PERKIN-ELMER

OS/32 FASTCHEK

Reference Manual

48-064 F02 R00

The information in this document is subject to change without notice and should not be construed as a commitment by The Perkin-Elmer Corporation. The Perkin-Elmer Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license, and it can be used or copied only in a manner permitted by that license. Any copy of the described software must include the Perkin-Elmer copyright notice. Title to and ownership of the described software and any copies thereof shall remain in The Perkin-Elmer Corporation.

The Perkin-Elmer Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Perkin-Elmer.

The Perkin-Elmer Corporation, Data Systems Group, 2 Crescent Place, Oceanport, New Jersey 07757

Printed in the United States of America

TABLE OF CONTENTS

CHAPTERS (Continued)

PREFACE

This manual describes the functions and features of the OS/32 FASTCHEK program, Part Number 03-344. OS/32 FASTCHEK is used to check the integrity of disk packs and to initialize and rename packs in a fast, efficient and convenient manner.

User Prerequisites
Users of OS/32 FASTCHEK should be familiar with the operation of OS/32.

Synopsis of Chapters
Chapter 1 provides a general overview of the program's capabilities. Chapter 2 describes when to run FASTCHEK and how to load and start it.

Chapter 3 describes command entry and Chapter 4 contains a description of the operation of FASTCHEK and tuning information.

Chapter 5 discusses error handling and documents all the messages that can be generated. Internal failure conditions are discussed in Chapter 6.

Appendixes provide: a command summary; End of Task codes; device characteristics; format of the Pack Administration file; Link procedure; logical unit usage; a comparison with OS/32 DISCHECK and OS/32 DISCINIT; notes on the journal feature; and notes on the compatibility with other products.

System Requirements
OS/32 FASTCHEK executes as a segmented user task under OS/32 R06.2 or higher. R07.2 and higher support 25 MB CDD50 disk.

OS/32 FASTCHEK functionally replaces the OS/32 DISCHECK and OS/32 DISCINIT programs. Current users of these products will find a comparison of FASTCHEK with DISCHECK and DISCINIT commands in Appendix G.

For information on the contents of all Perkin-Elmer 32-bit manuals, see the 32-Bit Systems User Documentation Summary.

fd — is the file descriptor of the device or file containing the task image load module of FASTCHEK. If this parameter is omitted, the default is "taskid.TSK".

segsize increment — is a decimal number in kb specifying the additional main memory workspace to be added to the module's impure segment. This value, if given, overrides the "WORK=n" option which may have been specified when the task was linked.

NOTE

FASTCHEK will generally execute faster if it is allocated more memory. For information on the choice of optimum segment size increment see Chapter 4.

2.2.2 Loading FASTCHEK from an MTM Terminal

This command loads FASTCHEK from an MTM terminal. (Note that because packs can only be marked on and off from the system console, FASTCHEK is not normally run from an MTM terminal although there is no operational reason why it cannot be.)

Format:

```
LOAD  fd  ,segsize increment
```

Parameters:

fd — is the file descriptor of the device or file which contains the task image load module of FASTCHEK.

segsize increment — is a decimal number in kb specifying the additional main memory workspace to be added to the module's impure segment. This value, if given, overrides the "WORK=n" option which may have been specified when the task was linked.

NOTE

FASTCHEK will generally execute faster if it is allocated more memory. For information on the choice of optimum segment size increment see Chapter 4.

2.3 STARTING FASTCHEK

After FASTCHEK has been loaded, the START command may be used to begin execution of the program. The disk must be marked off, and FASTCHEK must be made the current task. The format of the START command is identical in both the OS/32 System Console and MTM environments.

The START command that should be used for FASTCHEK has the general format:

```
START [,param1 [,param2 [.. [,paramn] ..]]]
```

Depending on the parameters used in the START command, FASTCHEK will commence operation in one of three possible command entry modes: interactive, batch or immediate.

Note that once FASTCHEK has validated all commands and is about to commence the requested operation, then, if the commands have been entered other than via the system console, the following message is logged to the console:

```
OS/32 FASTCHEK Rnn-nn <function> disk: STARTING
```

where <function> is one of INITIALIZE, CHECK or RENAME.

2.3.1 Starting FASTCHEK in Interactive Mode

This command is used to start FASTCHEK in Interactive Mode so that the FASTCHEK commands can be entered in a conversational manner.

Format:

```
START [COMMAND [=] [idev:] [,LIST [=] [fd]]]
```

Parameters:

COMMAND= idev: specifies the input device from which commands are to be entered and must be an interactive device. If this parameter is omitted or idev: is omitted, the default is the command entry device CON:. Note that throughout this manual all references to the device name CON: in fact refer to the name of the system console if this is other than CON:.

2.3.3 Starting FASTCHEK in Immediate Mode

This command starts FASTCHEK in Immediate Mode. This mode is identical to batch mode (in that the commands to FASTCHEK are identical), except that no command device is specified. All the commands required by FASTCHEK must be passed to FASTCHEK as arguments of the START command.

Format:

START ,cmd1, cmd2 [, cmd3 [, cmd4..]]

Parameters:

cmd1, cmd2, etc. are the FASTCHEK commands. These are discussed in Chapter 3.

Functional Details:

In Immediate Mode all commands to FASTCHEK must be passed via the START command. It is not possible to pass some commands to FASTCHEK via the START command and the remainder from a command file. Thus, for example, the following START command will be rejected:

```
START ,CHECK DSC2:,COMMAND=SCRT:FASTCHEK.CMD
```

Immediate mode is best used when the number of commands to FASTCHEK can be fitted into the START command. If the required commands cannot be fitted into the START command, then batch mode must be used.

After FASTCHEK is started, this message is displayed:

```
OS/32 FASTCHEK Rnn-nn
```

where nn-nn gives the revision and update level of FASTCHEK. The commands passed in the START command are then processed.

Examples:

```
START ,INIT DSC1:,VOL=SCRT,READCHECK,WRITEREC

ST ,CHECK=DSC4:,LIST PRIN:

ST ,RENAME DSC2:,V=GA04
```

2.4 STOPPING FASTCHEK

Once FASTCHEK has processed all its initial commands, it commences operation. If the user desires to halt FASTCHEK after it has begun execution, he may use the SEND command of OS/32 to direct a message to FASTCHEK. The two messages which FASTCHEK will recognize are STOP and PAUSE.

Note that if FASTCHEK is being executed outside the MTM environment then it must be selected as the current task by the OS/32 TASK command before the SEND command is entered.

2.4.1 SEND STOP

This command instructs FASTCHEK to terminate in an orderly manner. Note that if some operation has been started then the SEND STOP command will not leave the pack in its original state. Thus if an Initialize operation is mistakenly started on the wrong pack, then using SEND STOP will not prevent the destruction of the data on the pack.

Format:

SEND STOP

Functional Details:

This command, when received by FASTCHEK instructs the program to make an "orderly" shutdown. To effect its shutdown, FASTCHEK will wait for any outstanding I/O operations to complete, close any open logical units, and delete any workfiles currently being used.

