

ADS 1stPass Help Index

For help on a function key press that key now.

(Press F1 for information on "Using Help")

Press Alt-F1 for an index of all 1stPass commands.

Select one of the following options for more information.

1.0 Basic 1stPass Operation

1stPass is a simple command-driven program designed to give the user access to Personal Computer system hardware at the lowest and most detailed level, without having to "thumb in" programs using a software debugger. 1stPass accepts commands from the keyboard and executes them one at a time. If a command is not recognized as one of the 1stPass built-in commands, a script file by that name is opened and commands are read from the file and executed as if they had been typed at the keyboard. Nesting of script files is limited only by available memory.

Built-in commands allow access to PC memory within the first megabyte and to all I/O addresses by bytes, (16-bit) words, and (32-bit) doublewords. Additional commands provide for elementary testing and looping operations, operator communications and buffer management.

1.1 Help for 1stPass Screen Layout Display

The 1stPass main display screen comprises 4 windows (from top to bottom):

- Menu bar (top line)
- Response Window
- Message/status line
- Command Window

Both the Response Window and the Command Window have an associated scollable history buffer. The arrow and Page Up/Down keys may be used to scroll the window through the buffer.

Normally, both windows are idle. The command prompt (usually >) is displayed in the command window and the response window displays any output from currently executing commands. If an ordinary character key is pressed, the command window becomes active and its video attribute changes to indicate this fact. The command window remains active until the command is executed by pressing Enter or is deleted by pressing ESCape. The command window may also be activated by pressing ^, to recall the previous command from the history buffer.

Pressing PgUp activates scrolling in the response window, changing its video attribute and scrolling up one page in its history buffer. The ^, ^, PgUp, PgDn, Home and Endkeys may be used to scroll through the response history.

1STPASS.TXT

Pressing Tab or ESC returns the response window to its inactive (non-scrolling) state.

By default an inactive window uses normal video; an active window uses inverse video. See help for built-in variables and the sample script file setcolor for information on changing the video attributes.

1.2 Help for Editing Keys Display

The behavior of the cursor positioning and editing keys depends on the active display window:

If the Scrolling Response Window is Active:

HOME	Display oldest line in Response History
END	Display newest line in Response History
PgUp	Scroll Response Window up one page
PgDn	Scroll Response Window down one page
UP	Scroll Response Window up one line
DOWN	Scroll Response Window down one line
ESC	Deactivate Response Window
TAB	Deactivate Response Window
INS	Ignored
DEL	Ignored

If the Command Window is Active:

HOME	Move cursor to start of line
END	Move cursor to end of line
PgUp	Make Response Window active; scroll up
PgDn	No effect
UP	Display previous entry in Command history
DOWN	Display next entry in Command history
ESC	Cancel current Command entry
TAB	Activate Response Window
INS	Toggle between insert and overwrite mode
DEL	Delete character under the cursor

Pressing Control-Enter places command input in "repeat mode"; the command window does not clear when Enter is pressed.

1.3 Help for Command syntax Display

All 1stPass commands consist of a keyword optionally followed by one or more arguments. Keywords and arguments are separated by spaces, tabs or commas. The following special characters are significant:

;####(semi colon) separates multiple commands on a line.

(pound) ignores the rest of the line as a comment.

* (asterisk) as first character of a line displays that line in the response window before it is interpreted. Commands read from script files are printed to the screen only if the echo

1STPASS.TXT

option is enabled or the command is prefixed with an asterisk.

\$ (dollar) indicates script file argument or variable substitution.

Commands and most keyword arguments are case-insensitive; a command may be abbreviated to its shortest unique substring. The number of arguments on each command line is limited to 10 and the length of each argument is limited to 128 characters.

Command arguments are of several types:

KEYWORD

A keyword argument must be one of a specific set of strings. For example, the buffer fill command, `bfill`, takes a pattern argument which must be a legal pattern name (e.g. `0`, `1A5`, `ckb`, `inc`, ...) All keywords except buffer formats are case-insensitive.

