XVM/RSX-PART VIII
BATCH PROCESSING

CHAPTER 1

INTRODUCTION TO BATCH PROCESSING

1.1 INTRODUCTION

RSX supports an extensive set of batch-processing functions. The Batch Processor (BATCH) is a specialized program that has many of the characteristics of an I/O handler. RSX allows the use of BATCH as an extension of on-line task development (TDV) under MULTIACCESS or as a total system environment.

BATCH offers fast, efficient data-processing by enabling an operator to stack multiple jobs to be executed in sequence. BATCH is file-oriented, device-independent, and suitable for use in both card-oriented environments and terminal job-entry/remote-access operations. It has been designed to work well in small configurations without sacrificing the high throughput and short turnaround that large systems require.

Some BATCH commands allow each job to invoke all TDV functions. Other BATCH commands also provide various control and message logging functions.

Job accounting and account summaries in the Batch System are handled by the TDV functions ACI and ACD. Control lines are listed on the line printer and, for additional backup, an operator log is maintained. BATCH also provides:

- Job header and trailer pages
- Full FORTRAN, MACRO and loader facilities, including overlays
- System protection from undebugged jobs
- Convenient operator communication and control
- SLIP, a sophisticated file editing and updating program with search capabilities (described in Part VII of this manual)

1.2 SAMPLE BATCH SEQUENCE

Figure 1-1 illustrates a sample set of commands read during a batch-processing operation. The figure shows the commands in card format, but they could also have entered (into a disk file) from a terminal.

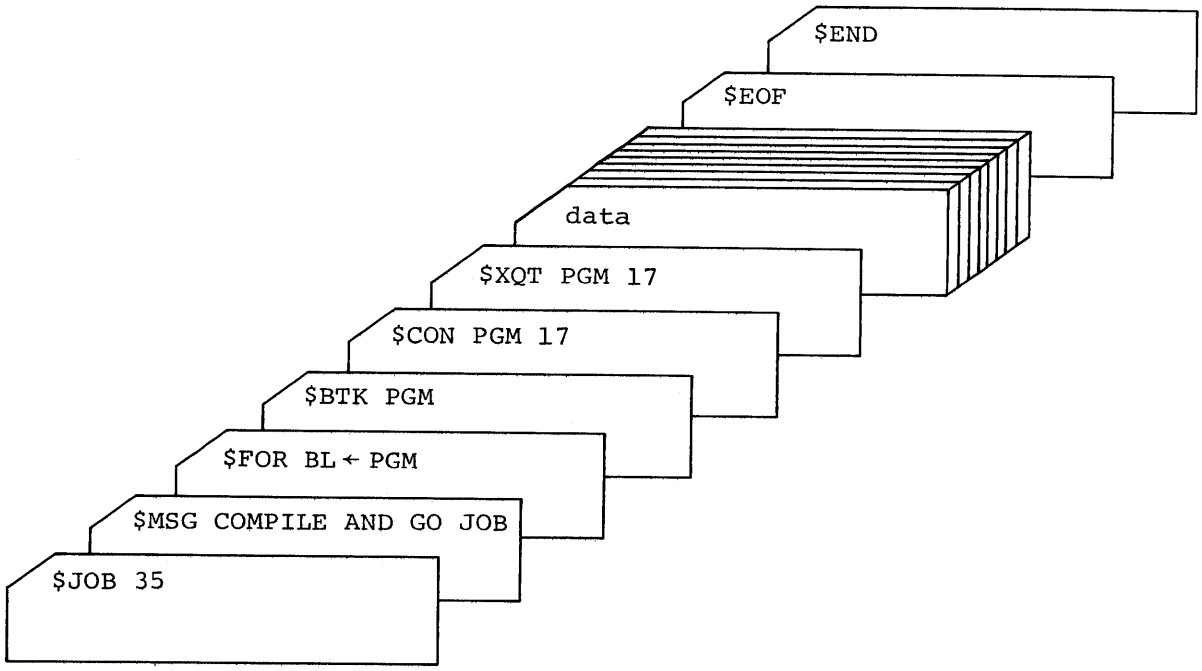


Figure 1-1 Batch-Processing Example

In this example, the $JOB line provides accounting information and initiates the job.

$MSG is a message line.

The program source file is already on disk. If it is not, the following sequence is also needed prior to compilation of the FORTRAN source file:

```
$DECK file name
  .
  .
  .
source program
  .
  .
  .
$EOF
```

$FOR requests FORTRAN compilation of PGM. The options specified in this example include a binary file (B) and a listing (L).

$BTK calls the Basic Task Builder to build a task with the same name as the binary file that serves as input to BTK. BTK builds the task to run in a partition named TDV, gives the task a default priority of 400, and builds the task to run in user and page modes. The task cannot contain overlays. (If nonstandard options are desired, the user must call TKB, a more flexible version of the Task Builder.)

$CON stores the task on a user disk. $XQT executes it from that disk.

When the task is running, it is capable of reading formatted data (IOPS ASCII) from the BATCH input device. The data section is terminated by $EOF (end-of-file). After the task completes execution, it exits. Should the task fail to exit, the operator can force it to do so by using the OPR MCR Function task (see Chapter 4).

$END indicates the end of the job file.

CHAPTER 2

INVOKING AND USING BATCH

2.1 JOB FILE SUBMISSION

Before BATCH is invoked, job files must be submitted to it by means of the QJOB directive, which can be issued by:

- TDV function QUEUE
- BATCH command $QUEUE, which calls TDV function QUEUE
- User-written programs in FORTRAN or MACRO

In each case, the QJOB directive supplies the following information:

- Name of the job file (if the device is file oriented)
- LUN from which the file is to come
- Maximum time that the job can run
- Job class
- Whether the operator is required
- Whether sequencing is required
- Whether the file is to be deleted after execution
- Time of job submission (supplied by the system)
- Memory use
- Use of hold mode
- Whether this is a priority job
- Login device, unit and UFD

2.2 INVOKING BATCH PROCESSING

BATCH is invoked by the operator using the MCR command:

```
MCR>OPR BATCH
```

BATCH initializes itself and waits. Following an OPR GO command, BATCH scans the job file request list, selects the job file that

passes certain logical tests and has the highest priority, and begins processing that job file. After that job file has been processed, another one is usually selected. The operator can use special OPR commands to supervise BATCH operations. Because BATCH loads and waits, the OPR GO command is needed before the first job file is executed.

