VTOS

4.0

[This manual was typed in by Ira Goldklang off of an nth generation photocopy which was heavily degraded. The first 2 pages were missing as well as any pages which may come after Appendix A. It is intended that this document will be updated when and if should those pages be discovered. The REV date in the footer will be updated accordingly]

Now that you are ready to do something, the VTOS 4.0 loading process will proceed as follows once the <RESET> button has been depressed:

1) If the <BREAK> key is held down while the <RESET> button is depressed, ROM BASIC will be entered.

2) At this point, VTOS 4.0 should load and announce itself. A message giving the current date (if entered since power-up) or the absence thereof will be issued.

3) If the <UP-ARROW> key is held down during the boot process, the keyboard de-bounce routine will be disabled (useful for tape programs).

4) If the <CLEAR> key is held down during the boot process, your special system configuration file will NOT be loaded. (See “SYSTEM” command).

5) If a non-breakable AUTO command line exists, it will be executed now.

6) If the <ENTER> key is held down, the system will now go immediately into the command line interpreter and display ‘VTOS READY’.

7) If the <D> key is held down, the system will go directly into the disk DEBUG routine. Note that this will NOT be the extended debugger.

8) If a breakable AUTO command line exists, it will now be executed.

9) Since nothing else to be done can be found, the command line interpreter will now display ‘VTOS READY’.

Please note that this entire boostrap loading process will take place below X’5200’ and none of the BASIC information kept in low memory will be altered.

Regarding the keyboard, you now have several keyboard features readily at hand to serve you. They are:

1) Shift case reversal for alphabetic keys. Your keyboard will normally enter upper case characters unless the <SHIFT> key is held down, in which case, lower case characters will be entered. To reverse this function, type a shifted zero <SHIFT><0>.

2) Control characters may be entered by using a shifted down-arrow as the control key, and holding it down while entering the desired character.

3) The entire keyboard has an auto-repeat feature built in. If you hold any key down for one second, it will then start repeating at a rate of 10 times per second.

4) Any key may have a special ‘FUNCTION’ associated with it. This works just like a shift except that the <CLEAR> key is held down instead of the <SHIFT> key while the particular key is hit. The function performed is determined by the program which is running at the time, however, if the program is not specifically written to handle the function, the characters entered will all be either graphics or space compression codes.

5) In order to perform a clear screen operation, a <SHIFT><CLEAR> will be required due to item number 4, above.

6) A <SHIFT><BREAK> will re-select the last disk drive which was used and, if it was a mini-floppy, turn its drive motor back on. This will prove to be very helpful in cases where a selected drive has a door open, etc. and would otherwise require a system re-boot.

GENERAL INFORMATION

VTOS 4.0 is TOTALLY upward compatible with TRSDOS!

Note however, that certain advanced features of VTOS (such as loading a new device driver, etc.), which are not in TRSDOS, will place relocatable code in high memory. When this is done, programs which do not honor HIGH$ (such as those coming out of DENVER), will not work.

A pointer called HIGH$ will be maintained in a manner such that it always points to the highest available UNUSED memory location in the computer. Most programs written will honor this pointer (which, by the way, has been around since TRSDOS 2.0). A few programs, such as Michael Shrayer's Electric Pencil, do not honor HIGH$ and therefore require a patch in order to properly operate when high memory drivers are present. (A patch for PENCIL is on your disk --- See Appendix).

Although VTOS uses a file structure which is upward compatible with TRSDOS, files written under NEWDOS are not guaranteed to be compatible with VTOS since several standards developed for TRSDOS were not maintained within NEWDOS.

VTOS is designed to be device independent. That is, if you have a program which is written to operate using certain input/output devises (e.g., a serial printer, parallel printer, or the display), and you find that your needs have changed, or that your printer has quit, or that you don't want to waste paper on test runs; then the VTOS device independent I/O structure will allow you to convert all of your program's input/output requests to use the new device by simply entering just one system command.

Example? try this one: your BASIC program normally produces printed output via LPRINT commands, with the command:

ROUTE *PR to *DO

Your output will be redirected to the display!

You may also 'fake out' your programs by creating logical devices to be used in lieu of non-existent hardware (physical devices) which they normally expect to use. In these cases, the logical device will actually perform its I/O to a disk file.

example: **ROUTE *PR TO PRINTFIL**

Your printer output will now be placed on the disk to be printed at a later time on a system with a printer.

Please note that logical devices need to be closed when you are through using them. This should be accomplished through the use of the RESET command.

example: **RESET *PR**

There are normally six logical devices present in the VTOS system. These are:

*KI	keyboard input

*DO	display output
*PR	printer output
*SI	source input (normally = *KI)
*LO	listing output (normally = *DO)
*JL	job log (normally set to NIL)

The logical JOBLOG device (*JL), when enabled, will contain a ‘TIME-STAMP’ of all commands, errors, etc., encountered during the operation of VTOS 4.0. This is very useful for long unattended operations where you need to monitor the status of several programs or where you need to obtain the execution time of certain programs or those expensive time-sharing calls.

example: **ROUTE *JL TO MICRONET/LOG**

This will provide you with a file containing all commands, etc. along with the time they were encountered. Be sure to RESET the file in order to properly close it! Some of the more advanced software on the market, such as ST80III, take advantage of this feature and allow you to make direct JOBLOG entries via the Control-J feature. Note that you may also make entries via VTOS comment commands (any line beginning with a period) or through BASIC by opening the *JL device and writing to it just as though it were a disk file.

The JOBLOG is especially handy if you are interested in obtaining the execution time of particular program or set of programs, etc.

OPERATIONAL FEATURES

The VIRTUAL TECHNOLOGY OPERATING SYSTEM Version 4.0 contains numerous internal operational features and characteristics which improve reliability, simplify usability, and eliminate many of the user frustrations found in TRSDOS, NEWDOS, and other operating systems for the TRS-80.

The following is a partial list of several of these features which you may be interested in knowing about:

1) Large (8”) drive support.

2) Double Sided drive support.

3) Double Density drive support.

4) 80 Track drive support.

5) Winchester technology fixed drive support.

6) Supports any combination of the above drives up to a max of 8 drives.

7) Supports double-speed processor clock modifications.

8) FASTER! Improved overlay structure using ISAM accessing techniques improves loading times by up to 1400%.

9) General purpose output spoolers of a true, symbient design provide simultaneous output and program execution without any user intervention.