NUMBER

Numeric arguments are specified according to the C language conventions for numbers, with the following extensions:

Number Begins with	Base is
0	8
0x or 0X	16
0d or 0D	10
Other	numbase

Some commands have their own default base, as noted in the command descriptions.

ADDRESS

Address arguments are memory addresses within the first megabyte of PC system memory. A memory address argument may be given as either a 20-bit "flat" address or in segment:offset form as two 16-bit numbers separated by a colon.

An address in segment:offset form is equivalent to the "flat" address obtained by shifting the segment portion left 4 bits and adding the offset portion to it. For example, the following two commands address the same memory location (default base 16 assumed):

```
rmemw 1200:0345 and rmemw 12345
```

STRING

A string argument is a character string containing no comma or white space (space or tab) characters. Typical string arguments are symbol names and operators in arithmetic expressions.

TEXT

A text argument is a character string extending to the end of the command. Part or all of the string may be enclosed in quotes (") which are required if the string contains a semicolon (;), pound (#), quote or control character. The quoted and unquoted substrings are concatenated to form the final argument. The backslash (\) serves as an escape character. Use \" to include the quote character in the string; \\ to include the backslash character itself. In addition, the following C language escape sequences are recognized:

\a Alert (bell)
 \b Backspace
 \f Form feed

\n Newline
 \r Carriage return
 \t Horizontal tab
 \v Vertical tab
 \ddd ASCII character: octal notation
 \xdd ASCII character: hex notation

Note that variable substitution is performed only in the un-quoted portions of string arguments.

1.4 Variables & Variable Substitution

Symbolic names may be defined for variables, each containing one 32-bit number. Variable substitution occurs in a 1stPass command whenever

- (a) A variable name is used for a number argument
- (b) A variable name in parentheses follows a dollar sign (\$) in a string or text argument.

When the dollar sign form is used, the format of the string to be substituted may be specified by following the variable name with a colon and a format specifier acceptable to the C library printf routine. For example, the string \$(abc:x) substitutes the value of the variable abc converted to hexadecimal; while \$(abc:3o) converts the value as a three digit octal number.

1.5 Help for Built-In Variables Display

The following variables are built into 1stPass. Note that unlike user-defined variables, some of these are not 32-bit numbers and the ones marked (*) are read-only. The value of an expression in a set command will be truncated, if necessary, before storing to a word or byte variable. The result of setting a variable to an illegal value (e.g. std_i osi ze = 5) is undefined.

Name	Contents
----	-----
abortmode	If zero, pressing ESCape exits only the current program loop. If non-zero, all nested loops and scripts are exited on ESCape.
echo	Contains bit flags which control printing of commands to response window and/or log file. See echoctl command for more info.
eprintlimit	Maximum number of buffer compare error lines to print (Initial value is ??.)
errlimit	Maximum number of errors to report on buffer

1STPASS.TXT
compares (Initial value is ??.)

numbase Default base for converting numbers from command input. (Initial value is 10.)

rtc * The MS-DOS real time clock.

std_bfrsize Default element size for buffers: 1, 2 or 4 bytes. (Initial value is 1.)

std_iosize Default size of I/O and memory read/write operations: 1, 2, or 4 bytes. (Initial value is 1.)

step If non-zero, scripts are executed in single-step mode.

timeout Number of milliseconds for read-wait operations if no timeout is given on the command. (Initial value is 0, meaning no timeout).

.p * I/O port address used in last rio... or wio... command

.m * Address used in last rmem... or wmem... command

. * Last I/O port or memory address used in any read or write command.

= *#####Value of last expression evaluated by a set command, or the result of a read command.

@SA0 Video attribute for inactive response window

@SA1 Video attribute for active response window

@SA2 Video attribute for inactive command window

@SA3 Video attribute for active command window

@SA4 Video attribute for menu bar (See the example script file setcolor for more information on setting screen attributes.)