2.3 CONTROL LINES

Only control lines (lines beginning with $) have meaning to BATCH. Other lines, which contain data, FORTRAN source text or other text, are passed on to the tasks called by control lines. Control lines fall into two groups:

1. Special BATCH commands.

2. Commands that call TDV Function tasks.

Table 2-1 lists commands of the first type. Chapter 3 describes them in more detail.

Table 2-1
Special BATCH Commands

Command	Effect
$JOB options	JOB identifier; initiates a job.
$MSG message	Prints a message on both the operator console (operator log) and listing device.
$PAUSE message	Temporarily suspends execution of the current job and requests that BATCH pause. To continue processing the job, the operator uses an OPR GO command. $PAUSE is printed on both the operator console and listing device.
$LOG message	Prints a message on the listing device only.
$EJECT	Prints a form feed on the listing device.
$	Prints a dollar sign (echoes the command line) on the listing device.
$END comment	End-of-job-file; prints a job trailer page on the listing device and notes end-of-job-file on the operator log.
$QUIT	Similar to $END. Equivalent to a physical end-of-file.
$ERROR comment	Requests that BATCH process subsequent records if the job is killed. $ERROR is ignored unless this condition exists. $ERROR is printed on the listing device.

All TDV Function tasks can be used with BATCH. Part VII of this manual describes these functions in detail. TDV commands must be preceded by $ when used with BATCH.

2.4 TERMINATING BATCH PROCESSING

BATCH continues to run jobs until the operator exits with an OPR EXIT command. This command causes BATCH to exit after completing the current job file. If BATCH finishes processing all waiting job files, it remains idle until more job files are submitted or until the operator exits.

CHAPTER 3

SPECIAL BATCH COMMANDS

3.1 $JOB: BEGIN A JOB

Each job must begin with a $JOB line. BATCH ignores all lines until it recognizes a $JOB line, but the TDV function QUEUE requires that the first line of a job file be a $JOB line.

$JOB lines have the format:

Form:	$JOB nn[T=time][DEL][HLD][C=class] [M=memory][SEQ][OPR][FRC][UFD=Rmn<ufd>]
Where:	nn is a user account number in the decimal range 1 to 99 time is an integer in the decimal range 1 to 1023, representing the maximum number of minutes that the job can run class is an integer in the decimal range 0 to 7 memory is an integer in the decimal range 1 to 128, representing memory use (in K) m is F, K or P, representing the type of disk:RF DECdisk, RK cartridge disk, or RP disk pack, respectively n is the unit number of the disk ufd is the name of the user file directory
Example:	$JOB 15 UFD=<EAG> SEQ

$JOB line options are the same as those for the QUEUE command.

A job file should contain only a single job; however, multiple jobs can be included in a single job file by beginning each job with a $JOB line. The second and subsequent $JOB lines terminate the previous job and begin a new job. All jobs in the same job file are executed using the same set of option flags. Only the first occurrence of a particular option is recognized. Subsequent occurrences of an option are ignored.

BATCH does not process $JOB line options. The $JOB line is passed to a Job Startup Processor (JOB...) that determines the user account number. All options in the $JOB line are processed by the TDV function QUEUE when it queues the job request. (Refer to Part VII of this manual for a description of QUEUE.) QUEUE scans the job file,

recognizes $JOB lines, processes any options found and encodes the option information into the QJOB directive.

On recognizing a $JOB line, BATCH first terminates the previous job, if any. BATCH then initiates the new job and invokes JOB.... This task determines the user account number from the $JOB line, prints a job header page and logs the start of a new job on the operator console. The user account number is saved in the BATCH accounting file for use by the BATCH Job Termination Processor (END...).

The job time limit ($JOB T=time option) restricts the length of time that a job file can run. When that amount of time has been used, the job is terminated, unless the system manager grants an extension. Job execution time is real time (clock time) less any time spent on the following functions:

1. Printing job header and trailer pages.

2. Waiting for the operation to proceed from a $PAUSE.

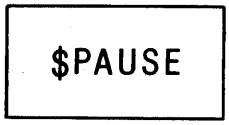
3. Waiting for a TDV partition to become available (TDV "PARTITION(S) BUSY" message).

3.2 $MSG: SEND A MESSAGE

The $MSG line is used to communicate with the operator. BATCH processes it internally. The entire record is printed on both the operator console and listing device. $MSG records can be interspersed with both data records and control or TDV command lines.

$MSG lines have the format:

Form:	$MSG message



3.3 $PAUSE: SUSPEND BATCH

The $PAUSE line delays processing of the job until the operator resumes it. BATCH processes the $PAUSE record internally. It is printed on both the operator console and listing device. $PAUSE records can be interspersed with both data records and control or TDV command lines. Time is not charged to the job until the operator resumes the job.

$PAUSE lines have the format.

Form:	$PAU[SE] [message]

The message normally tells the operator what to do before resuming processing. Including a message in the $PAUSE line makes it unnecessary to precede the line with a $MSG line.

To resume processing, the operator must issue the OPR GO command. Because $PAUSE requires operator intervention, BATCH jobs should avoid using it unless strictly necessary.

3.4 $LOG: LOG A COMMENT

$LOG sends a message to the listing device. BATCH processes $LOG lines internally. They can be interspersed with both data records and control or TDV command lines.

$LOG lines have the format:

Form:	$LOG [message]

3.5 $EJECT: EJECT A PAGE

$EJECT prints a form feed on the listing device. BATCH processes $EJECT lines internally. They can be interspersed with both data records and control or TDV command lines. $EJECT lines are not printed.

$EJECT lines have the format:

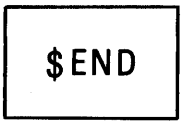
Form:	$EJE[CT]

3.6 $: PRINT A LINE

$ prints a line containing just a dollar sign (in column one) on the listing device. The rest of the printed line is blank. $ lines are processed by TDV. The $ can be interspersed with only control or TDV command lines. They cause task termination when interspersed with data lines.

