MEX: a Modem Executive for CP/M-80

USER'S GUIDE

TABLE OF CONTENTS

APPENDICES

1) Overview: What is MEX?

MEX is an acronym (sort of) for Modem EXecutive; its purpose is to build upon the capabilities provided by various terminal emulation and file transfer programs written for CP/M-80. MEX provides, in one program, a phone-number librarian and editor (supporting mnemonnic 12-character names for phone numbers and load/save for phone number files), a powerful autodialer (allowing lists of numbers to be called repeatedly until one answers with a modem tone; the entire list may be repeated any number of times), a file transfer facility supporting three common protocols (and "batch" file transfers), a "dumb-terminal" emulator that includes the ability to dynamically define multiple-line character strings under a single key (and save/load sets of keystrings to and from disk files), a STAT command that allows you to examine and change a wide range of operating parameters (especially useful with the CLONE command, which allows you to save a new copy of MEX to disk, including any currently defined phone numbers and keystrings), a batch-command file processor that allows the program to take its input from a disk file (allowing unattended use of MEX), and an on-line help facility (employing CP/M's random-access file accessing functions for quick access) that completely documents the program.

MEX combines the capability of many public domain utilities, and is a superset of such programs as MODEM2, MODEM7 and MDM. MEX provides more functionality than many commerical programs, and is gaining ground on the top-of-the-line modem packages, such as Microstuf's Crosstalk (TM). But enough horn blowing ... let's get on with it.

2) Getting started: the physical modem overlay

First, let me note that MEX, as distributed, contains no modem I/O at all; it will run on your computer as is (in fact, you can do an initial evaluation of MEX without doing any installation at all), but commands dealing with the modem will elicit an error message. To be fully functional, MEX needs modem driver code (written in 8080 assembly language) installed internally; this driver code is called the physical modem overlay.

If you're a user of Irv Hoff's MDM7 program, you have an easy progression path: MEX uses a compatible overlay format, and thus almost any MDM overlay may be loaded directly into MEX. In fact, you can use the MLOAD program distributed with MEX to install the overlay relatively painlessly (the use of the dreaded DDT is not required).

In addition, there are a large number of overlays available, written specifically for MEX, that make available the full power of the program.

If you don't have a MEX or MDM7 overlay, you'll have to write your own, or, if you're not proficient in assembly language programming, find someone to write one for you. If you have to take this route, use the Godbout Interfacer overlay distributed with MEX (named MXO-GBxx.ASM, where xx is the revision level) as a model. This file is a full-featured overlay, containing modem dialing drivers and a fully functional SET command (SET is a user-defined command, used for such things as baud-rate selection, answer/originate modem setup, etc). You can implement as little or as much of the "fancy stuff" as you like. If you redistribute your overlay, please follow the naming conventions I've established for MEX overlay files: "MXO-nnxx.ASM", where nn is a 2-letter code identifying the overlay, and xx is the revision level.

Once you have an overlay file, assemble it with ASM or MAC, then bind it into the system with MLOAD. The following example as-

sumes you're using the Godbout overlay; if not, just substitute the name of your overlay for MXO-GB10:

```
ASM MXO-GB10                          <<--- assemble the overlay
MLOAD MEX.COM=MEX10.COM,MXO-GB10      <<--- load a new MEX.COM
```

-Smartmodem overlay installation

If you have a Smartmodem (Hayes, USR, etc), you'll want to take advantage of one of the the Smartmodem logical overlays (MEX has no inbound Smartmodem code); these overlays come in two varieties, and you can tell one from the other by the first three characters of the filename: MXO for the (older) "small" overlay, and "MXM" for the newer and more capable overlay. Note that the MXO overlay type is shared (confusingly enough) with the MXO physical overlays (although I hope MXO physical overlays will die

out eventually as they are upgraded to MXM types). You should be aware the the MXO logical overlay is located at the top of the overlay area (about 500 bytes starting at 0B00H), and will work with any physical overlay that terminates before 0B00H (all known overlays do), while the newer MXM overlays require about 1000 bytes, and start at 0900H (you should check that your physical overlay ends before this point; most do).

The Smartmodem overlays provide autodial capabilities and, optionally, disconnect capability, using Smartmodem commands.

Use the following example as a guide for installing a Smartmodem overlay and your physical overlay simultaneously (the Smartmodem overlay is named MXM-SMxx.ASM -- xx is the revision level -- and MXO-GB10.ASM is the name of a typical MEX overlay; replace the former with the name of your own overlay):

```
ASM MXO-GB103                          <<--- assemble phys. overlay
ASM MXM-SM10                          <<--- asm. Smartmodem overlay
MLOAD MEX.COM=MEX11.COM,MXO-GB10,MXM-SM10 <<--- build new MEX.COM
```

(The order is important in the MLOAD command line due to the structure of the overlays).

Some notes about the preceeding discussion:

The version of MLOAD used MUST be 2.0 or greater; earlier versions didn't have the ability to load a HEX file over a non-HEX file.

A very few MDM7 overlays are not compatible; such overlays are typified by the requirement that they be revised with each suc-

ceeding revision of MDM7. These overlays typically implement dialing facilities for unusual modems (actually non-PMMI and non-Smartmodem), and require specific addresses within MDM7 in order to function. This was necessary because MDM7 does not provide a redefinable interface for dialing code (in spite of the overlay table entry present for the dialing routine), and worked 'naturally' only with Smartmodems and PMMIs. As a result, the majority of overlays written do not contain any dialing code, and thus will function correctly with MEX (note that MEX allows a fully re-definable dialing vector -- see MXO-PMxx.ASM for details -- and thus may be extended for use with virtually any modem).