10) Keyboard Type-Ahead feature permits you to enter keystrokes before your programs need them.

11) User definable keys. (See KSM Driver)

12) Built-in graphic string picker allows you to directly enter graphic symbols into a BASIC program from the keyboard through the use of the <CLEAR> key.

13) Dated files. All files are accompanied by the date of their last modification.

14) All files which have been updated are accompanied by a ‘+’ until they are backed up. This permits the BACKUP utility to copy only those files which have actually been updated since a previous BACKUP.

15) File transfer by class. Allows transferring of all files of a similar directory clasification such as ‘/SYS’, INV, etc.

16) Built-In SYSTEM command contains lower case display driver, screen print, break key disable, blinking cursor driver, disk drive stepping rate and motor-on delay adjustments, and much more.

17) User may SYSGFN a custom VTOS system configuration containing special I/O drivers, device LINKing and ROUTEing, SPOOLing and DEBUG tasks, etc. which will be automatically loaded during the BOOT process.

18) Non-BREAKable AUTO and CHAIN commands.

19) Wild-card DIRectory. Permits you to locate all files of a certain classification such as '/BAS'. Uniformly indicates file size in K (1024 bytes) regardless of drive type.

20) Dynamic file name defaults in APPEND, COPY, and RENAME commands allow you to specify only minimal information about file names.

21) COPY and APPEND commands execute up to 300% faster.

22) ALLOCate command for pre-allocation and nonreleasibility of file space. File space will never shrink.

23) MEMORY command for directly setting upper memory limit.

24) Variable Length file support which blocks short user data records across sector boundaries thereby taking maximum advantage of disk file space.

COMMAND SUMMARY

The format of all VTOS command lines is a command name, optionally followed by one or more file specifications (filespec), device specifications (devspec), prepositions (prep), or parameter fields (params).

A ‘FILESPEC’ consists of a file name (one to eight alphanumeric characters) optionally followed by a slash ‘/’ and a file extension (one to three alphanumeric characters), optionally followed by a period and a password (one to eight alphanumeric characters), optionally followed by a colon ‘:’ and a logical disk drive number. If no disk drive number is given, a global search will be used where all disks in the system will be searched for the file or for space to create it on a non-write protected disk.

Example: MYFILE/TXT.SECRET:1

A ‘DEVSPEC’ consists of an asterisk ‘*’ followed by exactly two alphanumeric characters.

Example: *DO (that is your display)

A ‘PREP’ is any one of several prepositions which are imbedded into VTOS purely for the ‘readability’ of the command line and are skipped by the system when processing a command line. These preps are: TO, ON, OVER, and USING. It should be noted that a space or comma may be used in place of these prep’s.

The ‘PARAMS’ are items which are included within parenthesis. The actual definition and usage of the parameter fields depends upon the command.

If a ‘FLAG’ is noted as a parameter, then any one of the values: ON, OFF, YES, NO, Y, or N may be supplied. The UN flag setting is assumed if no value is given.

For the command formats given here, any parameters enclosed in square brackets ‘[‘ and ‘]’ are optional fields for the command.

ALLOC <filespec> [(SIZE=nnnn)]

The ALLOC command will pre-allocate space to a file in the most contiguous manner possible (smallest segment count). Although the stated file SIZE is the minimum the file will ever have (regaledless of the EOD position), the file will automatically increase in size if data is written to it at or past the EOF. The SIZE param is optional --- if present, the stated amount of disk space (in K) will be allocated to the file; if not present, the file will simply be created in your directory but will not actually occupy and disk space. if the stated file already exists, the greater of the file’s current size or the SIZE given will be used as the minimum size. The content of a file (if any) will not be altered in any way by this command.

example: **ALLOC PERMFILE/DAT (SIZE=32)**

This will set the file’s minimum size to 32K. See also DIR.

APPEND <filespec/devspec> [prep] <filespec/devspec>

The APPEND command appends two files together. The first file/device specified is appended to the end of the second file/device specified. If the second specification is for a file, dynamic defaults will be used for the name, extension, and password. This will cause any of these fields which are not specified to be duplicated from the first file specification. Note that this does not apply to the drive number due to its global nature.

example: **APPEND NEWDATA TO OLDDATA**

ATTRIB <filespec> ([ACC=a], [UPD=u], [PROT=p], [INV], [VIS])

The ATTRIB command alters a file's protection attributes. The access and update passwords will be changed to a and u respectively, while the level of protection granted to those knowing only the access password will be limited to p. The values p may take on are: EXEC(ute only), READ (only), WRIT(e), RENA(me), NAME, KILL, and FULL (or ALL). The 'INV' and 'VIS' options are used to declare the file is either invisible or visible, respectively.

example: **ATTRIB GONE/BAS (INV)**

Your file is now invisible.

AUTO [*] <command line>

The AUTO command will cause the specified command line to automatically be executed every time the system is loaded due to a power-on or <RESET> unless the <ENTER> key is held down during the load. The presence of an asterisk will cause VTOS 4.0 to disable the <BREAK> key. Note that the system will be configured prior to the execution of the specified command line. (See SYSTEM command).

BOOT

This commend causes the disk in drive zero to be booted into the system. This has the same effect as the reset pushbutton switch on the back of the keyboard unit.

BUILD <filespec>

The BUILD commend will invoke a simple ASCII file builder. The builder accepts keyboard input on a line basis and dumps it to disk. Each input line may be 'edited' by using the back arrow (backspace) and shifted back arrow (cancel) keys prior to hitting the <ENTER> key. The file will be written to disk when the <BREAK> key is hit as the first keystroke of an input line.

This command was primarily designed for creating short text files such as those required by CHAIN, PATCH, and the KSM driver. No default file extension exists, however, if an extension of ‘KSM’ is used, each input line will be preceded with the keyboard character which will invoke the respective line. The input line length will be limited to 64 characters except in the case of a KSM file which may be up to 256 characters in length.

CHAIN [*][=][$] <filespec> [(params)][;]

The CHAIN command allows the user to create a sequence of command or other keystrokes for the system to automatically execute. These inputs are stored on a disk file by the user and then passed to CHAIN via the file’s name (filespec).

example: **CHAIN DEMO**

Note that the default file extension is ‘JCL’ and secondly, if you desire, you may abbreviate ‘CHAIN’ to a ‘!’ (commonly referred to as a “bang”).