$ lines have the format:

Form:	$



3.7 $END: END A JOB FILE

$END is usually the last line of a job file.

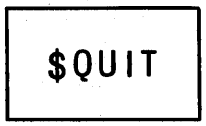
$END lines have the format:

Form:	$END

$END invokes the BATCH Job Termination Processor (END...). This task prints a job trailer page, logs the job termination on the operator console and updates the BATCH accounting file. The $END line is not printed.

END... is invoked whenever a job terminates. Besides a $END line, job termination is caused by:

- End-of-file on input of job file
- Appearance of a $JOB line when a job is still in progress
- Early termination of a job file because the job file has exceeded its time limit, because of an error condition or by action of the operator



3.8 $QUIT: END BATCH INPUT

$QUIT signifies the end of batch input. It is logically equivalent to a physical end-of-file. This line is not printed.

$QUIT lines have the format:

Form:	$QUI[T]

$QUIT is provided for compatibility with older versions of the Batch System. Whenever possible, $END should be used as the last line of a job file.

$ERROR

3.9 $ERROR: PROCESSING AFTER JOB TERMINATION

The $ERROR record ensures that certain processing occurs even if a job exceeds its time limit or if the operator stops it by means of the OPR KILL command. $ERROR has no effect unless one of these conditions exists. If one does exist, $ERROR:

- Restores job status
- Causes processing of records that follow $ERROR
- Prints the $ERROR line, including any comments, on the listing device

The location of $ERROR is important.

$ERROR lines have the format:

Form:	$ERR[OR] [comment]

The following example illustrates the use of $ERROR. The commands between $ERROR and $END are obeyed even if the job is terminated by an OPR KILL command or if the time limit is exceeded before $ERROR is encountered.

```
$JOB
.
.
.
$ERROR
.
.
.
$END
```

$ERROR is ignored if a job is terminated with the OPR STOP command.

OPR

CHAPTER 4

OPERATOR INTERACTION WITH BATCH

4.1 OPERATOR CONSOLE

The operator console (operator log), is usually assigned during BATCH operations to the same hard-copy terminal as the device for MCR command input. The operator invokes BATCH with the command:

```
MCR>OPR BATCH
```

When this is done, a line feed is printed on the operator console and BATCH enters the wait state (see Table 4-1).

As BATCH begins to process a job, it prints a message on the console. When BATCH finishes processing a job, it prints another message on the console. These messages are printed by the tasks JOB.... and END..., respectively. The operator also uses the console for the special operator (OPR) commands described in the following sections.

4.2 OPR COMMANDS AND BATCH STATES

Certain OPR commands give the operator control over batch processing that is unavailable to users.

BATCH can be in one of the states listed in Table 4-1. The OPR commands perform various functions, including changing the BATCH state. Table 4-2 lists the operator commands.

The operator can determine the current BATCH state by invoking OPR without specifying a command:

```
MCR>OPR
```

OPR responds by printing the current BATCH state, a slash (/) and the state that BATCH will enter following completion of the current job file. The number of job files queued is also printed.

Table 4-1
BATCH States

State	Meaning
Run	If this is printed as the current BATCH state (before the slash), BATCH is currently executing a job file. If printed as the next BATCH state, BATCH will continue to process job files when the current job file is complete.
Idle	BATCH is not currently executing a job file.
Pause	BATCH is waiting for an OPR GO command to continue from a $PAUSE. This state implies that BATCH is executing a job file.
Wait	BATCH will wait between job files. Following completion of the current job file, BATCH becomes idle until an OPR GO or EXIT command is issued.
Exit	BATCH will exit following completion of the current job file.

Table 4-2
OPR Commands

Command	Effect
BATCH	Load BATCH
SC[HEDULE]	Sets job file selection parameters
JO[B LIST]	Lists queued job files
ON	Indicates that the operator is available
OF[F]	Indicates that the operator is unavailable
WA[IT]	Causes BATCH to wait between job files
GO PR[OCEED]	Causes BATCH to go on to the next job file after WAIT or continues the job file after $PAUSE
HO[LD] n [day]	Indefinitely prevents processing of a job file
RE[LEASE] n [day]	Releases the job file after HOLD
EX[IT]	Causes BATCH to exit after the current job file
FO[RCE] n [day]	Forces a job file to run next
CA[NCEL] n [day]	Cancels a job file request
CA[NCEL] ALL	Cancels all job file requests
TL[ACT]	Specifies the action BATCH should take after a job file exceeds the time limit
ST[OP]	Terminates the current job file
KI[LL]	Terminates the current job file
MO[RE] [time]	Allows the current job file to exceed the time limit
AB[ORT]	Terminates the current job file

4.2.1 BATCH: Load BATCH

The operator loads BATCH by typing a command in the format:

Form:	OPR BATCH

BATCH loads into core, initializes itself and waits. The operator should set job file selection parameters, using the OPR SCHEDULE command. The operator should then use the OPR GO command to initiate job selection and execution.

4.2.2 SCHEDULE: Set Job File Selection Parameters

SCHEDULE sets the parameters that are used in determining job file priority. These parameters are read either from a file with extension SCH (already existing on disk) or directly from the OPR SCHEDULE command line.

After the operator types the command in the format:

Form:	OPR SC[HEDULE]

an asterisk appears on the next line. The operator can then set parameters directly or name a file containing parameter specifications. To set parameters directly, the operator supplies them in the format (in any order):

Form:	TF=n WF=n CF=n TM=n WM=n CM=n
Where:	TF represents the time factor WF represents the wait factor CF represents the class factor TM represents the time maximum WM represents the wait maximum CM represents the class minimum n is a decimal number
Example:	TF=5 WF=10 CF=0 TM=60 WM=120 CM=0

The numbers in the example are typical entries. Section 6.3 explains the parameters in greater detail.

Instead of setting the parameters in this way, the operator can respond to the asterisk by entering the name of an existing file. A scheduling file consists of one line of scheduling parameters in the format previously given. For example, if the operator types:

```
*NOON
```

BATCH looks up NOON SCH. If the file is not there, it prints:

```
FILE NOT FOUND
```

If the file is there, it is printed and the parameters that it specifies are set into the schedule table in BATCH.