Regardless of the initial command mode (i.e., interactive, batch or immediate), reception of the SEND STOP command will cause FASTCHEK to terminate.

Note that the use of this command is preferred to simply cancelling FASTCHEK because it results in a more orderly shutdown.

2.4.2 SEND PAUSE

This command is used to instruct FASTCHEK to pause so that the operator can perform some required action.

3.2.1 INITIALIZE

The INITIALIZE command is used to select the Initialize function and also to specify the disk device containing the pack to be initialized.

Format:

INITIALIZE [=] devn:

or INITIALISE [=] devn:

Parameters:

devn: is the device name of the disk drive containing the pack to be initialized.

Functional Details:

The pack on the specified drive will be initialized in the selected mode (FILL, READCHECK or NOREADCHECK). Note that if no mode is specified, then NOREADCHECK is used as the default. The Volume Descriptor, Bit Map and Directory will be initialized and written. The Pack Administration file will be created in the FILL or READCHECK modes and will be updated in the NOREADCHECK mode. Automatic mode switching from NOREADCHECK to READCHECK will occur if the Pack Administration file does not exist or is invalid. For further details see Chapter 4.

The specified device must be currently ready, not write protected, and marked off.

NOTE

It should be noted that if a pack is to be initialized in order to clear all currently existing files from the pack, it is most efficient to use the NOREADCHECK mode. This mode is also preferred because it preserves the Pack Administration file. Moreover, if the user wishes to both initialize a pack and also perform a surface check, and the pack currently contains a Pack Administration file, then this is best done by first initializing the pack in NOREADCHECK mode and then checking it in READCHECK mode, since this will preserve the Pack Administration file.

Examples:

```
INIT DSC1:

INITIALIZE FLP3:

INIT=D5FX:
```

3.2.2 CHECK

The CHECK command is used to select the Check function and also to specify the disk device containing the pack to be checked.

Format:

CHECK [=] devn:

Parameters:

devn: is the device name of the disk drive containing the pack to be checked.

Functional Details:

An integrity check will be performed on the pack mounted on the specified drive. The extent of the checking performed is determined by the mode selected (READCHECK, NOREADCHECK, CLOSE or CLOSEONLY). Note that if no mode is specified, then the default is CLOSE. The Volume Descriptor and Directory are always checked. NOREADCHECK forces a check of the file allocation and access paths, and READCHECK causes a surface check to be performed as well as the file check. Mode switching from CLOSE to NOREADCHECK to READCHECK will occur automatically if found necessary, but no mode switch will occur if CLOSEONLY is selected.

Further operational details can be found in Chapter 4.

The specified device must be currently ready and marked off. The drive must not be write protected unless the REPORTONLY option is used, in which case it may be write protected if desired.

Examples:

```
CH DSC1:

CHECK FLP3:

CHE=D5FX:
```

3.2.3 RENAME

The RENAME command is used to select the Rename function and also to specify the disk device containing the pack to be renamed.

Format:

RENAME [=] devn:

Parameters:

devn:	is the device name of the disk drive containing the pack to be renamed.

Functional Details:

The pack on the specified drive will be renamed by changing the volume name in the Volume Descriptor to the specified name. Note that the Volume Descriptor is checked to be valid (see section 4.1.3) before effecting the rename. Note also that a warning message is output if an attempt is made to rename a pack to its current name.

The specified device must be currently ready, not write protected, and marked off.

Examples:

```
REN DSC1:

RENAME FLP3:

RENA=D5FX:
```

3.2.6 FILL

The FILL command selects the mode in which an Initialize function is to be performed. In an Initialize/Fill operation, the specified data pattern is written to every sector on the pack before carrying out a surface check and then initializing the pack.

Format:

FILL [=] [xxxxxxxx]

Parameters:

xxxxxxxx	is a string of up to eight hexadecimal digits to be used as the fill data pattern for each fullword in every sector on the pack. If less than eight digits are specified, then leading zeros are assumed. If no data pattern is entered, then a fill pattern of 00000000 is used.

Functional Details:

In an Initialize/Fill operation the pack is first filled with the specified data pattern. The operation then proceeds as for Initialize/Readcheck; that is, a surface check is performed, the Pack Administration file is created and used to record the defective sector addresses, the Bit Map is allocated and initialized, the Directory is allocated and initialized, and the Volume Descriptor is written.

Examples:

```
FILL                    (data pattern set to 00000000)

FIL = BDBDBDBD

F = 5555                (data pattern set to 00005555)
```

3.2.7 CLOSE

The CLOSE command selects the mode in which the Check function is to be performed. In a Check/Close operation the integrity of the Directory is checked and any open files which can be safely closed are closed.

Note that if no mode is specified for a Check operation then Check/Close is selected.

Format:

CLOSE

Parameters:

None.

Functional Details:

In a Check/Close operation an integrity check of the Directory is performed. Any Contiguous file found to be open is closed and any Indexed file open only for read is closed. If any Indexed file is found to be open for write or the integrity check of the Directory fails then the NOREADCHECK mode is automatically selected and a Check/Noreadcheck operation is performed.

The CLOSE mode is the preferred mode for the Check operation since only the minimum required checking is performed. If however, the user knows that the pack contains Indexed files open for write, and thus that a mode switch to NOREADCHECK will occur, it is slightly more efficient to initiate the operation in NOREADCHECK mode.

Examples:

```
CLOSE

CLO
```

Examples:

WR

WRITERECOVERY

WRITEREC

NOWR

NOWRIT

NOWRITERECOVERY

3.2.13 BLOCKS and DIRECTORY

The BLOCKS and DIRECTORY commands are used to set the size and position of the Directory allocated during the Initialize operation. The two commands are equivalent except that they set the Directory size in terms of blocks and files respectively. Only one of the two may be used, and, if neither is entered, default values are used.

Format:

BLOCKS [=] [bbb] [/ [ccc]]

DIRECTORY [=] [fff] [/ [ccc]]

Parameters:

bbb	gives the required size of the Directory in terms of the decimal number of directory blocks.
ccc	gives the decimal number of the cylinder on which the Directory is to start (counting the first cylinder as zero).
fff	gives the required size of the Directory in terms of the decimal number of files it can contain.

Functional Details:

Each directory block occupies 1 sector and can contain up to 5 file entries. Thus the commands

BLOCKS = 100/1 and DIRECTORY = 500/1

are equivalent. If the size of the Directory is not specified, then a default value appropriate to the type of pack being initialized is used. Selected examples are given below. A full list is given in Appendix C.

Disk Type	Default Blocks	Default Files
256 MB	320	1600
67 MB	128	640
25 MB	64	320
5 MB	24	120
Floppy	1	5

If the starting cylinder number is not specified or is specified as zero, the Directory is located starting on cylinder 1 (i.e., the second cylinder). Cylinder zero is already partly allocated for the volume descriptor sector.

If the area required by the Directory is found to contain defective sectors, the Directory is relocated to the next available error free area (of the required size) that starts on a track boundary.

When the Directory is allocated, the successive directory blocks are not on successive sectors but are on each Nth sector where the value varies depending on the type of disk. Selected examples are given below (see also Appendix C).