The following variables affect the configuration of 1stPass and may only be changed by set commands in the program initialization file. Once the program has initialized, these variables are no longer accessible.

@rsp_history Number of lines maintained in the scrolling history buffer for the response window. (Default value is 200.)

@cmd_history Number of lines maintained in the scrolling history

1STPASS.TXT
buffer for the command window. (Default value is 20.)

1.6 Help for Data Buffers Display

Any number of Data Buffers may be defined, limited only by available memory. A buffer is defined and named by the `bdefine` command, which may also specify its size and other attributes. Buffers may be filled with data patterns, read, written, viewed and compared. Fill, compare and view operate on the entire buffer, while reading and writing take place one data element at a time. Each buffer has a pointer to the next element to be loaded or stored and an increment value used to modify the pointer when a buffer read or write command is executed.

Available buffer manipulation commands are:

<code>bdefine</code>	Define buffer
<code>bview</code>	Display buffer
<code>bfill</code>	Fill buffer with data pattern
<code>bformat</code>	Set buffer display format
<code>bset</code>	Set buffer attributes
<code>bget</code>	Get buffer attributes
<code>bcmp</code>	Compare buffer contents
<code>bload</code>	Load buffer from disk file
<code>bsave</code>	Save buffer to disk file
<code>bltxt</code>	Load buffer with string

The `bset` and `bget` commands modify and examine the following buffer attributes:

<code>type</code>	Buffer element size: 1, 2 or 4 (bytes).
<code>size</code>	Number of elements of type bytes contained in buffer.
<code>increment</code>	Increment (in bytes) to be added to pointer after each read/write operation.
<code>pointer</code>	Current buffer pointer expressed as an offset, in bytes, from the buffer base address.
<code>address</code>	Base address of the buffer, in "flat" (20-bit absolute address) format.

1.7 HELP Test key

This key is used solely to display the help text from this section for assistance in debugging help formatting.
theseshould be BOLDwords

So should these

And in the middle of this line should be twoBOLD words. But if I want

to use the one-line form to bold an entire phrase while filling, can I do that, too?

commands

1.8 USING HELP

Activate the help system by pressing F1. If the command window is empty, help information for the current screen will be presented.

If the command window contains a character string, help for any command matching the string will be returned. If the string cannot be identified as a unique command, a menu of all matching commands will be shown.

Pressing ALT-F1 at any time displays the index of all 1stPass commands.

Press space or PgDn for next page.

When help text is displayed, the following keys have special functions:

Space	Go to next page
PgDn	Go to next page
PgUp	Go back one page
HOME	Go to first page displayed
END	Go to last/next page
ESC	Exit help

The markers ^ and ^ in the right-hand border of the Help window indicate the PgUp and PgDn keys, respectively, may be used to scroll through the current topic.

1.9 Exit 1stPass

Pressing this key will ask for confirmation, then exit the program by executing the equivalent of a "quit all" command.

Press Y to exit; any other key to cancel the operation.

1.10 User-definable function key

1STPASS.TXT

This key may be defined by the "keydef" command.

Type the command `keydef` (with no arguments) to see the current function key definitions.

2.0 rio Read from I/O space

```

rio      port  [variable]  Read any size
riob     port  [variable]  Read byte
riow     port  [variable]  Read word
riod     port  [variable]  Read doubleword

```

Reads specified size item from port. If variable is present, stores value there; otherwise, displays the value in decimal and hexadecimal. In either case, the value is left as the last computed result (the variable).

The generic command, rio reads the size item specified by the variable `std_ysize`.

2.1 rio-w Read from I/O space, test and wait

```

rio-w    port  target [mask [time]]  Read any
riob-w   port  target [mask [time]]  Read byte
riow-w   port  target [mask [time]]  Read word
riod-w   port  target [mask [time]]  Read dbl wd

```

This command reads the specified size item from port and compares it against target. If mask is present, it is and'ed with the input item before making the comparison.