In this way, the OPR SCHEDULE command can enter parameters from a scheduling file, but it cannot create such a file. This must be done by ordinary editing. The scheduling file can be named for the time of day that it will be used.

Instead of using the OPR SCHEDULE command, the operator can use a combination of OPR HOLD and FORCE commands to request that each task run in turn. This, however, is much less convenient.

4.2.3 JOB LIST: List Queued Job Files

The operator can request a list of the currently running job file and the queued job files by typing a command in the format:

Form:	OPR JO[B LIST]

If the queue is empty, the message is printed:

NONE WAITING

Otherwise, the BATCH system activity and backlog are listed, including:

- Job file sequence number (assigned by the QJOB directive)
- Day of the month when the job file was queued
- Any job characteristics that have been specified, such as run time, memory required, class, sequencing and operator

Cancelled job file requests are listed with a zero job file sequence number. These requests are removed from the job file requested queue when BATCH next selects a job file.

4.2.4 ON: Indicate That The Operator is Available

The operator can indicate availability by typing a command in the format:

Form:	OPR ON

Job files that need operator assistance are not run unless the operator is at the system. The operator indicates this with the OPR ON command. When BATCH is loaded, OPR ON is assumed.

4.2.5 OFF: Indicate That The Operator is Unavailable

The operator can indicate unavailability by typing a command in the format:

Form:	OPR OF[F]

This prevents job files that need assistance ($PAUSE lines, tape mounting, special forms, etc.) from being selected for execution.

4.2.6 WAIT: Cause BATCH to Wait Between Job Files

The operator can request that BATCH wait after finishing the current job file and before starting the next job file. This command gives the operator time needed to mount new tapes or to perform other system functions. It has no effect on the job file currently in progress.

The OPR WAIT command has the format:

Form:	OPR WA[IT]

4.2.7 GO and PROCEED: Cause BATCH to Continue

When BATCH is waiting between job files because of an OPR WAIT command or is pausing within a job file because of a $PAUSE record, operation is continued by typing a command in the format:

Forms:	OPR GO OPR PR[OCEED]

These two commands are equivalent. OPR treats them identically. Because BATCH loads in the wait state, it is necessary to use an OPR GO command before the first job file is executed.

4.2.8 HOLD: Prevent Processing of a Job File

The operator can indefinitely keep a job file from running by typing a command in the format:

Form:	OPR HO[LD] n [day]
Where:	n is the job file sequence number (printed by the OPR JOB LIST command) day is the day of the month when the job file was queued (printed by the OPR JOB LIST command). It is optional, except when two or more job files have the same job file sequence number.
Example:	MCR>OPR HOLD 19

4.2.9 RELEASE: Release a Job File After HOLD

To RELEASE a job file that has been prevented from running by a HOLD command or that had initially been queued with a HOLD specified, the operator can type a command in the format:

Form:	OPR RE[LEASE] n [day]
Where:	n is the job file sequence number (printed by the OPR JOB LIST command) day is the day of the month when the job file was queued (printed by the OPR JOB LIST command). It is optional, except when two or more job files have the same job file sequence number.
Example:	MCR>OPR RE 13 26

4.2.10 EXIT: Cause BATCH to Exit

The operator can request that BATCH exit after finishing the current job file by typing a command in the format:

Form:	OPR EX[IT]

4.2.11 FORCE: Force a Job File to Run Next

The operator can FORCE a job file to run next by typing a command in the format:

Form:	OPR FO[RCE] n [day]
Where:	n is the job file sequence number (printed by the OPR JOB LIST command) day is the day of the month when the job file was queued (printed by the OPR JOB LIST command). It is optional, except when two or more job files have the same job file sequence number.
Example:	MCR>OPR FORCE 238

4.2.12 CANCEL and CANCEL ALL: Cancel Job File Requests

The operator can cancel a job file request by typing a command in the format:

Form:	OPR CA[NCEL] n [day]
Where:	n is the job file sequence number (printed by the OPR JOB LIST command) day is the day of the month when the job file was queued (printed by the OPR JOB LIST command). It is optional, except when two or more job files have the same job file sequence number.
Example:	MCR>OPR CA 2 1

The entire job file request queue is cancelled by typing a command in the format:

Form:	OPR CA[NCEL] ALL

All job files and temporary files are cancelled. This command is useful when, for example, the operator wishes to go to DOS.

CANCEL does not remove requests from the job queue. Requests are removed by BATCH the next time that it chooses a job file for execution. At that time, the job file is deleted, if appropriate.

4.2.13 TLACT: Specify Action After Time Limit Is Exceeded

The operator can specify the action that BATCH should take when a job file exceeds its time limit, by typing a command in the format:

Form:	OPR TL[ACT] option
Where:	option is A, S, K, R or I (explained below)
Example:	MCR>OPR TL R

Options A, S and K stand for abort, stop and kill, respectively. Normally, TLACT prints a warning message and, one minute later, terminates the job file in the specified manner. Option R (report) specifies that only the warning message should be printed. Option I (ignore) causes time limits to be ignored. Option K should normally be selected.

4.2.14 STOP: Terminate The Current Job File

The operator can terminate the current job file when the current task completes by typing a command in the format:

Form:	OPR ST[OP]

This command allows the current task to complete, then causes BATCH to skip to the next job file. $ERROR lines are not recognized.

4.2.15 KILL: Terminate The Current Job File

The operator can terminate the current job file by typing a command in the format:

Form:	OPR KI[LL]

This command allows the current task to complete. BATCH then skips to the next $ERROR line and resumes normal processing until a $JOB line, $END line or other job terminator is reached. Subsequent jobs in the same job file are not executed and $ERROR lines in subsequent jobs are not recognized. If no $ERROR line is found, the action taken is equivalent to OPR STOP.

If a job has not yet begun execution, the OPR KILL command is equivalent to OPR STOP ($ERROR is not recognized). OPR KILL is the preferred way to terminate a job file.