Disk Type	Block Sequencing
256 MB	every 32nd sector
67 MB	every 32nd sector
25 MB	every 31st sector
5 MB	every 6th sector
Floppy	every 2nd sector

Thus, on a 256 MB pack, (which has 64 sectors per track) successive blocks are allocated on sectors 0, 32, 1, 33, 2, 34, ...29, 61, 30, 62, 31 and 63 and then on the same sectors on the next track. This allocation technique optimizes the time required by the Operating System to scan the Directory.

It should be noted that during initialization the Pack Administration file is always created. If the file did not previously exist, the minimum Directory size is one block. If zero blocks are requested, one will be allocated.

It is important to realize that although it is valid to request the allocation of a Directory that is smaller than is required for the number of files that will be allocated on the pack (since the Operating System will automatically extend the Directory), the Check function of FASTCHEK will execute considerably faster if the Directory is initialized to be large enough to hold all files to be placed on the pack.

It will be apparent that, for a given type of pack, there is a 'reasonable' upper limit to the Directory size. The maximum allowed size for the Directory is taken to be one eighth (1/8) of the total pack size. This represents over 500,000 files on a 256 MB pack.

Examples:

```
B = 300/100
DIR = 1500/100

BLOCK                (implies default values)
DIRECTORY

BLOC = /             (implies default values)
DIRECT = /

BLOCKS 100           (implies default start cylinder)
DIRECTORY 500

BL /20               (implies default number of blocks)
DIR /20

BLO = 0              (minumum Directory allocation)
DIRECT = 0
```

3.2.14 VOLUME

The VOLUME command specifies the volume name of the pack and is mandatory for the Initialize or Rename function (and invalid for the Check function).

Format:

VOLUME [=] voln

Parameters:

voln is the volume name.

Functional Details:

The specified volume name can be any valid volume name. That is, it cannot be longer than 4 characters, cannot contain any imbedded blanks, the first character must be alphabetic, and the remaining characters, either numeric or alphabetic. Note that lower case letters are allowed but will be translated to upper case.

It should be noted that it is unwise to use a volume name that is the same as one of the device names in the system (i.e., naming a pack DSC1 if one of the disk drives is called DSC1:) since the Operating System will not allow this pack to be marked on. It is also unwise to use names that are the same as the keywords used in the Display Devices command (e.g., OFF), since this can lead to confusion.

Examples:

```
V SCRT

VOL = work

VOLUME=OS32
```

3.2.15 LIST

The LIST command is used to specify the file or device to which messages are output. If this command is omitted, then PR: is used as the default list device unless it cannot be assigned, in which case CON: (or rather the device name of the system console) is assigned as the list device.

Format:

LIST [=] fd

Parameters:

fd	is the file descriptor of the file or device to be used as the list device.

Functional Details:

The LIST command may be entered either as part of the Operating System START command or (if batch command entry is used) as one of the commands read from the batch file. However, if batch command entry is used, then the LIST command cannot be specified in both the START command and as one of the batch commands.

If the file descriptor specifies a file and FASTCHEK is being executed in the MTM environment, then the extension can be entered as either P, G or S (or omitted, in which case P is assumed). If FASTCHEK is being executed outside the MTM environment, then the account number can be entered as a number (between 0 and 255 inclusive) or omitted (indicating account 0). If P, G or S is used, then this will be taken to mean account 0.

If a file is specified as the list device, then it must currently exist. Output to this file will be appended after any existing data.

Note that nothing is output to the list device until all commands have been processed and validated. Thus, if, for example, errors occurred while reading commands from a batch file, then the resulting error messages would not be output to the list device but to the system console (or MTM terminal).

3.3 INTERACTIVE COMMAND ENTRY

This method of command entry is provided for inexperienced and occasional users who are not familiar with the various command keywords and parameters.

The Interactive Command Entry mode is invoked when the program is started with an interactive device (i.e. a terminal) as the command device. That is, the START command has one of the following formats:

START ,COMMAND=idev: [,LIST=fd]

START [,LIST=fd]

where idev: is the device name of an interactive device. The second format results in CON: being used as the interactive command device.

When started in this mode, FASTCHEK outputs a series of prompts requesting various parameters, values or Yes/No responses. If the response to a given prompt is not valid, an error message is output, and then the prompt is redisplayed so that a valid response can be entered.

After all the questions have been answered, a message is output giving the selected function, its mode and any options. The user is then asked to confirm that this data is satisfactory. A negative response causes the complete dialogue to begin again.

The majority of the prompts will accept a carriage return as indicating that a default value is to be used. In these cases, the default response is indicated in the prompt message by a number sign character (#). For example:

Mode (#NOReadcheck, REAdcheck, or Fill=xxxxxxxx) ?

where NOREADCHECK is the default mode. Note that the default value can be explicitly selected if desired; that is, the response "NOREADCHECK" is valid in the above case. However, "#NOREADCHECK" or "#" are not.

The prompt also shows the minimum abbreviations of the allowed keyword responses in upper case with the remainder of the keyword in lower case (provided that the terminal being used supports lower case).

The dialogue is conducted in such an order that mandatory (non-defaultable) parameters are requested first. Once these have been input, the user can elect to default the remaining parameters by using the response

!GO

In this case, the dialogue is terminated and the requested operation commences immediately, thus bypassing the remaining

questions and also the confirmation step. Note that the !GO response is not allowed until all the mandatory parameters have been obtained but once this has been done the !GO response can be given to any prompt thus defaulting the remainder.

If the user wishes to change his response to a previous prompt he can cause the complete dialogue to be restarted by entering the response

!RESTART

to any prompt.

Two other special responses are recognized and can be input for any prompt. These are

!PAUSE

and

!STOP

The !PAUSE response causes the program to be paused. When it is continued the current prompt message is redisplayed. The !STOP response causes the program to be terminated in an orderly fashion with end of task code 250.

The following sections give the prompt messages used and the allowable responses. The first prompt requests the function to be performed. Since the dialogue varies depending on the selected function, separate sections are used to describe the conversation for each possible function.

Note that in these sections the functional details of each response are not documented since they have been given in sections 3.2.1 through 3.2.15 and these should be consulted if any clarification is required.

3.3.1 Dialogue for the Initialize Function

The first prompt is

```
Function (INITialize, CHeck, or RENname) Devn: ?
```

to which the response must be

INITIALIZE [=] devn:

or INITIALISE [=] devn:

to select the Initialize function and devn:, which is the device name of the drive containing the pack to be initialized.

If the Initialize function is requested, the dialogue continues with the prompt

```
Volume Name ?
```

to which the response must have the form

voln

where voln is a valid volume name.

The next prompt requests the mode in which the Initialization is to be performed. It is as follows:

```
Mode (#NOReadcheck, REAdcheck or Fill=xxxxxxxx) ?
```

If the default mode of NOREADCHECK is not to be used, then the response must be one of the following:

NOREADCHECK

READCHECK

FILL [=] [xxxxxxxx]

where xxxxxxxx is a hexadecimal number of up to eight digits. Note that default responses can be made to both the above and all remaining prompts. Thus, the special response !GO can be used to terminate the dialogue and commence execution.