The command sets the last computed result (the variable to 1 if the comparison is equal and 0 if not equal. Time, if present, is the

number of milliseconds the command should continue repeating the test.

The generic command, `rio-w` reads the size item specified by the variable `std_size`.

If time is omitted, the variable `timeout` determines the timeout.

2.2 `rmem` Read from Memory Space

```
rmem    address [variable] Read any size
rmemb   address [variable] Read byte
rmemw   address [variable] Read word
rmemd   address [variable] Read doubleword
```

Reads specified size item from address. If variable is present, stores value there; otherwise, displays the value in decimal and hexadecimal. In either case, the value is left as the last computed result (the variable).

Address must be a "flat", 20-bit address in the range 0 - 0xFFFF. On the command line, the address may be specified in Segment:Offset form as two numbers in the range 0 - 0xFFFF separated by a colon. (e.g. A000:0002).

The generic command, `rmem` reads the size item specified by the variable `std_size`.

2.3 `rmem-w` Read from Memory Space, Test & Wait

```
rmem-w  port  target [mask [time]] Read any
rmemb-w port  target [mask [time]] Read byte
rmemw-w port  target [mask [time]] Read word
rmemd-w port  target [mask [time]] Read dbl wd
```

1STPASS.TXT

This command reads the specified size item from address and compares it against target. If mask is present, it is and'ed with the item read before making the comparison.

The command sets the last computed result (the variable to 1 if the comparison is equal and 0 if not equal. Time, if present, is the number of milliseconds the command should continue repeating the test.

Address must be a "flat", 20-bit address in the range 0 - 0xFFFF. On

the command line, the address may be specified in Segment:Offset form as two numbers in the range 0 - 0xFFFF separated by a colon. (e.g. A000:0002).

The generic command, rmem-w reads the size item specified by the variable std_i size.

If time is omitted, the variable "timeout" determines the timeout.

2.4 wio Write to I/O space.

wio	address	value	Write any size
wiob	address	value	Write byte
wiow	address	value	Write word
wiod	address	value	Write doubleword

Writes specified size value to port.

The generic command, wio writes the size item specified by the variable std_i size.

2.5 wmem Write to Memory Space

```
wmem  port  value  Write any size
wmemb port  value  Write byte
wmemw port  value  Write word
wmemd port  value  Write doubleword
```

Writes specified size value to address.

Address must be a "flat", 20-bit address in the range 0 - 0xFFFFF. On the command line, the address may be specified in Segment:Offset form as two numbers in the range 0 - 0xFFFF separated by a colon. (e. g. A000:0002).

The generic command, wmem writes the size item specified by the variable std_i size.

2.6 sbitio Set bit in I/O space

```
sbitio port  value  Set bit
sbitiob port  value  Set bit in byte
sbitiow port  value  Set bit in word
sbitiod port  value  Set bit in doubleword
```

Sets bit(s) in an I/O register by reading port, or-ing mask with the item read, then writing the result back to port. Note that this requires the device register to be readable as well as writeable.

The size item read/written is determined by the command used; the generic command, sbitio operates on the size item specified by the variable std_i size.

2.7 sbitmem Set bit in memory space

```
sbitmem address value Set bit
sbitmemb      address value Set bit in byte
```

```

sbitmemw    address  value  Set bit in word
sbitmemd    address  value  Set bit in doubleword

```

Sets bit(s) in a memory location by reading address, or-ing mask with the item read, then writing the result back to address. As with sbitio, this requires the addressed register to be readable as well as writeable.

The size item read/written is determined by the command used; the generic command, sbitmem operates on the size item specified by the variable std_size.

2.8 rbitio Reset bit in I/O space

```

rbitio port  value  Reset bit
rbitiob port  value  Reset bit in byte
rbitiow port  value  Reset bit in word
rbitiod port  value  Reset bit in doubleword

```

Resets bit(s) in an I/O register by reading port, and-ing the complement of mask with the item read, then writing the result back to port. Note that this requires the device register to be readable as well as writeable.