4.2.16 MORE: Allow Extra Processing Time

If a job file exceeds its time limit, the operator can request more time for it by typing a command in the format:

Form:	OPR MO[RE] [time]
Where:	time is the decimal number of minutes to add to the time limit. If time is omitted, the default is to double the currrent time limit.

An OPR MORE command with no argument doubles the time limit. An argument is taken as decimal minutes and is added to the time limit. Time limits larger than 262143 seconds (about three days) disable time limit checking.

The clock starts when the job file starts. After the estimated job time has elapsed, a warning message is printed on the operator console (unless an OPR TLACT I command has been issued). One minute after the warning message is printed, the action specified in the OPR TLACT command is performed.

4.2.17 ABORT: Terminate the Current Job File

The operator can immediately terminate a job file by typing a command in the format:

Form:	OPR AB[ORT]

This command is similar to OPR KILL, except that the current task is terminated immediately.

4.3 ERROR MESSAGES

Errors in OPR commands cause the messages in Table 4-3 to be printed on the operator console.

Table 4-3
OPR Error Messages

Message	Meaning
JOB NOT QUEUED	Job file request specified in a FORCE, HOLD, RELEASE or CANCEL command cannot be found.
ILLEGAL ARGUMENT	Error in the specification of a job file request in a FORCE, HOLD, RELEASE or CANCEL command.
FORMAT ERROR	Job file request specification is missing in a FORCE, HOLD, RELEASE or CANCEL command.
BATCH NOT RUNNING	BATCH is not running.
BATCH SYSTEM ERROR nnn	See Appendix B.
BATCH ALREADY ACTIVE	BATCH is already running. The command is ignored.
BATCH RESOURCE FAILURE	BATCH is not in the system.
TASK NOT AVAILABLE	Scheduling task (SC.OPR) is not in the system.
TWO JOBS SAME NUMBER	A job file sequence number given in a FORCE, HOLD, RELEASE or CANCEL command needs a date specifier.

CHAPTER 5

SYSTEM MANAGEMENT

5.1 ORGANIZATION OF THE ACCOUNT FILE

BATCH maintains an account file as a disk-resident data set named "USERS RSX". It maintains both current and historical information on the use of the RSX system. The file contains the following general information:

- Current user account number
- Account period starting date and time

For each account, it also records the number of jobs run and the total time used.

The file provides space for 100 accounts. Account numbers 1 to 99 are reserved for users. Account 100 is used by all runs processed with incorrect $JOB lines. Incorrect $JOB information prevents the operator log from keeping track of who is using the machine. Therefore, whenever a job account cannot be determined, the job is assigned to account 100 and a warning message is printed on the operator console.

Account numbers are most meaningful when assigned on a project basis. A block of numbers can be assigned to the project and a separate number within the block can be assigned to each programmer/user on that project. This also helps the operator contact the proper individual when a job gets into trouble.

The account file is updated by JOB... and END.... In addition, two TDV commands are provided to facilitate account file management: ACI and ACD. ACI permits account file creation, initialization and modification. ACI records the date and time in the account file when the file is initialized (at the start of the current accounting period). ACD prints the contents of the account file, showing all accounts that have been used. Because ACD does not modify the account file, the file can be printed as often as desired.

ACI

5.2 ACI: INITIALIZE THE ACCOUNT FILE

The ACI TDV Function task initializes the account file for batch processing. It permits the system manager to create an account file, to edit the account file, or to reset all usage data to zero at the beginning of a new accounting period. The current time and date are set according to the system. It is advisable to verify that they are correct or to change these numbers using the ETI MCR Function task. ACI may be used only when BATCH is not running, otherwise, errors can result from simultaneous account file updates.

The ACI TDV command has the format:

Form:	ACI∇
Example:	TDV>ACI

ACI responds with a request for a password:

```
ENTER PASSWORD:
```

The correct password in the system as distributed is RSX. The system manager should change this password to suit the processing environment and to guarantee the integrity of the accounting data. This change can be made in the source code of ACI, which is written in FORTRAN.

After the password is entered, ACI overprints it and checks its validity. An invalid password causes ACI to exit. If the password is correct, ACI continues with the questions that follow.

If no account file currently exists, ACI asks whether one should be created:

```
CREATE NEW ACCOUNTING FILE (YES/NO)?
```

A reply of "NO" causes ACI to exit with no action taken. A reply of "YES" causes the account file to be created and initialized. ACI prints an appropriate message at the completion of each action.

If an account file already exists, ACI asks whether it should be initialized (i.e., whether all accounts should be reset to zero):

```
RESET ALL ACCOUNTS (YES/NO)?
```

A reply of "YES" causes ACI to initialize the account file, print an appropriate message and exit. A reply of "NO" allows the system manager to edit individual accounts. ACI asks for an account number:

```
ENTER ACCOUNT NUMBER TO BE EDITED, OR 0 TO EXIT:
```

The user responds with an account number in the range 01 to 100. ACI prints the current number of runs and the time used (in seconds), then prints:

```
ENTER NEW VALUES:
```

The user must type separate lines to enter the revised number of runs and the revised amount of time used. To keep the current number of one or both values, the user must still enter two numbers.

The process of requesting an account number and editing the stored values repeats until an account number of zero is entered, at which time ACI exits.

ACI uses LUN 10 to access the account file. The file is usually stored on the system disk in the RSX UFD.

ACD

5.3 ACD: DISPLAY THE ACCOUNT FILE

The ACD TDV Function task permits the system manager to list the account file. The system manager can then identify the users of the batch system and the amount of time used by each. ACD overcomes the need for detailed checking of operator logs. The ACD summary listing also allows the manager to evaluate the impact of any system changes. Listings can be requested as often as desired, because they do not affect the account file in any way.

The ACD TDV command has the format:

Form:	ACD▽
Example:	TDV>ACD

The summary appears on LUN-16. It lists the following information concerning the entire account file:

- Date and time of the start of the accounting period
- Date and time of the end of the accounting period
- Total number of jobs processed
- Total time used (in seconds)

The summary also gives information on each account actually used, so only nonzero numbers appear. The summary of each account has the following information:

- Account number
- Number of runs during accounting period
- Time used (in seconds)

This part of the summary includes information on runs made to invalid accounts.