The Directory allocation information is then requested using the prompt

```
Directory (#nnn Files / Cylinder #m) ?
```

where nnn and m indicate the default allocation for the type of disk previously specified. If the default values are not to be used, then the response should have the form

[fff] [/ [ccc]]

where fff is the decimal number of files that the Directory is to contain and ccc is the decimal cylinder number on which the Directory is to start.

If the NOREADCHECK or READCHECK mode was requested earlier, then the prompt

Attempt Write Recovery (#Yes or No) ?

is output. A default (null) or YES response will enable the WRITERECOVERY option. A NO response will disable it.

If the List device was not specified in the START command, the following prompt is displayed:

List Device (#PR:, @=idev: or FD) ?

where idev: is the device name of the interactive terminal being used. The default (null) response will select PR: as the list device. A response of "@" will select the terminal being used as the list device. Alternatively, any required file or device can be selected by entering its file descriptor. Note that if a file is specified, it must currently exist; the listing information will be appended to any existing data in the file.

At this point all required data has been entered and a message in the following form is output:

Initialize devn: Mode = { Fill with xxxxxxxx | Readcheck [with Writerecovery] | Noreadcheck [with Writerecovery] }

Volume voln Directory for nnn Files at Cylinder m Requested

and this is followed by the prompt

OK to Run (Yes or No) ?

If the response is NO, then the complete dialogue is restarted. If the response is YES (or !GO), then execution commences. Note that no default response is allowed to this prompt.

3.3.2 Dialogue for the Check Function

The first prompt is

Function (INITialize, CHeck, or RENname) Devn: ?

to which the response must be

CHECK [=] devn:

to select the Check function; devn: is the device name of the drive containing the pack to be checked.

If the Check function is requested, the dialogue continues with the prompt

Mode (#CLOse, CLOSEOnly, NOReadcheck or REAdcheck) ?

If the default mode of CLOSE is not to be used, then the response must be one of the following:

CLOSE

CLOSEONLY

NOREADCHECK

READCHECK

Note that default responses can be made to both the above and all remaining prompts. Thus the special response !GO can be used to terminate the dialogue and commence execution.

If the requested mode was other than CLOSEONLY, then the required settings of the EXTENDALLOWED, WRITERECOVERY and KEEPSPOOL options are solicited using the following prompts:

Extend Indexed Files (#No or Yes) ?

Attempt Write Recovery (#Yes or No) ?

Keep Aged Spool Files (#No or Yes) ?

In all cases, a YES response will enable the option and a NO response will disable it. Default (null) responses will disable EXTENDALLOWED and KEEPSPOOL but enable WRITERECOVERY.

Then, irrespective of the selected mode, the prompt

Report Only (#No or Yes) ?

is output. A default (null) or NO response will disable this option, whereas a YES response will enable the REPORTONLY option, thus preventing any modification of the current state of the pack.

If the List device was not specified in the START command, the following prompt is displayed:

List Device (#PR:, @=idev: or FD) ?

where idev: is the device name of the interactive terminal being used. The default (null) response will select PR: as the list device. A response of "@" will select the terminal being used as the list device. Alternatively, any required file or device can be selected by entering its file descriptor. Note that if a file is specified, it must currently exist; the listing information will be appended to any existing data in the file.

At this point, all required data has been entered, and a message of the following form is output:

Check devn: Mode = { Close / Closeonly / Noreadcheck / Readcheck }

with [Writerecovery] [Extendallowed] [Keepspool] [Reportonly]

and this is followed by the prompt

OK to Run (Yes or No) ?

If the response is NO, then the complete dialogue is restarted. If the response is YES (or !GO), then execution commences. Note that no default response is allowed to this prompt.

3.3.3 Dialogue for the Rename Function

The first prompt is

Function (INITialize, CHeck, or RENname) Devn: ?

to which the response must be

RENAME [=] devn:

to select the Rename function; devn: is the device name of the drive containing the pack to be renamed.

If the Rename function is requested, the dialogue continues with the prompt

Volume Name ?

to which the response must have the form

voln

where voln is a valid volume name.

If the List device was not specified in the START command, the following prompt is displayed:

List Device (#PR:, @=idev: or FD) ?

where idev: is the device name of the interactive terminal being used. The default (null) response will select PR: as the list device. (Note that the !GO response to this prompt will also select PR: as the list device.) A response of "@" will select the terminal being used as the list device. Alternatively, any required file or device can be selected by entering its file descriptor. Note that if a file is specified, it must currently exist; the listing information will be appended to any existing data in the file.

If the !GO response was not made to the above prompt, a message of the following form is output:

Rename devn: as voln

and this is followed by the prompt

OK to Run (Yes or No) ?

If the response is NO, then the complete dialogue is restarted. If the response is YES (or !GO), then execution commences. Note that no default response is allowed to this prompt.

CHAPTER 4
FASTCHEK OPERATION

4.1 DESCRIPTION OF OPERATION

This section describes the operation of FASTCHEK. For the sake of simplicity each function (Initialize, Check, and Rename) is treated separately. Note that the Pack Administration file is discussed in detail in section 4.4. Information about using FASTCHEK with mirror disks is discussed in Section 4.5.

4.1.1 Initialize Function

Certain actions are common to all (Fill, Readcheck, and Noreadcheck) modes of Initialization.

The first operation performed during an Initialization (irrespective of the mode) is a validity check of the Volume Descriptor. This check involves reading the Volume Descriptor; and then rewriting it with a volume name of "NULL", the Bit Map and Directory pointers set to zero, and the Volume On-line Attributes bit set. The sector is then re-read and the data read compared with that which was written. (Note that initially setting up the Volume Descriptor in this way ensures that the pack cannot be marked on if the Initialize function terminates abnormally, since a "DUPL-ERR" error will occur because the name of the pack conflicts with that of the Null device.) If an error occurs while writing or re-reading the Volume Descriptor the standard I/O error message (see Chapter 5) is output followed by

```
WHILE ACCESSING VOLUME DESCRIPTOR
```

and then the program terminates with end of task code 10. If the data read back does not match that written the program will terminate with end of task code 31 after printing the message

```
VOLUME DESCRIPTOR DATA VALIDATION ERROR
    IN FULLWORD AT xx EXPECTED yyyyyyyy FOUND zzzzzzzz
```

Any defective sectors are then located (either by a surface check or from the information in the Pack Administration file) and space for the Pack Administration file is either allocated (in the Fill and Readcheck modes) or determined (in Noreadcheck mode).

The Directory is then allocated. Initially an attempt is made to allocate a directory starting on the first track of the requested cylinder. If this is not possible because the required area

contains defective sectors, then the starting address is incremented by one track at a time and further attempts are made. If the directory cannot be allocated the program terminates with end of task code 20 after printing the message

INSUFFICIENT ERROR-FREE SPACE FOR DIRECTORY

Note that no attempt is made to reposition the Directory at a location before the start cylinder specified by the user. Thus if the Directory cannot be allocated, the user should rerun the program specifying either a smaller start cylinder number or a smaller directory.