All programs which work with the normal keyboard line input handler (KEYIN) will work under CHAINing. It should be noted that the single character input handlers (KBD, KEY, and INKEY$) will still use the actual keyboard for their input. The BREAK key (if not disabled) will be examined during the fetching of each input line and will cause a MANUAL CHAINING ABORT condition if depressed. An end-of-file condition will issue the message END OF FILE ENCOUNTERED to the display (without any actual input) and return control to the previous *KI driver unless a new *KI driver was established during the program chaining operation.

The actual sequence of events which normally take place during program CHAINing consist of two phases. The first, which is called the compile phase, raids the data from the specified file, massages it, and writes it out to the ‘SYSTEM/JCL’ file. The execution phase then executes directly from this file.

Three exceptions to this procedure exist and are implemented by means of a single character following the ‘CHAIN’, they are: 1) ‘*’ means to rerun the last CHAIN, 2) ‘$’ means to perform the compile phase only (for testing your JCL), and 3) ‘=‘ means to skip the compile phase (execute directly). Be careful! If you skip the compile phase, some JCL features will be ignored, keep reading.

The parameter field on the CHAIN command is dependent upon the actual contents of the file. This field consists of symbols and optionally, values for the symbols.

example: **CHAIN DEMO (BASIC,GAME=CHECKERS)**

This would create two symbols, BASIC and GAME, and set the string value of GAME to CHECKERS.

If you run out of room on the command line, you may extend it with a semicolon between pairs of parenthesis on each line.

Two special field types exist within the JCL, a string substitution field (denoted by a symbol placed within pound symbols, e.g. #GAME#), and an expression (consisting of one or more symbols with logical operators placed between them). During the compile phase, any string substitution symbols found are replaced with their current string value. For example -- if GAME=CHECKERS in the parameter field, and if #GAME# appears within the specified file, then the #GAME# will be changed to CHECKERS. Any expression fields found are also evaluated during this phase. A symbol is logically ‘TRUE’ if it has been defined and is otherwise ‘FALSE’. The logical operators are: a period ‘.’ For an ‘AND’, a comma ‘,’ for an ‘OR’, and a minus ‘-’ for a ‘NOT’.

The JCL which is processed during the compile phase is:

character string substitution (and concatenation)

logical expression evaluation (AND, OR, NOT)

//. <compile comment> --
placed in JODLOG

//INCLUDE <filespec> --
to include (or nest) other files

//IF <expr>--
start of conditional block

//ELSE -- alter state of conditional block

//END -- end of conditional block

//SET <symbol> --
to set the symbol TRUE

//RESET <symbol>--
to set the symbol FALSE

//ASSIGN <symbol>=<value>--
to establish a new symbolic value

//QUIT-- end of compile phase,
no execution (error detected)

Note: conditional blocks may be nested.

The JCL which is processed during the execution phase is:

.__<comment> --
comment is displayed and logged.

//EXIT -- to exit the chaining (no message)

//ABORT -- to abort the chaining (no message)

//STOP -- to quit chaining and return to user (no message)

//PAUSE <comment>--

to wait for an <ENTER> or <BREAK>

//ALERT_[(]<noise>,<silence>[)]--
gives an audible alert (w/ display)
<ENTER> or <DREAK> to stop

//FLASH <duration>
-- attracts attention

//DELAY <duration>
-- delays execution for stated time
period. <ENTER> or <BREAK> to stop

//WAIT hh:mm:ss <comment> --
wait for a predetermined time-of-day before continuing

//KEYIN <comment>--
to request and accept execution phase conditionals

//<char> -- to start a new execution phase conditional block

/// -- to end all execution phase conditional blocks

Any ‘PAUSE’ commands found in a JCL file which are not proceeded by slashes (apparently coming from earlier systems), will automatically be translated to ‘//PAUSE’ during the compile phase for compatibility with VTOS 3.0.

Here are a few examples of CHAINing files:

```
.GAME LIBRARY LOADING PROCEDURE
//. USE: ‘CHAIN <filespec> (<gamename>) ‘ AS COMMAND LINE
//.
//. SEE WHICH GAME IS TO BE PLAYED
//IF CHECKERS
BASIC
RUN CHECKERS
//ELSE
//IF STARTREK
BASIC
RUN STARTREK
//ELSE
//IF INVADERS
VADERS
//ELSE
.NO GAME NAME SPECIFIED
//QUIT
```

```
. PROGRAM ASSEMBLY PROCEDURE
//. USE: ‘CHAIN <filespec>
```

```
(<NAME=name><,OBJECT><,LIST>)’
M80
//IF OBJECT.LIST
#NAME#,#NAME#=#NAME#
//ELSE
//IF OBJECT
#NAME#,=#NAME#
//ELSE
//IF LIST
,=#NAME#
//ELSE
=#NAME#
//END
//END
//END
```

• EXAMPLE OF A SASSY ALARM CLOCK

```
//. USE ‘CHAIN <filespec>‘
//.
//PAUSE PLEASE HIT <ENTER> TO CONTINUE
.GOOD
CLOCK
.JUST IN CASE THE CURRENT TIME ISN’T SET….
TIME 08:00:00 (JUST A WILD ADDUMPTION)
.THIS ALARM CLOCK WILL ‘GO OFF’ IN 30 SECONDS
//WAIT 08:00:30 PLEASE BE PATIENT ..
.HIT <BREAK> TO SILENCE ALARM, OTHERWISE……
//ALERT 25
.GOOD, NOW THAT YOU ARE AWAKE RIGHT?
//DELAY 25
.WRONG!
.
.WAKE UP, YOU LAZY FOOL!
//FLASH 25
.STILL NOT UP?
.
. YA CAN’T SEE
. YA CAN’T HEAR
.
. YA MUST BE DEAD!!
.
//ALERT (1,1,1,1,1,5,5,5,5,5,5,5,1,1,1,1,1,10)
. HEY, YOU WEREN’T...
//DELAY 10
. GOOD!
```

. EXAMPLE OF MENU SELECTION PROCEDURE

```
//. USE  ‘CHAIN <filespec>‘
//.
. WHAT DO YOU WANT--
. 1) A DIRECTORY LISTING.
. 2) A LISTING OF THE AVAILABLE DISK SPACE.
. 3) A LISTING OF THE DEVICES IN THE SYSTEM.
//KEYIN YOUR RESPONSE (1-3)
//1
DIR :0
//EXIT
//2
FREE
//EXIT
//3
DEVICE
//EXIT
///
. SO, YOU DON’T LIKE WHAT’S ON THE MENU, HUH?
. WELL, HERE IS A COMPLETE LIST OF MY LIBRARY,
. GOOD LUCK!
LIB
.OK, WHAT WILL IT BE?
//STOP
```

CLOCK [(flag)]

This command will turn the realitime clock display on or off according to the stated flag condition. This command has no effect on the internal clock (it is always running), just the displayable one.