The size item read/written is determined by the command used; the generic command, rbitio operates on the size item specified by the variable std_size.

2.9 rbitmem Reset bit in memory space

```

rbitmem address  value  Reset bit
rbitmemb    address  value  Reset bit in byte
rbitmemw    address  value  Reset bit in word
rbitmemd    address  value  Reset bit in doubleword

```

Resets bit(s) in a memory location by reading address, and-ing the complement of mask with the item read, then writing the result back to

address. As with `sbitio`, this requires the addressed register to be readable as well as writable.

The size item read/written is determined by the command used; the generic command, `sbitmem` operates on the size item specified by the variable `std_ysize`.

2.10 `pbitio` Pulse bit in I/O space

```
pbitio  port  mask [rept [Ton [Toff]]] (any)
pbitio  port  mask [rept [Ton [Toff]]] (byte)
pbitio  port  mask [rept [Ton [Toff]]] (word)
pbitio  port  mask [rept [Ton [Toff]]] (dbwd)
```

This command first sets, then resets bit(s) in an I/O port register by the following sequence of operations:

- 1) Read item from port
- 2) OR mask with the item and write back to port.
- 3) Delay Ton milliseconds.
- 4) AND the complement of mask with the item and write back to port.
- 5) Delay Toff milliseconds.
- 6) Repeat steps 1-5 `rept` times.

If any of `rept`, `Ton`, or `Toff` is omitted, its default value is 1. (e.g. If only `port` and `mask` are given one pulse with 2ms period, 50% duty cycle is generated).

Note that the register size read/written is determined by the command and that the register must be readable as well as writable.

2.11 `pbitmem` Pulse bit in memory space

```
pbitmem address  mask [rept [Ton [Toff]]] Pulse bit
pbitmemb          address  mask [rept [Ton [Toff]]] Pulse bit in byte
```

1STPASS.TXT

pbitmemw address mask [rept [Ton [Toff]]] Pulse bit in word
 pbitmemd address mask [rept [Ton [Toff]]] Pulse bit in doubleword

This command first sets, then resets bit(s) in a memory location by the following sequence of operations:

- 1) Read item from address
- 2) OR mask with the item and write back to address.
- 3) Delay Ton milliseconds.
- 4) AND the complement of mask with the item and write back to address.
- 5) Delay Toff milliseconds.
- 6) Repeat steps 1-5 rept times.

If any of rept, Ton, or Toff is omitted, its default value is 1. (e.g. If only address and mask are given one pulse with 2ms period, 50% duty cycle is generated).

Note that the register size read/written is determined by the command and that the register must be readable as well as writable.

2.12 rbfrr Read item from buffer

rbfrr buffer [variable] Read nominal size
 rbfrrb buffer [variable] Read byte
 rbfrrw buffer [variable] Read word
 rbfrrd buffer [variable] Read doubleword

Reads specified size item from buffer. If variable is present, stores value there; otherwise, displays the value in decimal and hexadecimal. In either case, the value is left as the last computed result (the variable). The item actually read is located at buffer's base address plus its current offset. After the read operation, the current increment for buffer is added to the pointer.

The generic command, rbfrr reads the nominal size item specified when the buffer was defined.

2.13 wbfr Write item to buffer

wbfr buffer value Write nominal size

wbfrb buffer value Write byte
 wbfrw buffer value Write word
 wbf rd buffer value Write doubleword

Writes specified size value to buffer. The item actually modified is located at buffer's base address plus its current offset. After the write operation, the current increment for buffer is added to the pointer.

The generic command, wbfr writes the nominal size item specified when the buffer was defined.

2.14 bdefine Define buffer

bdefine name [length [type]] Define buffer
 bdefine name List named buffer
 bdefine List all buffers

If no arguments are given, all defined buffers are listed. If a buffer name only is given, that buffer is listed. If more than one argument is given, the buffer name is defined to contain length elements of element_size bytes.