Once the Directory has been allocated it is initialized. The Bit Map is then allocated in the first error free area of the required size immediately following the Directory. If this cannot be done the program terminates with end of task code 21 after printing the message

INSUFFICIENT SPACE FOR BIT MAP

Note that if this occurs and a relatively high start cylinder number was specified for the Directory, then the user should rerun the program specifying either a smaller starting cylinder number or a smaller directory.

In general, if the program is unable to allocate either the Bit Map or the Directory, and the user has specified a low start cylinder number for the Directory, a hardware failure is indicated because there will be a large number of defective sectors.

Once the Bit Map has been allocated it is initialized to reflect the allocation of the Directory, the Volume Descriptor, the Pack Administration file, the Bit Map itself, and any defective sectors. The Bit Map initialization is done by first clearing the complete Bit Map and then setting the required bits. Note that a check is made to ensure that the Bit Map can be read and actually contains all zeros. If a defective sector is detected then a mode switch to Readcheck will occur. However, if the data is successfully read but is not all zeros the program terminates with end of task code 30 after printing the message

DATA VALIDATION ERROR IN BIT MAP AT SECTOR xxxxxx
IN FULLWORD xx EXPECTING 00000000 FOUND zzzzzzzz

If this occurs a hardware failure is indicated.

A similar check is made while initializing the Directory. The actions are as for the Bit Map check except that the message

DATA VALIDATION ERROR IN DIRECTORY AT SECTOR xxxxxx
IN FULLWORD yy EXPECTING 00000000 FOUND zzzzzzzz

is printed.

types of disk and various buffer sizes are given in the following table. (Figures are given for Bit Map plus track and a half sized buffer because this size is the optimum for Check/Noreadcheck operations.)

Disk Type	Buffer Size (KB)	Equivalent To	Surface Check Time (sec)
256 MB	304	cylinder	274
256 MB	147	Bit Map + 1.5 trks	302
256 MB	32	2 tracks	398
256 MB	16	track	521
67 MB	80	cylinder	82
67 MB	57	Bit Map + 1.5 trks	96
67 MB	16	track	137
25 MB	32	cylinder	60
25 MB	34	Bit Map + 1.5 trks	60
25 MB	16	track	73
5 MB	independent	of buffer size	92
FLOPPY	4	cylinder	26

In the above table, the maximum buffer size given for each disk type is the optimum for that disk. That is, further increasing the buffer size will not decrease the required time. Note also that the time is independent of buffer size for 2.5 and 5 MB disks because of the different algorithm used which requires only a 1-sector buffer. It should be apparent both from the table and from the formula, that significant reductions can be made in the buffer size without greatly affecting the performance. In particular, if a track sized buffer is used instead of (the optimum) cylinder buffer, then the required time will always be less than double the minimum possible. Thus, for a 256 MB disk, the buffer size can be reduced from 304 to 16 KB (i.e., by 94%) with less than a doubling of the required time.

The time required for a Fill operation is equal to that required for a surface check. That is, an Initialize/Fill will require twice the time given in the table.

The time required for a Check operation depends critically on the specified mode (or the mode used if a mode switch occurs). A full Check will require a Directory check, a File check, and a surface check. The Directory check is essentially very fast, and a Check/Closeonly or a Check/Close in which no mode switch occurs will process in excess of 2000 files per second in the preallocated portion of the Directory, and approximately 150 files per second in the non-preallocated portion. These figures assume that sufficient memory is available to hold one complete track of the preallocated portion of the Directory. (Note that these figures apply to the 256, 67, and 25 MB disks; for the 5 MB disks the figures are approximately 1000 and 75, respectively; and 100 and 5, respectively, for a Floppy.)

The File check requires more time than a Directory check and is optimized by specifying sufficient memory to hold the entire Bit Map in core together with a track sized buffer for the Directory and a half track buffer for the Index file check tree. Assuming that the average Contiguous file is 75 KB in size, and the average Indexed file contains 1.5 index blocks (equivalent to a source file of about 1500 records) then on 256 and 67 MB disks, the File checking logic will process approximately 100 Contiguous files per second and 20 Indexed files per second within the pre-allocated part of the Directory. In the non pre-allocated portion, these figures drop to approximately 80 and 15 respectively. These figures assume that sufficient memory is available to hold the entire Bit Map in memory. If only one quarter of the Bit Map can be held in memory at one time, there will be approximately a 10% degradation in the above figures. Timings for the other types of disks can be estimated by reducing these figures by the performance ratios evident from the Directory check estimates.

4.3 TUNING INFORMATION

If only a single disk has to be processed, then optimum performance is achieved by using the maximum available amount of memory (up to the limit useable by the program). The following table gives the segment size increments required for optimum performance for each function/mode for selected types of disks. The figures given are calculated by using a cylinder sized buffer for surface check operations, a Bit Map plus a track and a half sized buffer for file check operations, and a track buffer for Directory check operations.

DISC TYPE	INIT READ CHECK	INIT NORD CHECK	INIT FILL	CHECK READ CHECK	CHECK NORD CHECK	CHECK CLOSE	CHECK CLOSE ONLY	RE-NAME
256 MB	304	139	304	304	147	16	16	0
67 MB	80	49	80	80	57	16	16	0
LARK 25 MB	32	24	32	32	40	16	16	0
5 MB	12	3	12	12	12	6	6	0
FLOPPY	4	4	4	4	5	4	4	0

Note that FASTCHEK is supplied with a default segment size increment of 16 KB.

It should be noted that (as discussed in section 4.2) the performance penalty of using less than the optimum segment size increment is not severe. Thus if multiple disks have to be processed, then it is advantageous to run multiple copies of FASTCHEK in parallel especially if the disks are on independent channels. In particular, if there are a large number of disks to be checked after a system failure, multiple copies of FASTCHEK should be used to simultaneously check disks on independent channels, and then the second (and any subsequent) disks on each channel should be checked in further parallel runs. Note that since FASTCHEK is a segmented task, multiple copies will share one copy of the code (i.e. pure segment). The fixed and removable packs in a 10 MB disk system should not be processed in parallel since these share a common head arm.

It is important to realise that when a pack is Initialized, specifying a pre-allocated Directory of sufficient size to contain all files to be allocated is critical in achieving high performance during Check functions. It should also be noted that because the Directory check phase is extremely fast, there is almost no penalty in specifying Close mode for a Check function, since if there are no Indexed files open for write the operation will complete almost immediately, but very little time is lost if a mode switch to NOREADCHECK is required.

4.4 PACK ADMINISTRATION FILE

The Pack Administration file, PACKINFO.DIR/0, contains both a list of the defective sectors on the pack and a record of the administrative history of the pack. Its primary function is to provide the defective sector information so that when a pack is Initialized or Checked and the Bit Map has to be rebuilt, this can be done without performing a surface check to find the defective sectors.