COPY <filespec/devspec> [prep] <filespec/devspec>

The COPY command will copy data from the first file or device to the second file or device. If the second specification is for a file, dynamic defaults will be used for the name, extension, and password. This will cause any of these fields which are not specified to be duplicated from the first file specification. Note that this does not apply to the drive number due to its global nature.

example: **COPY TEST. SECRET/BAS /OLD**

If either specification is for a device, the copy will be performed on a character by character basis. Otherwise, the copy will be performed on a file basis using memory which is not otherwise used.

The operation may be aborted at any time with the <BREAK> key if a character oriented, logical device, copy is being performed.

DATE mm/dd/yy

This command resets the internal realtime date to that which is specified. Note that this date will be placed in memory and automatically reused upon re-booting VTOS (provided it is not clobbered by foreign systems, etc.)

DEBUG ([flag][EXT])

The DEBUG command, unlike all other commands, does not appear to perform an immediate task. Instead, it simply enables an internal GHOST job which waits for the appropriate time to show itself. Those times are when the <BREAK> key is depressed (if <BREAK> is enabled), or after a program has been loaded and the first instruction in the program is about to be executed.

DEBUG may be automatically entered after VTOS 4.0 is loaded by holding down the ‘D’ key on the keyboard during the bootstrap operation. This will enable you to debug any code which exists above the system area (X’5200’) without destroying it by issuing commands to VTOS.

An extended DEBUG facility may be loaded into high memory by setting the ‘EXT’ flag.

It should be noted that DEBUG will be disabled during the execution of any EXECute only programs.

The following commands are available once the use has entered the debugger, those indicated with an asterisk are only available in the extended debugger. Those fields which are delimited with pointy bracket ‘<>‘ are optional. They are:

A --- ALPHANUMERIC display mode

* B <s>,<d>,1
--- BLOCK MOVE 1 bytes of memory from s to d

C --- instruction / CALL STEP; execute the next instruction,
if it is a call, perform entire subroutine.

Daddr --- set memory DISPLAY address to addr

* E<addr> --- ENTER data into memory

* Faaaa,bbbb,cc

--- FILL memory from aaaa to bbbb,
inclusively, with the data byte cc

Gaddr<,bkp1><,bkp2>
--- GOTO address addr
(optionally setting breakpoints at
addresses bkp1 and bkp2) and start
execution.

H --- set HEXADECIMAL display mode

I --- single INSTRUCTION set

* J --- Jump over the next instruction byte
(increment PC by 1)

* L [aaaa],[dd]--- LOCATE the next occurrence

of the data byte dd, optionally starting at address aaaa

M [aaaa] --- set memory MODIFICATION mode,
Optionally starting at address aaaa

* N [aaaa] --- position to the NEXT relative
block/instruction (such as JR X or load block)

* O --- normal return to VTOS

* Paaaa,bbbb --- PRINT memory from aaaa through bbbb,
inclusively, in both hex and ASCII

* Qii --- QUERY input port ii for data

* Qoo, dd --- send data byte dd to output port ii

Rrp dddd --- alter REGISTER pair ‘p’ to become dddd

S --- set full SCREEN display mode

* T[aaaa] --- TYPE ASCII data into memory,
optionally starting at address aaaa

U --- dynamic display UPDATE mode

* V [aaaa],[bbbb],[lth]
--- VERIFY the memory block of 1th bytes
at aaaa to the memory block at bbbb

* W [aaaa],[dddd]
--- search memory for the WORD ADDRESS
dddd, optionally starting at address aaaa

X --- set EXAMINE display mode

+ (or ;) --- increment the memory display address ‘page’

- --- decrement the memory display address ‘page’

* d,[c],[s],o,addr,[1th]
--- transfer a cylinder or sector to or from a disk unit.

Where d is the drive, c is the cylinder, s is the sector, addr is the starting memory address, o is the operation (an R for read, a W for write, a * for directory write), and 1th is the length (in sectors). A full cylinder read/write will occur if the sector number is null. The directory cylinder will be used if no other is given.

NOTICE: Be extremely careful when using this DEBUG command since it encompasses a large variety of different drive types, each of differing formatting structures.

DEVICE

The DEVICE command will cause all logical devices which are currently enabled in the system to be displayed. Along with this, the I/O direction (indicated with arrow head symbols), any ROUTEing which has been performed, as well as the address of the actual I/O driver module currently in use will be displayed.

DIR [name][/ext][:d] ([A][I][P][S])

This is the directory command which is used to obtain an itemized list of the files currently residing in the system. Included in the itemization is the file's protection level, the EOF record number, the total size of the file (in K, where K=1024), the date the file was last modified (created or written), and a '+' indication if new information has been placed on the file since the last time the file was backed up.

The listing will include all files whose name or extension STARTS with that specified and which resides on the specified drive. If any of these fields are not specified, all possible ones will be used as the default.

The :d is used to specify which drive is desired, while the other optional flags are used to indicate initialization is desired (A), and whether invisible files (I), and System Files (S) are to be included. An additional printed directory may be produced via the (P) option.

DUMP <filespec> ([START=s][END=e][TRA=t])

The DUMP command dumps memory from the starting address s through the ending address e to the specified disk file. The file will include the transfer address t, if specified, otherwise, the transfer will be back to the system. The parameters which DUMP accepts are:

START (or S) ---- starting address of block
END (or E) ---- ending address of block
TRA (or T) ---- transfer block's length

Either decimal or hexadecimal addresses (preceded by X') may be used. The memory area must be above X'6000'. The default file extension is 'CIM' (core image).

example: **DUMP TEST (S=X'6000', E=X'64FF', T=X'6000')**

FILTER <devspec> [prep] <filespec> [prep] [(params)]

The FILTER command is used to establish a new logical I/O path within the VTOS system which will include a routine that filters (or massages) the data passed through it. The <filespec> must contain the filter routine. The <params>, if any, are defined by the filter routine itself. The default file extension is ‘FLT’.’