If length is zero, buffer name is deleted. Once a buffer has been defined, its length and type (element_size) may not be changed; it must be deleted and re-defined.

The display resulting from a "list buffer" command contains the buffer length (in number of elements), element size, current increment (in bytes), base address and current read/write pointer (both in segment:offset format).

2.15 bview View Buffer Contents

bview bname-1 [bname-2 [bname-3 [bname-4]]]

The display is switched to the Buffer Display Screen where the buffers named as arguments are displayed in columns.

2.16 bfill Fill buffer with data pattern

bview name patname [length]

Buffer name is filled with pattern patname. If length is given, only the first length elements will be filled. Otherwise, the entire buffer will be filled.

Data patterns apply to the nominal data element (byte, word or doubleword) defined for the buffer, unless specifically stated, (e.g. "incrementing bytes"). Available data patterns are:

0 All zeros.

1 All ones (bytes of FF hex)

5 Alternating 1/0 (bytes of 55 hex)

A Alternating 0/1 (bytes of AA hex)

A5 Alternating elements of 5555.../AAAA...

0F Alternating elements of 0000.../FFFF...

ckb Byte checkerboard: alternating bytes of 00/FF

ib Incrementing bytes: 00, 01, 02, 03, etc.

db Decrementing bytes: FF, FE, FD, FC, etc.

iw Incrementing words: 0000, 0001, etc.

dw Decrementing words: FFFF, FFFE, etc.

inc Incrementing buffer element.

dec Decrementing buffer element.

w1b Walking 1 in byte: 01, 02, ..., 80

w0b Walking 0 in byte: FE, FD, ..., 7F

w1w Walking 1 in word: 0001, 0002, ..., 8000

w0w Walking 0 in word: FFFE, FFFD, ..., 7FFF

w1l Walking 1 in doubleword: 00000001, ..., 80000000

w0l Walking 0 in doubleword: FFFFFFFE, ..., 7FFFFFFF

2.17 bformat Set buffer display format

1STPASS.TXT

bformat fmt-1 [fmt-2 [fmt-3 [fmt-4]]]

Defines display format for up to four columns on the Buffer Display screen. If fewer than four arguments are given, format codes are applied to display columns in order from left to right. Valid format codes are listed below. These codes are displayed in the header line on the Buffer Display screen, to the right of the buffer name.

- Retain current format.

b 8-bit bytes

W Big-Endian 16-bit words. *

w Little-Endian 16-bit words. **

L Big-Endian 32-bit words. *

l Little-Endian 32-bit words. **

F32 32-bit IEEE floating point numbers.

F64 64-bit IEEE floating point numbers.

----- .br

* Lowest memory address is most significant byte

** Lowest memory address is least significant byte

2.18 bset Set buffer attributes

bset name item value

Sets the attribute item of buffer name to value. Attributes which may be set are:

increment The number of bytes added to the current buffer pointer after each buffer read or write operation.

pointer The offset into the buffer of the next item to be read or written.

2.19 bget Get buffer attributes

bget name item [variable]

Gets the value of attribute item of buffer name. If variable is

1STPASS.TXT

present, stores value there; otherwise, displays the value in decimal and hexadecimal. In either case, the value is left as the last computed result (the variable).

Valid attributes names are:

type Buffer element size (1, 2 or 4)

size Buffer size in units of type.

increment Increment in bytes to be added to buffer pointer after each read or write command.

pointer Current offset into buffer for next read or write command.

address Physical base address of buffer, as a 20-bit "flat" address.

2.20 bcmp Compare buffer contents

bcmp buffer-1 buffer-2

Compare the contents of buffer-1 and buffer-2, displaying the differences on the Buffer Display screen.