ACD uses LUN 10 to access the account file. The file is usually stored on the system disk in the RSX UFD.

CHAPTER 6

BATCH OPERATIONS

6.1 BATCH SYSTEM ORGANIZATION

The Batch System of XVM/RSX comprises the following components:

- The QUEUE TDV Function task (QUE...)
- The Executive QJOB directive
- The Batch Processor (BATCH)
- The Job Startup Processor (JOB...)
- The Job Termination Processor (END...)
- The MCR OPR command (...OPR)
- The OPR task used to implement the OPR SCHEDULE command (SC.OPR)
- The ACI TDV Function task (ACI...)
- The ACD TDV Function task (ACD...)

The names in parentheses are the task names of the respective components. The following sections provide descriptions of the individual components and how they interact.

6.1.1 QUEUE TDV Function Task (QUE...)

The QUEUE command is the main user interface with the Batch System, providing a convenient mechanism for submitting jobs. QUEUE determines all of the flags and values that affect job scheduling, including such items as run time limit, priority class, memory requirements and operator requirements. This information is obtained from the QUEUE command line and by scanning the job file for $JOB lines. For each parameter, the first value found is used. In this way, the command line overrides $JOB lines, the first $JOB line overrides the second, and so on. The job information is encoded into a QJOB directive CPB and the job is submitted to the system.

QUE... processes the entire $JOB line, except for the user account number. BATCH does no $JOB processing at all. For this reason, when the NCK option is used with QUEUE, all scheduling options on $JOB lines are ignored.

6.1.2 Executive QJOB Directive

The QJOB directive adds a node to the job queue and copies the CPB information into it. I/O device specifications are translated from a LUN number to a device name, unit number and UFD name. This prevents subsequent LUN reassignments from affecting the job. Finally, a job file sequence number is assigned. Sequence number 1 is assigned to the first job submitted on a particular date, 2 to the second job, and so on. The sequence number, taken in conjunction with the submittal date, is a unique job ID. The QJOB directive event variable is set to the job file sequence number.

6.1.3 Batch Processor (BATCH)

BATCH emulates a user at a terminal. TDV treats BATCH as though it were an additional unit of the terminal handler.

After selecting a job for execution, BATCH simulates a CTRL/T and responds to the TDV login prompter. BATCH then enters commands and data, just as a user might. The only difference is that BATCH gets the command and data lines from the job file.

BATCH is controlled by the MCR OPR command. OPR interfaces with BATCH via a Batch control vector, whose address is stored in SCOM.

BATCH prints several messages (primarily error messages) on the operator log or listing device. The bulk of the operator log, however, is printed by JOB... and END..., which also print the job header and trailer pages. BATCH provides extensive information describing the job to these tasks. For details of the job information format, refer to the source code listing of JOB.nn SRC and END.nn SRC.

6.1.4 Job Startup Processor (JOB...)

JOB... determines the job account number, stores the number in the account file, prints the job header page and prints the operator log. This is done using information provided by BATCH in the form of input lines that JOB... reads. One line is an image of the $JOB line, from which the account number is determined. The format of the other lines is documented in the source code listing of JOB.nn SRC.

BATCH invokes JOB... at the beginning of every job. JOB... is written in FORTRAN to allow easy alteration by the system manager.

6.1.5 Job Termination Processor (END...)

END... prints the job trailer page, prints the operator log and updates the account file to reflect the job just run. The account number of the job is obtained from the account file, where it was stored by JOB.... The other information used is provided by BATCH in the form of input lines that END... reads. The format of these lines is documented in the source code listing of END.nn SRC.

BATCH invokes END... at the end of every job. END... is written in FORTRAN to allow easy alteration by the system manager.

6.1.6 MCR OPR Command (...OPR)

OPR provides operator control of the Batch System. OPR controls a table called the Batch control vector within BATCH. BATCH examines this table and reacts accordingly. BATCH stores the base address of the Batch control vector in an SCOM word.

The functions of OPR are fully described in Chapter 4.

6.1.7 OPR SCHEDULE Command (SC.OPR)

The OPR SCHEDULE command task is an overlay to OPR. The functions of SCHEDULE are fully described in section 4.2.2.

6.1.8 ACI TDV Function Task (ACI...)

ACI allows the system manager to create, initialize and edit an account file. The functions of ACI are fully described in section 5.2.

ACI is written in FORTRAN to allow easy alteration by the system manager.

6.1.9 ACD TDV Function Task (ACD...)

ACD displays the contents of the account file. The functions of ACD are fully described in section 5.3.

ACD is written in FORTRAN to allow easy alteration by the system manager.

6.2 I/O FUNCTIONS

BATCH implements the I/O functions HINF, READ, WRITE and ABORT. ATTACH and DETACH are also accepted (the event variable is set to +1), but are otherwise ignored.

6.2.1 HINF

For HINF, BATCH sets the event variable to +300021 to indicate:

Bit	Contents
0	Set to 0 to make the event variable positive
1-2	Set to 3 to indicate an input and output device
3	Set to 0 to indicate a non-directory-oriented handler
4-11	Device unit 0
12-17	Device code 21

6.2.2 READ

BATCH satisfies a READ function by using a line from the job file. The event variable is always set to +2 (as though the request line were terminated with a carriage return), regardless of the line terminator. Input lines are restricted to 132 characters. Only the IOPS ASCII data mode is supported for BATCH input.

In response to a read request from TDV, BATCH always supplies a control line. The first character of each control line is $. An exception to this is $EOF, which is considered a data line. Intervening data lines are skipped (ignored) until a control line is found.

In response to a read request from any other task, BATCH supplies a data line. If any task other than TDV attempts to read a control line, an end-of-file indication is returned and the requesting task is aborted.

Exceptions to the previous rules are $LOG, $MSG, $PAUSE and $EJECT lines. Each of these is processed internally by BATCH and can be intermixed with both control and data lines.