The file is a Contiguous file of some 9 sectors for hard disks (and only one sector on Floppy disks since no administration history is maintained on these disks). The file is protected against deletion and update by normal application tasks by maintaining the Directory entry with a Write Count of -1. The file is organised as a set of 64 byte records packed 4 to a sector. The first record is a control record containing global information and pointers to the data records which are either history records or defective sector records.

The history records record the following events:

- pack initialization and mode of initialization
- pack name set by rename or initializing (last four times)
- surface check performed (last four times)
- Check/Close or Check/Closeonly performed
- File integrity check performed

Note that additional types of history records are supported by other utilities.

The Pack Administration file is created when FASTCHEK is used to Initialize a pack in the FILL or READCHECK mode. It will also be created if the NOREADCHECK mode is specified since a mode switch to READCHECK will occur if the file does not exist. Hence the administration history records only this and subsequent events. Thus packs should be Initialized in NOREADCHECK mode so as to preserve any prior history.

Since the file is created at Initialization time, it is always the first file in the Directory and always occupies the first error free area of the required size (which will, in general, directly follow the Volume Descriptor).

All Check and Rename operations performed by FASTCHEK will be recorded in the Pack Administration file provided that it exists. The existence and validity checking are performed as follows: First, an existence check is made by checking whether PACKINFO.DIR exists as the first entry in the Directory and is a Contiguous file. If not, the program assumes that no Pack Administration file exists on the pack, and the message

```
WARNING: PACK ADMINISTRATION FILE PACKINFO.DIR NOT FOUND
```

is printed.

The validity of the file is then checked by first checking that the data pointers in the control record are in nondecreasing order and that the last data pointer is equal to (filesize*4-1). The validity of the contents of the Defective Sector record(s) are then checked as follows: the addresses of the defective sectors are checked to be in ascending order and to be greater than zero and less than or equal to the maximum sector address for the given type of pack. The first address found to be zero is assumed to flag the end of the list, and subsequent addresses are checked to be zero. If the Defective Sector record(s) are full, then the number of defective sectors as held in the Control record and the latest Surface Check History record are checked to be equal and greater than or equal to the number in the Defective Sector record(s). If the Defective Sector record(s) are not full, these three values must be equal. If any of these checks fail, a warning message is output as follows:

```
WARNING: PACK ADMINISTRATION FILE PACKINFO.DIR CORRUPTED
```

The message

```
WARNING: PACK ADMINISTRATION FILE PACKINFO.DIR OVERFLOWED
```

will be output if the Defective sector records are full and the number of addresses recorded is less than the count of defective sectors held in the Control record.

If an Unrecoverable I/O error (status X'84') occurs while the file is being accessed, the message

```
WARNING: PACK ADMINISTRATION FILE PACKINFO.DIR UNUSABLE
```

is output.

If the file exists and is valid, the current system date/time is checked to be later than the 'last updated date/time' held in the Control record. If this check fails, the utility pauses after issuing the message

```
PACK ADMINISTRATION FILE LAST UPDATED ON mm/dd/yy hh:mm:ss
ADJUST SYSTEM DATE/TIME IF REQUIRED, THEN CONTINUE
```

On being continued, the program will use the current system date/time (thus allowing the operator the correct the date/time if it is incorrectly set).

If the user desires to examine the Pack Administration file, it can be dumped using the DISPLAY command of OS/32 COPY. The dump can then be interpreted using the record layouts given in Appendix D.

4.5 USING FASTCHEK WITH MIRROR DISKS

Three of FASTCHEK'S functions, initializing a formatted disk drive pack, renaming a pack and checking the integrity of a pack, are particularly relevant to mirror disks.

The initialization function, which sets up all the control information on a disk pack, is used by the MARK ON processing to determine compatibility for mirroring. This is described in detail in the OS/32 Operator Reference Manual. Note that all disks to be used in the mirrored environment must have been initialized by FASTCHEK to set up control information including defective sector details.

The renaming function is also necessary for mirror disk purposes since disk packs to be used for mirroring must have the same name.

Finally, the integrity checking function has a feature that rebuilds a disk pack's sector allocation map. For example, when the MARK ON processing is establishing the compatibility of two disks for mirroring, the bit map is updated.

Additionally, in case of failure on one of the disks in a mirrored pair, FASTCHEK must be used to reinitialize the faulting disk prior to any attempt to restore the mirrored environment. Refer to the OS/32 System Support Utilities Reference Manual for a complete description of mirror disk synchronization.

BIT MAP RELOCATED TO xxxxxx THROUGH yyyyyy

where xxxxxx is the start address of the Bit Map

yyyyyy is the end address of the Bit Map.

Meaning:

The Bit Map has been relocated to the indicated sector addresses. This message only occurs if the original Bit Map was found to contain a defective sector and had to be relocated.

BIT MAP CONTAINS DEFECTIVE SECTOR AT xxxxxx
IN SECTOR nnnn OF BIT MAP

where xxxxxx is the sector address

nnnn is the number of the sector (base 0) within the Bit Map.

Meaning:

The surface check performed during a Check/Readcheck operation has located a defective sector within the area of the Bit Map. See also section 4.1.2.2.

BIT MAP CONTAINS RECOVERED DEFECTIVE SECTOR AT xxxxxx
IN SECTOR nnnn OF BIT MAP

where xxxxxx is the sector address

nnnn is the number of the sector (base 0) within the Bit Map.

Meaning:

The surface check performed during a Check/Readcheck operation has located a defective sector which was recovered within the area of the Bit Map. This is only an informatory message since Bit Map is about to be completely rebuilt.

```
                {CLOSE      }
                {           }
CHECK devn:  MODE={CLOSEONLY  }
                {           }
                {NOREADCHECK}
                {           }
                {READCHECK  }
```

[EXTENDALLOWED] [WRITERECOVERY] [KEEPSPOOL] [REPORTONLY]

where devn: is the device mnemonic of the disc device

Meaning:

This message is output when all commands have been validated to indicate the function about to be performed.

CHECK COMPLETE - VOLUME voln READY TO BE MARKED ON

where voln is the name of the pack being checked.

Meaning:

This message is output when a Check/Close, Check/Noreadcheck, or Check/Readcheck operation (without the REPORTONLY option set) terminates sucessfully. The pack is then ready for normal use.

filename.ext/act CONTAINS DEFECTIVE SECTOR AT xxxxxx

where xxxxxx is the sector address.

Meaning:

The file contains a sector found to be defective.

filename.ext/act CONTAINS RECOVERED DEFECTIVE SECTOR AT xxxxxx

where xxxxxx is the sector address.

Meaning:

The file contains a sector found to be defective but which was recovered.

CURRENT BIT MAP DIFFERS FROM EXPECTED
nnn BITS STARTING AT xxxxxx <message>

where xxxxxx gives the equivalent sector address at which a string of nnn bits all differ in the same sense.

<message> is either "SET - EXPECTED RESET" or "RESET - EXPECTED SET"

Meaning:

The new Bit Map built during a Check operation with the REPORTONLY option set is not the same as that currently on the pack. See also section 4.1.2.5.