FREE

This command is used to obtain a summary of the amount of unallocated (free) disk space and empty directory entries available on all mounted disks.

KILL <filespec/devspec>

The KILL command is used to delete the specified file or device from the system.

LIB

Display the primary VTOS 4.0 command library.

LINK <devspec> [prep] <devspec>

This command is used to link together input and output to/from multiple logical I/O devices. Once linked, all output sent to the first device will also be sent to the second device and all input requested from the first device may be satisfied from either device.

Note that the actual order of occurrence is important since input/output for the second device may not take place on the first device; i.e., it is a one-way linkage. Multiple LINKs of the same devices may produce an endless condition if the user is not careful.

LIST <filespec/devspec> ([NUM][LINE=l][HEX][REC=r][LRL=n]

The List command is used to display a listing of a file or device on the *DO device. Optionally, line number ‘NUM’ and/or hexadecimal listing ‘HEX’ may be specified. A starting line number ‘l’ for an ASCII file or a starting record number ‘r’ for a HEXadecimal file ay be specified. The logical record length ‘n’ may be specified for HEX listings. Two default file extensions exist, they are ‘TXT’ and if not found, all blanks will be used.

LOAD <filespec> [(X)]

The LOAD command is used to load a command file (/CMD) or a core image file (/CIM) into memory without execution. This command is especially useful for the single disk system where a program which does not reside on the VTOS system disk is to be loaded. You will be prompted for the appropriate disk insertions if the ‘X” option is specified. This command is primarily intended for loading high-memory drivers and BASIC functions which reside above address X’6000’. The default file extension is ‘CMD’.

MEMORY [(HIGH=addr)]

This command will reserve a portion of high memory for special utility routines, etc. such that VTOS and other programs which honor HIGH$ will not use the reserved memory area. The new HIGH$, if specified, must be lower than the one already in use by the system. If no address is specified, the currently existing address will be displayed. Note that this command comes in extremely handy for isolating expansion memory when [????] system behavior occurs.

example: **MEMORY (HIGH=X’FFFF’)**

This informs VTOS that no expansion memory is to be used.

PRINT <filespec/devspec> ([NUM][LINE=1][HEX][REC=r][LRL=n])

The PRINT command is used to print a listing of a file or device on the *PR device. Optionally, line numbering ‘NUM’ and/or hexadecimal listing ‘HEX’ may be specified. A starting line number ‘1’ for an ASCII file or a starting record number ‘r’ for a HEXadecimal file may be specified. The logical record length ‘n’ may be specified for HEX listings. Two default file extensions exist, they are ‘TXT’ and if not found, all blanks will be used.

PROT :d ([LOCK][UNLOCK][PW=pw][NAME=name][MPW=mpw])

The PROText command is used to alter the master protection information on a disk. The ‘LOCK’ and ‘UNLOCK’ options cause all non-system, non-invisible files to have both their ACCess and UPDate passwords altered to either the master password for that disk or to blanks, respectively. The other options request that either the master disk password cot the disk’s name be altered.

In all cases, the user will first be required to furnish the current master password if it is not passed via the MPW parameter for the disk in question with follow-on prompting if it is not passed via the MPW parameter. It should be noted that the PROT command provides two additional services which are not immediately apparent. These services perform certain maintenance functions needed for reading the directory of disks generated by non-VTOS systems. These services are required since certain disk-resident items are not examined, much less properly maintained, by non-VTOS systems. Any PROText command will automatically invoke these additional services including a dummy PROT command which alters a diskette’s name to itself.

PURGE :D [(MPW=mpw)]

The PURGE command provides a quick manner in which to eliminate unwanted ‘clutter’ which collects on a disk. All files (except BOOT and DIR) will be listed one at a time, and the user can respond with either ‘Y” to purge the file, or ‘N’ (or <ENTER> to leave the file intact.

Due to the severity of this command, the user will first have to provide the disk’s master password if it is not passed via the ‘MPW’ parameter.

RENAME <filespec> <prep> <filespec>

This command renames the file name provided in the first filespec to that provided in the second filespec. Dynamic defaults will be used for the name, extension, and password of the second specification (the new name). This will cause any of these fields which are not specified to be duplicated from the first file specification.

RESET <devspec>

This command will reset the specified logical device. If the device is a standard system device, it will be reset to its normal system power-on driver. If the device is a non-standard device, it will be set to a ‘NIL’ driver, i.e., no input accepted and all output ignored.

Note that if the RESET is issued without a <devspec>, a RESET ‘ALL’ will be performed. This will to set all system devices, clear all other devices, and attempt to reset HIGH$ to the top end of physical memory. Note that if any currently active foreground tasks are resident in high memory, HIGH$ will not be altered.

ROUTE <devspec> [(NIL)] [prep] <filespec/devspec>

The ROUTE commend creates and/or re-routes I/O for a logical device (devspec) to either a bit-bucket (NIL), a disk file, or a different logical device. If the routing is made to a disk file, both a device control block (DCB) and a blocking buffer will be dynamically allocated in high memory.

RUN <filespec> [(X)]

The RUN command causes the specified program to be loaded into memory and executed. This command is especially useful for the single disk system in that it allows the user to run programs which ado not reside on the VTOS system disk. When the ‘X” option is requested, the ‘RUN’ command will prompt the user for the appropriate disk to be inserted in the user’s drive.

This command is primarily intended for loading high-memory drivers and BASIC functions which reside above address X’6000’. The default file extension is ‘CMD’.

SET <devspec> [pre]) <filespec> [prep] [(params)]

The SET command is used to establish a new logical I/O device into the VTOS system. This driver will be active until the device is either RESET, ROUTEd, SET to a different driver, or the system is RESET. The default file extension is ‘DVR’.

example: **SET *CL TO RS232 (BAUD=300)**

SPOOL <devspec. [prep] <filespec> ([MEM=m][DISK=d])

The SPOOL command established a VTOS symbient queue for the specified output device. All output which is directed to the device after this command is issued will be placed in an output queue consisting of multiple buffers residing either in memory or on disk. The actual location of the buffers will be dynamically altered by the symbient process in order to provide the most efficient operating environment depending upon the total VTOS system processing requirements.