6.2.3 WRITE

BATCH normally satisfies a WRITE function by transmitting output lines to the listing device. Only the IOPS and Image ASCII data modes are supported for BATCH output. IOPS ASCII lines are restricted to 132 characters. Image ASCII lines are restricted to 80 characters. Longer lines are truncated.

A special feature of the BATCH WRITE function allows selected output lines to be omitted from the listing file. This feature is used to avoid printing command prompters and other support messages that are of no value in a BATCH job listing. TDV uses this feature to avoid printing command prompters ("TDV>") on the job listing.

Use of the selected-output feature is controlled by the sign bit in the line buffer header word (the word containing the word-pair count). If the sign bit is zero (the normal case), the output line is transmitted to the listing device. If the sign bit is set to one, the output line is not transmittd to the listing device, and the write request event variable is immediately set to +1.

When transmitting a line to the listing device, BATCH first copies the line into an internal buffer. The request event variable is set to +1 when the copy is complete. Listing errors are not returned to the user task. They are handled by BATCH. BATCH responds to a listing error by aborting the current task and immediately terminating the current job file. The error is reported to the Job Termination Processor (END...), which prints an error message on the operator console. END... also attempts to print a job trailer page containing the error message.

6.2.4 ABORT

Both abort-single-unit, issued by tasks, and abort-all-units, issued by I/O RUNDOWN, are supported.

6.3 JOB SELECTION

After initializing itself, BATCH scans the list of job requests. For each request that passes certain logical tests, BATCH computes a "priority" using a formula given below.

6.3.1 Priority Calculation

Job priority has different implications for batch processing and for ordinary RSX scheduling. Listed below are the elements of the priority calculation. They can vary with each job. All time parameters are in minutes.

Parameter	Meaning
Waittime	Delay between now and when job was queued.
Timelimit	Estimated run time of the job as specified in the $JOB line, QUEUE command line or QJOB CPB.
Class	A general parameter indicating who is requesting the run.
FRCFLG	An indicator set to show whether this job must run next.
OPRFLG	An indicator set to show whether this job needs operator assistance.
Memsiz	A quantity in the range 1 to 128 indicating the minimum TDV partition size that must exist when this job runs.
SEQFLG	An indicator set to show that this job cannot run until previously submitted and sequenced jobs have run.
HLDFLG	An indicator set to prevent this job from running until cleared by the OPR RELEASE command.

Waittime has a maximum value of 1440 minutes, or one day. If a job has been waiting more than one day, one day is used for Waittime.

Timelimit is evaluated in reverse log(2) form:

(10 - log(2) (runtime)) * Tfactr

so that 1 minute becomes a Timelimit of 10 and 1023 minutes becomes a Timelimit of 0. Tfactr (explained below), therefore, favors the short job. The log form is used to increase discrimination between jobs with short run times.

The OPR SCHEDULE command sets the additional parameters below and should be used when BATCH is first invoked. The same set of values for these parameters then applies to all jobs.

Parameter	Meaning
Timmax	Longest job to be run at this time of day.
Clsmin	Lowest class job to be run at this time of day.
Waitmax	Longest time a job should wait, regardless of characteristics.
Wfactr	A multiplier applied to Waittime.
Tfactr	A multiplier applied to Timelimit.
Cfactr	A multiplier applied to Class.

The OPR ON command sets the Opon parameter while BATCH is running. Opon is an indicator set to show whether operator assistance is available.

The logical tests that follow are performed in the order listed before BATCH attempts to calculate a priority.

Test	Action if True	Action if False
1. HLDFLG set	Task will not run	Test 2
2. OPRFLG set	If Opon equals 1, test 3; otherwise, task will not run	Test 3
3. Memsize>maximum TDV partition size	Task will not run	Test 4
4. SEQFLG set	If any previously submitted and sequenced task has not run, this task will not run; otherwise, test 5	Test 5
5. FRCFLG set	Task runs next	Test 6
6. Timelimit>Timmax	Task will not run	Test 7
7. Class<Clsmin	Task will not run	Test 8
8. Waittime>Waitmax	Task runs at maximum priority (131071)	Task runs at the priority computed by formula

All waiting jobs go through these logical tests. If none can run, the tests are repeated after a short delay. A priority is calculated for each job that goes through these tests without being rejected or selected to run next. The formula for computing priority is:

```
Priority =  (Cfactr * Class)
            +
            (Wfactr * Waittime/262144)
            +
            (Tfactr * Timelimit)
```

The priority calculated by this formula must be in the range 1 to 131071. If the formula yields a result of zero, a priority of 1 is used. If the formula yields a result greater than 131071, the value 131071 is used.

6.3.2 Other Selection Factors

From the job request list, BATCH selects the job request having the highest priority. If two priorities are the same, the job that has been waiting longest is favored. The job file name and other descriptive information is taken from the job queue node. The job is then run.

6.4 JOB FILES

BATCH assigns a priority to each job file. The job file is the unit of work submitted by the QUEUE command or QJOB directive. A job file is a file on file-oriented devices, or a sequence of input lines (appropriately terminated) on other devices. A job file is terminated by the first occurrence of an end-of-file indicator, a $END card, a $QUIT card or an I/O error.

Job files from non-file-oriented devices are normally stacked on a disk. This function is performed by the QUEUE command. BATCH has been optimized for this mode of operation.

6.4.1 Multiple Jobs

A job file can contain several jobs by separating them with $JOB lines. The entire job file is treated as a unit for scheduling and time-limit purposes. Each job has separate header and trailer pages, and is separately entered in the account file. If any job in a job file is terminated abnormally (i.e., for any reason other than reading a $JOB line), the remaining jobs in the job file are not run.

6.4.2 Printing Control Lines

Most control lines (those with $ as the first character) are printed on the listing device only. Exceptions to this are $JOB, $END, $QUIT and $EJECT lines, which are not printed anywhere. The effects of these lines, however, are visible in job listings and the operator log. $MSG and $PAUSE lines are printed on both the listing device and operator log. Data lines are not printed anywhere.

APPENDIX A

BATCH SYSTEM INSTALLATION CHECKLIST

Before installing the Batch System, the system manager must choose several LUNs and UFDs to be used. Most of these are used by BATCH and are assembly parameters to module BDRES. Others are specified when assembling SC.OPR.