CURRENT/PREVIOUS DEFECTIVE SECTOR DISCREPANCIES
xxxxxx (CHS=ccc/hh/ss) NOW {GOOD }
{DEFECTIVE}

where xxxxxx is the sector address

ccc is the hexadecimal cylinder number

hh is the hexadecimal head number

ss is the hexadecimal sector number.

Meaning:

The surface check performed during a Check/Readcheck operation has located different defective sectors to those recorded in the Pack Administration file. The second message is repeated for each discrepancy. If there are no discrepancies then this second line is replaced by "**** NONE ****".

IN INDEX BLOCK nnnnnn AT xxxxxx OFFSET yy

where nnnnnn is the index block number (base 1)

xxxxxx is the sector address of the index block

yy is the hexadecimal offset within the block

Meaning:

If the REPORTONLY option is set, then, when an error is detected in an index block, this message will follow the error message so as to give the location of the invalid index block.

```
                                  ⎧FILL WITH xxxxxxxx                    ⎫
INITIALIZE devn:  MODE=           ⎨READCHECK [WITH WRITERECOVERY]        ⎬
                                  ⎩NOREADCHECK [WITH WRITERECOVERY]      ⎭
   VOLUME voln    DIRECTORY FOR nnnn FILES AT CYLINDER mmm
                                                          REQUESTED
```

where devn: is the device mnemonic of the disk device xxxxxxxx is the data pattern for the Fill operation

voln is the new volume name for the pack

nnnn gives the capacity of the Directory to be allocated

mmm is the start cylinder number of the Directory

Meaning:

This message is output when all commands have been validated to indicate the function about to be performed.

IN SECTOR nnnn OF FILE

where nnnn is the sector number within the file.

Meaning:

If the REPORTONLY option is set and an allocation conflict or defective sector is found in a Contiguous file, this message will follow the error message to give the location within the file.

INSUFFICIENT ERROR-FREE SPACE FOR DIRECTORY

Meaning:

Insufficient error free Contiguous space is available to allocate the Directory. See also section 4.1.1.

Program Action:

The program will terminate with end of task code 20.

INSUFFICIENT SPACE FOR BIT MAP

Meaning:

Insufficient free Contiguous space is available to allocate the Bit Map. See also sections 4.1.1 and 4.1.2.2.

Program Action:

The program will terminate with end of task code 21.

APPENDIX A
FASTCHEK COMMAND SUMMARY

BLOCKS [=] [bbb] [/ [ccc]]

CHECK [=] devn:

CLOSE

CLOSEONLY

COMMAND [=] fd

DIRECTORY [=] [fff] [/ [ccc]]

END

EXTENDALLOWED

FILL [=] [xxxxxxxx]

INITIALISE [=] devn:

INITIALIZE [=] devn:

KEEPSPOOL

LIST [=] fd

NOREADCHECK

NOWRITERECOVERY

READCHECK

RENAME [=] devn:

REPORTONLY

VOLUME [=] voln

WRITERECOVERY

APPENDIX C
DEVICE CHARACTERISTICS

The following tables give pertinent characteristics for the various types of disks. Note that the table on page C-3 contains disks, which although usable under the current Operating System release, are no longer current products.

DISK TYPE	5 MB FIXED	5 MB REMOV	MSM80 REMOV	MSM300 REMOV	FLOPPY	MSM80F + HPT	MSM FIX
Nominal Capacity	5 MB	5 MB	67 MB	256 MB	250 KB	68.5MB	268
Device Code (Hex)	32	33	35	36	37	38	
(Decimal)	50	51	53	54	55	56	
Rotation Time (sec)	1/40	1/40	1/60	1/60	1/60	1/60	1/
Number of Cylinders	408	408	823	823	77	842.2	10
Number of Heads (i.e. tracks|cyl)	2	2	5	19	1	5	
Cylinder size (KB)	12	12	80	304	3.25	80	2
(sectors)	48	48	320	1216	13	320	10
Track size (KB)	6	6	16	16	3.25	16	
(sectors)	24	24	64	64	13	64	
Bit Map size (KB)	2.50	2.50	32.35	122.25	0.25	33.00	122.
(sectors)	10	10	129	489	1	132	4
Total size (KB)	4896	4896	65840	250192	250.25	67376	2621
(sectors)	19584	19584	263360	1000768	1001	269504	1048
Default Directory size (files)	120	120	640	1600	5	640	16
(blocks)	24	24	128	320	1	128	3
Directory Alloc'n Interleaving Factor	6	6	32	32	1	32	

DISC TYPE	HPT of MSM80F	MSM80 FIXED	MCCD32 REMOV	MCCD32 FIXED	MCCD64 FIXED	MCCD96 FIXED
Nominal Capacity	1.5 MB	67 MB	13.5MB	13.5MB	40 MB	67 MB
Device Code (Hex)	39	3A	3B	3C	3D	3E
(Decimal)	57	58	59	60	61	62
Rotation Time (sec)	1/60	1/60	1/60	1/60	1/60	1/60
Number of Cylinders	19.2	820	823	821	821	821
Number of Heads (i.e. tracks|cyl)	5	5	1	1	3	5
Cylinder size (KB)	80	80	16	16	48	80
(sectors)	320	320	64	64	192	320
Track size (KB)	16	16	16	16	16	16
(sectors)	64	64	64	64	64	64
Bit Map size (KB)	0.75	32.25	6.5	6.5	19.25	32.25
(sectors)	3	129	26	26	77	129
Total size (KB)	1536	65600	13168	13136	39408	65680
(sectors)	6144	262400	52672	52544	157632	262720
Default Directory size (files)	20	640	320	320	320	640
(blocks)	4	128	64	64	64	128
Directory Alloc'n Interleaving Factor	32	32	32	32	32	32

DISK TYPE	2.5 MB FIXED	2.5 MB REMOV	40 MB REMOV	CDD50 FIXED	CDD50 REMOV
Nominal Capacity	2.5 MB	2.5 MB	40 MB	25MB	25MB
Device Code (Hex)	30	31	34	2A	2B
(Decimal)	48	49	52	42	43
Rotation Time (sec)	1/25	1/25	1/40	1/60	1/60
Number of Cylinders	203	203	406	624	624
Number of Heads (i.e. tracks|cyl)	2	2	20	2	2
Cylinder size (KB)	12	12	100	31	31
(sectors)	48	48	400	124	124
Track size (KB)	6	6	5	15.5	15.5
(sectors)	24	24	20	62	62
Bit Map size (KB)	1.25	1.25	20.00	10.00	10.00
(sectors)	5	5	80	37	37
Total size (KB)	2436	2436	40600	19344	19344
(sectors)	9744	9744	162400	77376	77376
Default Directory size (files)	60	60	400	640	640
(blocks)	12	12	80	128	128
Directory Alloc'n Interleaving Factor	4	4	5	32	32

APPENDIX G
COMPARISON WITH OS/32 DISCHECK AND OS/32 DISCINIT

G.1 INTRODUCTION

OS/32 FASTCHEK is a functional replacement for both OS/32 DISCHECK and OS/32 DISCINIT. More specifically, the Check function of FASTCHEK replaces DISCHECK and the Initialize and Rename functions replace DISCINIT.

