATCH
Operator's Guide
DEC-20-OBOGA-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name ______ Date ______

Organization ______

Street ______

City ______ State ______ Zip Code ______
or
Country

If you require a written reply, please check here. ☐

Please cut along this line.

--- Fold Here ---

-- Do Not Tear - Fold Here and Staple --

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

digital