BATCH requires two LUNs for its exclusive use, plus a third LUN for the operator log. The exclusive LUNs are used to access the input and listing devices. BATCH assigns appropriate devices to these LUNs for each job file run. LUNs 6 and 7 are normally used for this. The operator log is normally printed on the MCR terminal via LUN 3. In a heavily used system, it may be desirable to use a different LUN and dedicate a terminal printer to the operator log. LUN assignments are specified using the JOBLUN, LSTLUN and OPRLUN assembly parameters to BDRES.

BATCH must know the location of the account file so that appropriate device assignments can be made for JOB... and END.... The account file device name and unit are specified using assembly parameter ACCTDEV to BDRES. The UFD is specified using assembly parameter ACCTUFD. The default location of the account file is the system disk, unit zero, UFD RSX. The account file name is defined by a DATA statement in source code files JOB.nn, END.nn, ACI.n and ACD.n.

The only other LUN assignment required is for the OPR SCHEDULE command. The LUN used to access schedule files is specified using the SCHLUN assembly parameter to SC.OPR. The default LUN is 8. The MCR REASSIGN command must be used to assign the correct disk and UFD to this LUN.

After assigning the previously described parameters, install the Batch System using the following checklist. Assembly and task building instructions are given in Parts III and XII of this manual.

1. Verify that the RSX Executive was assembled to allow batch processing. This is a question asked during the build procedure.

2. Create a partition named BATCH of size 3400 (octal). It must be within the lowest 32K of core.

3. Verify that a TDV partition of 9K or larger exists.

4. Build and install QUE....

5. Build and install BATCH.

6. Build and install JOB....

7. Build and install END....

8. Build and install ...OPR.

9. Build and install SC.OPR.

10. Build and install ACI....

11. Build and install ACD....

12. Build and install SLI....

13. Verify that LUNs 6 and 7 are assigned to NONE, and that LUN 8 is assigned to the location of the schedule files. Modify this step appropriately if LUN assignments were changed with assembly parameters.

14. If accounting records are to be kept, create an account file using ACI.

15. Using the OPR BATCH command, load BATCH.

16. Using the OPR SCHEDULE command, set scheduling parameters.

17. Using the OPR GO command, start BATCH.

18. Using the QUEUE command from any TDV terminal, submit jobs to BATCH.

APPENDIX B

BATCH SYSTEM ERRORS

B.1 GENERAL

BATCH contains extensive error-checking code to ensure reliable operation. Errors that affect only the current job cause the job to be terminated. Errors are reported to the Job Termination Processor (END...), which prints them on the operator log and job trailer page.

Other errors, particularly those that prevent all jobs from being processed, are referred to as Batch System errors. Batch System errors abort batch processing and are reported on the operator log in the form:

```
BATCH SYSTEM ERROR nnn
error description
```

where nnn is a code identifying the source of the error. The error description provides useful information to system programmers familiar with BATCH. After printing the message, BATCH disconnects from the PDVL node and exits.

If an I/O error is detected while BATCH is printing an error message, BATCH arranges for OPR to report the message. The error code number is saved and subsequently printed by OPR. In this event, however, the error description is not printed.

Most Batch System errors are in response to some major inconsistency in the operation of BATCH, the TDV Poller or some other component of the MULTIACCESS feature. These errors are meaningful only to someone knowledgeable in the internal implementation of the components. These errors should be treated as software faults and be reported via the standard SPR procedure. Copies of the operator log, job file and job listings should be submitted with the SPR.

Some Batch System errors can result from incorrect software installation or related errors. These errors are listed in the following section with appropriate corrective actions.

B.2 RECOVERABLE BATCH SYSTEM ERRORS

The following list describes all Batch System errors from which the user can recover. Any errors not in this list should be reported via the standard SPR procedure.

```
BATCH SYSTEM ERROR 102
LUN nn (INPUT LUN) MUST BE ASSIGNED TO NONE
```

Explanation: BATCH requires a dedicated job file input LUN, as it changes the device assigned to it.

Correction: Use the MCR REA command to assign the LUN to NONE, then rerequest BATCH.

BATCH SYSTEM ERROR 103
LUN nn (LISTING LUN) MUST BE ASSIGNED TO NONE

Explanation: BATCH requires a dedicated listing LUN, as it changes the device assigned to it.

Correction: Use the MCR REA command to assign the LUN to NONE, then rerequest BATCH.

BATCH SYSTEM ERROR 104
LUN nn (OPERATOR TTY) MUST BE ASSIGNED TO A TTY

Explanation: BATCH requires a LUN on which to print the operator log. This message is printed if the LUN is assigned to NONE rather than a suitable terminal.

This message cannot be printed on the operator log. It can be printed only by OPR. Therefore, the error description line is not printed.

Correction: Use the MCR REA command to assign the LUN to a terminal or other listing device.

BATCH SYSTEM ERROR 201
BATCH REQUIRES A 9K OR LARGER TDV PARTITION

Explanation: JOB... and END... each require a 9K partition. A partition this size or larger must be provided.

Correction: Use the MCR RCF or RCP command to create a TDV partition of sufficient size. TDV partitions have the letters "TDV" as the first three characters of the partition name.

BATCH SYSTEM ERROR 301
ASSIGN FAILURE--CANNOT ASSIGN LP

Explanation: BATCH requires the use of a line printer (device name LP) as a listing device. This message appears if BATCH is requested and the line printer is not available.

This error is detected after a job has already been removed from a job request queue. That job request is irretrievably lost.

Correction: Choose one of the following:

1. Don't use BATCH.

2. Modify the device handler for some other device to act as device name LP.

3. Modify BATCH modules BDOPEN and BDMSSG to use some other device as the default listing device.

In general, recovery from Batch System errors should be done by restarting BATCH using the OPR BATCH command. It is frequently necessary to assign the BATCH input and listing LUNs to NONE.

If BATCH aborts because of a bad CAL or other unexpected cause, a Batch System error should deliberately be caused in order to properly reinitialize the BATCH PDVL node. Batch System error 102 can be caused by assigning a device to the BATCH input LUN and requesting BATCH.

INDEX