2.21 bload Load buffer from disk file

bload buffer filename Load whole buffer
bload buffer filename length Partial load
bload buffer filename offset length

Loads length data elements into buffer, from file filename. The transfer begins at the offset'th element of buffer and continues for length elements. If offset and length are omitted, they default to 0 and the size of buffer, respectively. If the second form is used, offset is 0 and length elements are loaded.

2.22 bsave Save buffer to disk file

bsave buffer filename Save whole buffer
bsave buffer filename length Partial save
bsave buffer filename offset length

Saves length data elements from buffer, to file filename. The transfer begins at the offset'th element of buffer and continues for length elements. If offset and length are omitted, they default to 0 and the

size of buffer, respectively. If the second form is used, offset is 0 and length elements are saved.

2.23 `bl dtxt` Load buffer with string

`bl dtxt name string`

Loads buffer `name` with the `string` argument. The string is copied to the buffer's starting address verbatim without any newline or end-of-string (zero byte) characters being added. The string will be truncated if longer than the buffer.

The buffer pointer is not used or altered by this command.

2.24 `del ay` Delay execution

`del ay count [units]`

Delays execution based on the high-resolution PC timer (approximately 840 nsec/tick). The `units` argument determines how `count` is interpreted:

omitted	Delay count (840 nsec) ticks
m	Delay count milliseconds
s	Delay count milliseconds

2.25 `keydef` Define function key

`keydef code [label cmdstr] Define key`
`keydef List defined keys`

Associates a function key with a 1stPass command string, `cmdstr`. `Code` is the number of the function key to be assigned (2-9 are available for user assignment) and `label` is a string of 8 or fewer characters which will be displayed on the menu bar for the function key.

The `cmdstr` may contain any legal 1stPass command construct. Note that if multiple commands separated by semicolons are used, the `cmdstr` needs to be enclosed in quotes (") to prevent the command from being terminated prematurely.

If no arguments are given, the definitions for the currently defined keys are displayed.

2.26 `echo` Echo Text to Screen

1STPASS.TXT

```
echo    text    Display one line
echo    <<      Begin block display
```

Normally, text is displayed to the screen. If text is "<<", then lines are read from the current input source and displayed until a line consisting of only a period (.) is read. This option can be used in scripts to display whole screens full of text at one time. For example, the following command sequence will display a brief menu on the screen:

```
echo <<
      First menu option    - F1
      Second menu option   - F2
      Third menu option    - F3
      .
```

2.27 echoctl Control output to screen/logfile

```
echoctl option ... Enable echo option
echoctl -option ... Disable echo option
```

Set or clear flag bits in the variable echo which controls routing of output from 1stPass. Available options are listed below. Name is the option name used with echoctl, Mask is the hexadecimal value of the associated bit in echo:

Name	Mask	Function
file	0001	Echo script file input
interp	0002	Echo as executed
key	0020	Echo keyboard input
logerr	0004	Log error messages
loghelp	0008	Log help info
logkey	0010	Log keyboard input (tagged)
off	0000	All options off

Note: the logkey option logs keyboard input with an attached tag to simplify extracting keyboard input from the log file to create scripts.

2.28 msg Insert Comment in Log

```
echo    text    Display one line
echo    <<      Begin block display
```

Normally text is displayed to the screen and written to the logfile, if logging is enabled and a logfile defined. If text is "<<", then lines are read from the current input and displayed until a line consisting of only a period (.) is read. This command is the same as echo, with the added output to the log file.

2. 29 pause Suspend Execution

```
pause text
```

The program displays text and pauses waiting for a keystroke. Pressing <ESC> will cause the program to exit any loop(s) it is executing, depending on the state of abortmode. Pressing any other key will allow the program to continue normally.

2. 30 log Establish Message Log File

```
log filename      Start logging to filename
log              Discontinue logging
log ?            Show logging status
log .           Resume logging to prior file.
```

Opens filename as message log file. If no argument is given, rdcmd

closes the current log file. If argument is '.', resumes logging to same file opened (and closed) previously.