G.2 CHECK FUNCTION

The following table gives the equivalent FASTCHEK command for each of the DISCHECK START parameters. Note that because of FASTCHEK's automatic mode switching feature, the NOREADCHECK modes are not strictly equivalent.

DISCHECK Parameter	FASTCHEK Command
dev:	CHECK [=] devn:
list fd	LIST [=] fd
READCHECK	READCHECK
NOREADCHECK	NOREADCHECK
CLOSE	CLOSEONLY

Thus, the following Start commands are equivalent.

```
DISCHECK:    ST ,DSC1:,PR:,READCHECK
      or     ST ,DSC1:,PR:
FASTCHEK:    ST ,CHECK=DSC1:,LIST=PR:,READCHECK

DISCHECK:    ST ,DSC1:,PR:,NOREADCHECK
FASTCHEK:    ST ,CHECK=DSC1:,LIST=PR:,NOREADCHECK
      or     ST ,CHECK=DSC1:,LIST=PR:,CLOSE
      or     ST ,CHECK=DSC1:,LIST=PR:

DISCHECK:    ST ,DSC1:,PR:,CLOSE
FASTCHEK:    ST ,CHECK=DSC1:,LIST=PR:,CLOSEONLY
```

G.3 INITIALIZE FUNCTION

The following table gives the equivalent FASTCHEK command for each of the DISCINIT Start parameters when used to Initialize a pack.

DISCINIT Parameter	FASTCHEK Command
DISC=dev:	INITIALIZE [=] devn:
CLEAR	implied by INITIALIZE
VOLUME=voln	VOLUME [=] voln
BLOCKS=n [/m]	BLOCKS [=] [n] [[/]m]
FILL=byte	FILL [=] [xxxxxxxx]

Thus, the following Start commands are equivalent.

```
DISCINIT:   ST ,DISC=DSC1:,CLEAR,VOLUME=SYS,BLOCKS=100/1,
                                                    FILL=BD
FASTCHEK:   ST ,INITIALIZE=DSC1:,LIST=CON:,VOLUME=SYS,
                                BLOCKS=100/1,FILL=BDBDBDBD

DISCINIT:   ST ,DISC=DSC1:,CLEAR,VOLUME=SYS
FASTCHEK:   ST ,INITIALIZE=DSC1:,LIST=CON:,VOLUME=SYS,
                                                BLOCKS=0
```

G.4 RENAME FUNCTION

The following table gives the equivalent FASTCHEK command for each of the DISCINIT Start parameters when used to Rename a pack.

DISCINIT Parameter	FASTCHEK Command
DISC=dev:	RENAME [=] devn:
VOLUME=voln	VOLUME [=] voln

Thus, the following Start commands are equivalent.

```
DISCINIT:   ST ,DISC=DSC1:,VOLUME=SYS
FASTCHEK:   ST ,RENAME=DSC1:,LIST=CON:,VOLUME=SYS
```

INDEX

MANUAL UPDATE PACKAGE COVER SHEET

MANUAL TITLE: OS/32 FASTCHEK Reference Manual

PUBLICATION NUMBER: 48-064

OLD REVISION LEVEL: F01 R00

NEW REVISION LEVEL: F02 R00

This package of affected pages updates the current version of the subject manual. New features, as well as changes, deletions and additions to information in this manual are indicated by change bars in the page margins. Please discard the indicated old pages and replace or insert them with the supplied new pages.

OLD PAGES	NEW PAGES
Title Sheet/Disclaimer, F01 R00	Title Sheet/Disclaimer, F02 R00
Sheets i through iii, F01 R00	Sheets i through iii, F02 R00
Sheet iii, F01 R00*	Sheet v, F02 R00
Sheet 2-3, F00 R00	Sheet 2-3, F00 R00
Sheet 2-4, F01 R00	Sheet 2-4, F02 R00
Sheet 2-7, F00 R00	Sheet 2-7, F02 R00
Sheet 2-8, F01 R00	Sheet 2-8, F01 R00
Sheet 3-5, F00 R00	Sheet 3-5, F02 R00
Sheet 3-6, F00 R00	Sheet 3-6, F00 R00
Sheet 3-7, F00 R00	Sheet 3-7, F02 R00
Sheet 3-8, F00 R00	Sheet 3-8, F02 R00
Sheet 3-11, F00 R00	Sheet 3-11, F02 R00
Sheet 3-12, F00 R00	Sheet 3-12, F00 R00
Sheet 3-19, F00 R00	Sheet 3-19, F00 R00
Sheet 3-20, F00 R00	Sheet 3-20, F02 R00
Sheet 3-21, F01 R00	Sheet 3-21, F02 R00
Sheet 3-22, F00 R00	Sheet 3-22, F00 R00
Sheet 3-23, F01 R00	Sheet 3-23, F02 R00
Sheet 3-24, F00 R00	Sheet 3-24, F02 R00
Sheet 3-27, F00 R00	Sheet 3-27, F02 R00
Sheet 3-28, F00 R00	Sheet 3-28, F00 R00
Sheet 3-29, F01 R00	Sheet 3-29, F02 R00
Sheet 3-30, F00 R00	Sheet 3-30, F02 R00

* The preface in 48-064 F01 R00 was misnumbered so there were two page iii's. This has been corrected in 48-064 F02 R00.

OLD PAGES	NEW PAGES
Sheet 3-31, F00 R00	Sheet 3-31, F02 R00
Sheet 3-32, F00 R00	Sheet 3-32, F02 R00
Sheet 3-33, F00 R00	Sheet 3-33, F02 R00
Sheet 4-1, F00 R00	Sheet 4-1, F02 R00
Sheet 4-2, F00 R00	Sheet 4-2, F00 R00
Sheet 4-17, F01 R00	Sheet 4-17, F02 R00
Sheet 4-18, F00 R00	Sheet 4-18, F02 R00
	Sheet 4-18a, F02 R00
Sheet 4-19, F00 R00	Sheet 4-19, F02 R00
Sheet 4-20, F01 R00	Sheet 4-20, F01 R00
Sheet 4-21, F00 R00	Sheet 4-21, F02 R00
Sheet 5-9, F00 R00	Sheet 5-9, F00 R00
Sheet 5-10, F00 R00	Sheet 5-10, F02 R00
Sheet 5-11, F00 R00	Sheet 5-11, F00 R00
Sheet 5-12, F00 R00	Sheet 5-12, F02 R00
Sheet 5-25, F00 R00	Sheet 5-25, F02 R00
Sheet 5-26, F01 R00	Sheet 5-26, F01 R00
Sheet A-1, F00 R00	Sheet A-1, F02 R00
Sheet C-1, F01 R00	Sheet C-1, F02 R00
Sheet C-2, F01 R00	Sheet C-2, F01 R00
Sheet C-3, F01 R00	Sheet C-3, F02 R00
Sheet G-1, F01 R00	Sheet G-1, F02 R00
Sheet G-2, F01 R00	Sheet G-2, F02 R00
Ind-1, F01 R00	Ind-1, F02 R00
Ind-2, F01 R00	Ind-2, F02 R00