The <filespec>, if given, will be used for the disk buffer I/O. If no file is specified, a file named ‘dd/SPL” will be used where the ‘dd’ is the name of the SPOOLed device. The <filespec> may contain only a drive number (:n) if desired. You may optionally specify the amount of MEMory and DISK space to be used by the symbient through the m and d parameters, respectively. These are to be specified in 1024 byte ‘K’ blocks of storage. If not specified, they default to 1K of MEMory and approximately 5K of DISK, depending upon the disk type.

example: **SPOOL *PR**

SYSTEM (params)

The SYSTEM command allows you to change the basic configuration of your VTOS 4.0 system, its outward complexion, and the operation of the hardware attached to it. This command normally only configures the VTOS 4.0 system which is in memory. However, if you desire to make to changes more permanent, they may be saved to disk as your operating configuration through the use of the SYSGEN parameter. Note that some <params> require special hardware in order to operate properly.

The <params> which may be specified are:

BASIC --- to enter ROM resident BASIC (note that some hardware related parameters will be processed first.

FAST --- switches the processor to the high-speed clock.

SI.ON --- switches the processor to the standard speed clock.

BREAK[=flag] --- turns <BREAK> key detection ON or OFF.

BLINK[=flag] --- turns the VTOS blinking cursor ON or OFF.

BLINK[=nnn] --- turns on the blinking cursor and uses the respective ASCII character for the cursor.

LARGE --- sets up a large blinking cursor.

SMALL --- sets up a small blinking cursor.

TYPE --- this parameter enables the VTOS type-ahead feature. Your keystrokes are accepted and buffered until actually required as input to a program. Your input keystrokes will be displayed as they are accepted by programs for processing. While actually typing-ahead of the processor, auto-repeat will be disabled and a special ‘clear type-ahead’ feature will he performed if a <CLEAR><SHIFT>@ is entered. Since keyboard debounce is handled in a scheduled manner during type-ahead rather than simply placing the processor in a wait loop, your overall system performance will increase when keyboard type-ahead is active.

LOWER --- this enables lower-case output to your display.

JKL --- this parameter enables the screen-print feature. Once enabled, depressing the J, K, and L keys simultaneously will cause VTOS to print the contents of your display on the *PR device. The printing may be aborted at any time by hitting <BREAK>. Although the currently active program will be temporarily suspended during the print, active foreground tasks such as type-ahead or communications) will continue processing.

GRAPHIC --- this informs the JKL routine tat your printer is capable of printing graphic characters.

DRIVE=d --- the d parameter is the disk drive number to be used for any of the following disk related features.

DISABLE --- logically disable DRIVE d (turn it off, its busted).

ENABLE --- logically enable DRIVE d (turn it hick on, its fixed).

STEP=n --- alter DRIVE d’s stepping rage to n where n is 0-3. This is only valid for floppy drives with the following values: 0=3/6, 1=6/12, 2=10/20, 3=20/40 msecs for 5”/8” drives respectively.

DELAY<flag> --- turns extended motor-on delay ON/OFF for mini-floppy click drives. The default is ON which is a 1 second delay. If turned OFF, a delay of .5 sec will be used.

SYSGEN=<flag> --- this parameter is used for maintaining a configuration of the VTOS 4.0 system on disk according to your individual operating requirements. After changing the configuration of your VTOS 4.0 system in memory, executing this command will cause it to be written to the ‘SYSTEM’ disk in drive 0. This configuration will then be automatically loaded each time VTOS 4.0 is booted due to a <RESET> or power-on condition unless the <CLEAR> key is held down during the boot. If the flag is ‘OFF’, the special configuration will be eliminated from the disk. The loading of a system configuration will be eliminated from the disk. The loading of a system configuration will occur prior to the execution of any AUTO’d command lines. (See AUTO command).

Those items which are considered as user configurable are: altered device drivers (via the FILTER, LINK, RESET, ROUTE, or SET command), VTOS foreground tasks (via the CLOCK, DEBUG, SPOOL, or SYSTEM command), and special utility routines which you load into high memory (and protect with the MEMORY command).

TIME hh:mm:ss

This command is used to set the system's internal realtime clock. No display is associated with this command. (See CLOCK command).

TRACE [(flag)]

This command displays the user's program instruction counter (PC) on a realtime basis. Four hexadecimal digits are forcibly placed on the top line of the display and will continuously be updated with the current ptogram's execution address. Note that this display is only useful for assembly language programs.

VERIFY [(flag)]

The VERIFY command forces ALL disk writes to be performed with read-after-write verification regardless of the actual system call. Normally, only writes to VTOS system tables and explicit user calls to VERIFY will be performed with this read-after-write verification. The continuous usage of verification on all disk writes will insure the user with the maximum reliability possible, however, it will probably increase disk write time significantly since a read will occur after each and every write.

example: **VERIFY (ON)**

XFER <filespec>

This command exists primarily for the single disk system user. Its function is similar to COPY except that XFER will transfer a file from one diskette to another without requiring a VTOS system present on either of the two data disks. Appropriate user prompts will be made as each of the two data disks (referred to as 'Source' and 'Destination') and the 'System' disk (containing VTOS) are to be inserted into the drive. The filespec may contain a non-zero drive specification.

EXTENDED UTILITIES

The following utilities are supplied with VTOS 4.0:

```
BACKUP <filespec>:s [prep] :d
    ([MPW=mpw][SYS][INV][VIS][MOD][DRIVER])
```

This utility package is used to duplicate data from the source disk in drive ‘s’ to the destination disk in drive ‘d’. If the source disk contains a non-blank master password, you must provide it to the BACKUP program. If the destination disk is blank, it will automatically be formatted before the BACKUP takes place. If the destination is not blank, both its name and master password must match those of the source disk. In other words, the destination must either be blank or a previous backup of the source disk.

The files may be duplicated in whole or in part depending upon the presence or absence of a file ‘class’. If a class is given, only those files from the source disk which are members of the specified class will be duplicated. A ‘class’ is the set of files whose names and/or extensions start with those given in the <filespec>, and which are SYStem, Invisible, or Visible if the respective parameter is set. This is especially useful for updating to new releases of the VTOS operating system.

If the MODified parameter is specified, the additional class restriction of only duplicating files which have been created or modified since the last BACKUP (those with a ‘+’) will be invoked. This will be a significant time save if you have a large disk and must back it up using smaller media since you won’t have to BACKUP all files every time.

If the DRIVER parameter is specified, BACKUP will prompt appropriately for information regarding special disk drivers in order to generate a VTOS ‘SYSTEM’ disk which will operate on the specified hardware. This may require special software drivers on your disk prior to the execution of this command.