2. 31 quit Exit Script File

```
quit              Exit current script file
quit all         Exit all script files
```

If "quit all" is used, all script file execution is terminated. If argument is omitted, the current script file is closed. If no script file is open (i.e. this command is entered from the keyboard at the base level), rdcmd terminates execution.

2. 32 if Conditional Execution

2. 33 else Conditional Execution

2. 34 fi Conditional Execution

```
if if_expr       Begin conditional block
else             Complement conditional state
fi              End conditional block
```

The expression if_expr is evaluated and, if the result is TRUE, command execution continues with the next sequential command. If the result is FALSE, execution is suspended until an else or fi command is encountered. When the else command is encountered, if commands are being executed, execution is suspended until the fi command is read;

1STPASS.TXT

if commands are being skipped, execution resumes. A script file containing an if command must also contain the matching fi.

A valid if_expr may be any of the following forms, where varname is any built-in or user variable name, string is any symbol or script file argument (e.g. \$2) and value is any number of defined variable:

varname True if varname is defined and has a non-zero value

DEF varname True if varname is defined.

UNDEF varname True if varname is not defined.

BLK string True if string is blank.

NBLK string True if string is not blank.

varname rel value True if varname and value have the relation rel. Legal rel's are: == (equal) != (not equal) > (greater than) < (less than) >= (greater than or equal) <= (less than or equal).

2.35 repeat Repeat Command Sequence

2.36 loop Mark end of Command Loop

repeat [num] Begin command loop
loop [if_expr] Mark end of command loop

Repeats all commands between repeat and loop num times. If num is omitted, the loop repeats indefinitely. The optional if_expr may be any of the expressions accepted by the if command. If if_expr is absent or its value is TRUE, the loop continues with the command following repeat. If the if_expr is FALSE, the loop terminates, irrespective of the repeat count. An indefinite loop will only be exited when the if_expr evaluates to FALSE or the user presses the <ESC> key. A script file containing a repeat command must also contain the matching loop.

2.37 set Set Variable to Value

set varname [expr] Set variable to value

1STPASS.TXT

```
set . [expr] Re-use last varname
set varname = Re-use value of last expr
set ? List known symbols
+varname Set varname to 1
-varname Set varname to 0
```

Varname may be a 1stPass built-in variable or a user variable. If expr is present, varname is set to the value of expr and that value is stored as the last computed result (the variable .). Expr may be a number or a simple arithmetic expression consisting of a maximum of 10 variables, numbers and operators, all separated by white space. For example:

```
set l2 l1 + 3
```

```
set tx 2 * ( t2 - t1 )
```

See help on arithmetic expressions for a list of legal operators.

If expr is absent, the current value of varname is displayed. If varname is a question mark (?), names of all known variables are displayed. The period (.) used as a varname is a shorthand for the last variable name used.

2.38 ! Spawn Subprocess

```
! command [arg1] [arg2] [arg3] ... [arg9]
```

A subprocess is spawned, executing the command interpreter defined by the symbol COMSPEC, and is passed the command and any arg's.

2.39 ? Show command syntax

```
? List all known commands
? command ... List syntax for command.
? keyword List valid keyword values.
```

The list of known commands and their syntaxes is displayed. The following abbreviations are used for argument types.

```
str Character string argument
num Number: default base is (numbase)
dnum Number: default base is decimal
onum Number: default base is octal
xnum Number: default base is hexadecimal
unum Unsigned Number: default (numbase)
text Text string argument
num|opr Number variable name or arithmetic operator
adrs Memory address argument
```

```
<name> Keyword argument of type name
[arg] arg is optional
```

If argument(s) are present on the command line, the syntax will be shown for only the named commands. If a partial command name is given, all matching commands will be displayed. Requesting help on a keyword

1STPASS.TXT

type will show a list of valid keywords for that type. For example, ?
pattern lists the valid <pattern> keywords which may be used with the
bfill command.