A mirror-image BACKUP of all cylinders containing valid data will be performed if no <class> is designated and the drive configuration permits it. Only those files which were on the source disk prior to the BACKUP will be on the destination disk after a mirror-image BACKUP. A mirror-image BACKUP is the fastest BACKUP possible.

The requested BACKUP operation may require more than one destination disk due to a difference in disk drive configurations. If this is the case, you will be requested to swap disks in the destination drive each time one is filled.

In the case of a dingle drive system where both the drive specifications are the same, the user will be prompted as to when each of the disks, ‘Source’, ‘Destination’, and ‘System’, should be inserted.

DISKS CONTAINING ‘VTOS’, IN WHOLE OR IN PART, ARE FOR YOUR USE ONLY AND ARE NOT TO BE DISTRIBUTED TO OTHERS.

FORMAT :d ([NAME=name][MPW=mpw] [SDEN][DDEN][SIDES=s][CYL=c][STEP=n])

The format utility will format a disk in order for the VTOS system to properly record data to/from the disk. The formats produced by the FORMAT and BACKUP utilities of TRSDOS are also compatible.

The disk's 'name' and master password 'mpw' may be passed via the command line. Additionally, if the drive configuration permits, the disk's recording density (single 'SDEN' or double 'DDEN'), along with the number of sides 's', the cylinder count 'c', and the stepping rate ‘n’ may be altered from the configuration defaults.

If the disk to be formatted already contains data of either a compatible or incompatible format, a message will be issued, along with the disk's name and creation date, informing the user of the circumstances and will then allow the user to either continue or abort the FORMAT operation.

PATCH <filespec> [prep] <filespec> [(YANK)]

This utility will, hopefully, be your lifesaver when a defect is found in either the VTOS system, its utilities, or user programs which run under VTOS. This utility will install a program patch to the program file whose name is given in the first <filespec>. The patching information will either be read in from the ASCII file whose name is given in the second <filespec> or, in the case of a one line patch, it may be contained within parenthesis on the command line.

PATCHes will have the name contained in the second <filespec> associated with them except in the case of a patch by record location. The YANK parameter will cause a prior patch which has the name of the second <filespec> to be removed from the file. Note that YANK can not remove a patch by record location.

The PATCHing instructions which may be specified are:

.comment
-- comment line

Drecord,byte=dd dd dd
-- patch starting at the specified record location. dd may be either hex (X’dd) or ASCII (‘dd).

X’addr’= dd dd dd
-- patch load address ‘addr’ to become the data byte stream ‘dd dd dd’.

Any of these commands may be followed with comment by 'placing a semicolon '; ' after the patch data.

The default file extensions are 'CMD' for the program file and 'FIX' for the patch file.

KSR/CMD <devspec>

This is a keyboard send-receive terminal emulator. The logical device specification for the RS232 interface to be used must he supplied. Note that this allows this program to function with any RS232 hardware for which a driver exists. Once active, the emulator will recognize the DUPLEX, ECHO, AUTO-LF, VTCOMM, VTOS, ON and OFF functions described under VTCOMM.

VTCOM/CMD <devspec>

This is a more advanced communications package than KSR in that it allows machine-to-machine communications using any of the following six logical devices: keyboard (*KI), display (*DO), printer (*PR), comm line (*CL), file input (Fl), and file output (FO).

All memory not used for drivers (*CL, KSM, etc.), is dynamically used for buffering the devices. It is not necessary to use the VTOS 4.0 SPOOLer in order for VTCOMM to buffer your output in memory, however, it your output requirement is quite large, the use of the SPOOLer is recommended.

In order to make the functions easier to use, and allow simultaneous KSM operation, the local functions are all implemented on the numeric keys. A tape containing each key’s function placed directly above the top row of keys makes it even easier. The key definitions are as follows, remember to hold the <CLEAR> key down during the keystroke:

1	KI	!	DUPLEX
2	DO	“	ECHO
3	PR	#	---
4	CL	$	AUTO-LF
5	FI	%	REWIND
6	FO	&	PEOF
7	---	‘	---
8	---	(	---
9	ID	)	VTCOMM
0	RESER	-	---
:	ON	*	---
-	OFF	=	VTOS

In order to turn a particular device or flag on or off, the function code for the device must first be [?????]

example: **11<CLEAR>-3h& <CLEAR>-:. will turn the printer on**

If you desire to use a disk file for either input or output, you must first identify the file by means of 'FI' or 'FO' followed by 'ID' and a VTOS filespec. You then logically turn the file 'ON' and 'OFF' as you would any of the other devices. You must be sure to terminate file output to disk when using 'FO' in order to properly close the file. This can be accomplished with the 'FO' and

'RESET' sequence. The additional functions of rewinding a file 'REW' or positioning to the end of a file 'PEOF' are avail [????]

example: **<CLEAR>-6 & <CLEAR>-9 will request an output file name, after typing it in and hitting <ENTER>, a <CLEAR>-6 & <CLEAR>-: will turn it on and record everything from that point on to disk. You MUST enter a <CLEAR>-6 & <CLEAR>-0 to RESET (close) the file.**

The DUPLEX flag controls local duplex mode where the ‘ON’ condition places the program in half duplex mode, and the ‘OFF’ condition places it in full duplex mode. The ECHO flag, on the other hand, controls the echoing of characters received from the communications line; if ‘ON’ everything will be echoed. The VTCOMM flag indicates that a compatible set of control codes are in use at the other end of the comm. Line. This will typically only be used for communicating with another VTCOMM program.

Note that VTCOMM will run in excess of 300 laud without any pauses, nulls, or other delays required from the transmitting side of the communications line. This is only possible since all disk I/O within the VTOS system is performed without indiscriminately disabling the interrupt system for extended periods of time as is the case with ether disk operating systems.

DEVICE DRIVERS

PR/DVR (params)

This is a printer driver which allows special printers to be used with the system. The driver may be set up with the following parameters for governing its actual operation:

LINES=1 -- to establish the maximum number of lines per page *such as 6 for mailing labels)

PAGE=p -- to establish the physical page's line count

CHARS=c -- to establish the maximum number of characters which will fit on one line

INDENT=i -- to set the indentation level to be used for lines which exceed c characters in length.

example: **SET *PR TO PR (CHARS=40,INDENT=5)**

RS232/DVR (<params>)

This is a general purpose driver for the asynchronous communications board that fits in the expansion interface. The parameters are used for setting the different characteristics of the driver. Any parameters which are required (such as the BAUD rate) but not specified will be defaulted to those values present in the switches on the RS232 board itself. The parameters are:

BAUD=b -- set the baud rate (any rate which the board will support)

WORD=w -- set the word length (5-8 bits)

STOP=s -- number of stop bits (1 or 2)

PARITY<=flag>
-- turn parity generation/checking ON or OFF

ODD -- use ODD parity

EVEN -- use EVEN parity

DTR<=flag> -- set DTR to constant TRUE

RTS<=flag> -- set RTS to constant TRUE

DSR<=flag> -- DSR must be TRUE before attempting any I/O

CD<=flag> -- CD must be TRUE before receiving any data

CTS<=flag> -- CTS must be TRUE before transmitting any data

RI<=flag> -- if RI is TRUE, set DTR=TRUE (auto-answer)

example: **SET *CL TO RS232 (BAUD=1200)**

KSM/DVR <filespec>

This driver is the keystroke multiplication feature of VTOS which permits re-definition of the keyboard keys. When this driver is enabled, a predefined user KSM file containing) up to 26 phrases is read into memory and saved for use when any one of the alphabetic keyboard keys is depressed while the <CLEAR> key is held down. When this action takes place, the phrase associated with that particular alphabetic key is automatically entered by the KSM driver just as though each of the keystrokes were entered manually. The keyboard shift keys are transparent to this feature in order to allow either upper or lower case keys to be used depending upon your operating environment.

The KSM file is comprised of from 1 to 26 phrases consisting of characters which are normally typed in manually. Each phrase should be terminated with an <ENTER>. In cases where you actually want to place a carriage return into a KSM phrase, simply use the semi-colon, it will be translated into an <ENTER> when the phrase is actually used. (See BUILD)

Function keys passed to the KSM driver which do not have an associated phrase will he passed on to the program which originally called for the input. See the '/KSM' sample files on your VTOS 4.0 disk.

example: **SET *KI TO KSM USING VTOS**

OPERATION UNDER BASIC

Several additional features are available under VTOS 4.0 when used with BASIC and BAICR version 2.2 and 2.3. These features are distributed on the VTOS 4.0 'MASTER' disk in the form of patches since BASIC is a copywritten work of MICROSOFT.

These features are:

1) BASIC will no longer ask for the memory size or the number of files when you start it. The memory size will default to the maximum size available without crashing into high memory routines (by using HIGH$) and the file count will default to three. You may specify these via the command line as follows:

   BASIC (<MEM=aaaa>,<FILES=nn>, <VAR).

   Where the VAR parameter specifies that you wish to use variable length files.

2) You may execute VTOS system commands directly from BASIC via the CMD statement. A slight format change in the statement his been made in order to implement this feature, however, all of the previous functions of the CMD statement are still available. The format is: CMD "command".

3) You may perform a screen-print operation with the new command CMD "P".

4) An open at end-of-file statement is available to add data to the end of an existing file. The format of the statement is: OPEN "E",lun,<filespec>. The "E" file type is identical to the "O" file type once the file has been opened.

5) Variable length file support under BASIC allows you to develop a data base using records shorter than 256 bytes without being burdened by the housekeeping this generally entails. VTOS will automatically handle the reading and writing of your, file regardless of the record length, taking into account all blocking, deblocking, pre-read, pre-write, and other accounting measures which are necessary in order to insure proper file integrity with nominal disk I/O overhead for both sequential and random files.

   In order to use this future, you must specify the 'VAR' parameter on the BASIC command line and then specify the variable record length 'lth' on the OPEN statement as follows: OPEN "i/o","lun,"filespec",lth

6) The LOC and LOF functions will return a logical record number when in use with a variable length file. That is, they will return the record number which your BASIC program passed to VTOS based upon a record length of 'lth', not based upon a physical sector length of 256 bytes.

7) BASIC programs which are declared as 'read-only' by means of the ATTRIBute command may now be properly protected by VTOS and BASIC. The program can be "RUN" and not even "LOAD"ed! If a 'BREAK' or error is encountered, the program will be wiped from memory. The VTOS DEBUG command will be disabled during the protected program's residency.

8) Your disk's directories are now readable under BASIC! Not just printable, but READABLE! This allows you to perform certain data base maintenance functions not possible under other systems.

In order to install the PATCH, you should copy your current copy of BASIC (2.2 or 2.3) to a VTOS 4.0 disk. [????] SYS!MAKE(BASIC)d

APPENDIX

APPENDIX A – SYSTEM OVERLAYS

The VTOS 4.0 operating system makes use of system overlays in order to keep down the minium amount of memory required by the system.

Certain VTOS system functions will require the loading of one or more of these system overlays (if not already loaded) in order to complete the execution of the requested function. Since using this technique requires that a VTOS 'SYSTEM' disk always reside in the drive zero, the actual disk space occupied by the system overlays on the VTOS 'SYSTEM' disk can be critical for single drive users.

The following list should help those users who wish to create a 'minimal' system configuration for their particular application through the use of the PURGE command.

SYS0/SYS

This is the resident system. It is required for any VTOS booting operations and may be purchased from any disks which are not going to be used for booting.

SYS1/SYS

This overlay contains the command interpreter and numerous other internal routines. You should never consider purging this overlay.

SYS2/SYS

This overlay is required in order to open disk files and logical devices. Like SYS1/SYS, this overlay also performs numerous other internal VTOS functions and should not be purged.

SYS3/SYS

This is the system close overlay. This overlay is required in order to close a disk file or logical device. It must not be purged.

SYS4/SYS

This system overlay contains the VTOS system error dictionary. Without this overlay, all errors will issue a message of the form 'SYSTEM ERROR nn' where nn is the VTOS system error number in hex. If this error format is sufficient, (or if errors are being trapped in BASIC) you may purge this overlay.

SYS5/SYS

This is the VTOS system debugging overlay. This overlay is required for all VTOS debugging operations and may otherwise be purged.

SYS6/SYS

This overlay is the largest of the VTOS system overlays. It contains the VTOS system LIBrary commands. This overlay should be one of the first to go if you don't desire to use ANY of the VTOS LIBrary commands.

SYS7/SYS

This overlay is required during program CHAINing operations. If you don't use CHAIN (horror the thought), purge it!

SYS8/SYS

This overlay performs several functions which are intrinsic to the operation of VTOS. You should never purge this overlay.