3) Some precepts: Filespecs, string arguments, and multiple command-lines

Before detailing MEX's command set, there are a few topics we must address that apply generally throughout MEX.

MEX supports the same drive/user specification for files as Rick Conn's ZCPR and ZCPR2: A file may be prefixed by either the drive name, the user number or both (this extends to batch file transmissions).

Examples:

```
[MEX] A3>>SB B3:*.ASM A9:*.OBJ C6:UPDATE.DOC
[MEX] A0>>DIR A12:*.SUB
[MEX] B3>>TYPE 5:REPORT.DOC
[MEX] C3>>T B6:SESSION.LOG
```

LOGGING DRIVES: MEX supports relogging drive/user in two ways, both of which are pertinent to a discussion of DU specifications:

1) LOG command: Takes optional DU spec (trailing colon optional) resets disk system
2) At command prompt: simply enter the DU spec (colon required)

Examples:

```
    [MEX] A0>>LOG        (relogs current d/u: resets system)
    [MEX] A0>>LOG B:     (relogs current user, drive b, resets)
    [MEX] A0>>LOG B7:    (relogs drive b, user 7, resets)
    [MEX] A0>>B:         (logs current user, drv B, no reset)
    [MEX] A0>>A9:        (logs drive A, user 9, no reset).
```

Several MEX commands take string arguments. These strings must be surrounded by quotatation marks, and may contain imbedded control characters (with the singular exception of binary zero, which will generate an argument error). The caret ("^") is used to prefix a control character.

Thus, you can specify multi-line strings within one string specification. Examples (note that that the control-character ^M is the carriage return code, and ^J is a linefeed):

```
        KEY A="BILL USER;J^MWHATSNEW^MB:^MWHATSNEW^M"
        SENDOUT "SD B:*.ASM^MCRCK *.BIN^M"
```


Additionally, the slant character ('/') may be used to denote several common control-character sequences:

```
/R    specifies a newline (Carriage return code only; same as ^M)
/N    specifies carriage-return+linefeed
/T    specifies a tab
//    specifies the '/' character
/Onnn specifies any binary value (except 0):  nnn is the decimal
      value (all three digits must be present)
```

Note that the command-separator character may be included freely within a string; it separates your command strings outside of quotation marks only.

MEX supports multiple commands on a line, both in READ files, and in directly typed commands. A command line may optionally be entered on the CP/M invocation command line, and this line may also contain multiple commands. Note that a READ command will "stack" any current multiple commands; when input from the READ file is exhausted, the remaining commands will then be executed

(a control-C at any time will abort a running READ file and/or a multiple command line).

The multiple command separator in the file as distributed is the semicolon ";"). This character may be changed with the STAT command ("STAT ESC") or by modifying the patch file.

Examples:

from CP/M:

```
A>MEX DIR;READ MYFILE;SENDOUT "LOGOUT";BYE
```

This example will bring up MEX, which will first print the directory, then begin executing MYFILE.MEX, which may conain SENDOUT commands and R commands to send and receive files. When input from MYFILE.MEX is exhausted, MEX will send the string "LOGOUT" to the remote. Finally, MEX will execute the "BYE" command.

from MEX:

```
[MEX] A3>>COMMAND: RB;SENDOUT "BYE";BYE
```

This one receives a batch transmission, sends the string "BYE" (gracefully logout from a remote CP/M system, for example), then disconnects and exits to CP/M.

4) Terminal mode

MEX terminal mode is entered with any of the single-character commands T, L, or E (the differences among the three will be explained shortly).

While in terminal mode, you are communicating with the remote end as a console; your keyboard characters are transmitted to the remote, and the remote characters are displayed on your screen.

There are several functions that are available to you from terminal mode, all of which are invoked by typing the escape-character (do "STAT ESC" at command level if you don't know your escape

character) followed by one of several characters; this combination is called an "escape-sequence".

Terminal mode escape-sequences:

<ESC>-E exits back to command level
<ESC>-? prints a menu of escape sequences
<ESC>-S start copying incoming text into file buffer
<ESC>-U end (un-start) copying text into file buffer
<ESC>-P toggle copy-to-printer on/off if enabled in the overlay
<ESC>-T transmit a file to the remote (prompts for a filename)

Note that <ESC>-S and <ESC>-U require a filename argument with the T when you enter terminal mode (otherwise, there is no file active to write the incoming text into).

You can use <ESC>-P to copy incoming text to the printer. The text is held in a buffer, and prints only as the printer is ready for a character. Thus, your printer may be slower than the modem, and you won't lose characters unless the buffer fills up.

Terminal-mode files (also called CAPTURE or ASCII-SAVE files) are created by entering a filename with the T (or E or L) command; if the file already exists, you will be asked if the file should be erased. MEX will create the new file, and enter terminal mode. You may also append to an existing file by specifying the "A" secondary option (explained in more detail later).

At this point, incoming text is being saved; you can temporarily disable this by using the <ESC>-U command; use <ESC>-S to continue saving in memory (at the point where the last <ESC>-U left off). You can perform as many start/stop sequences as you like (when text-save is active, you'll see a colon at the start of each line as an indication that the save is active).

You can exit terminal mode and move freely among drives and user areas without affecting an open term-file. The LOG command (used to change drive/user and reset the disk simultaneously) will not be functional.

If you enter terminal mode with the T,E or L command witha file name, when a file is already open, the open file will be closed, and the new one opened.

You can transfer files using either Christensen or CIS protocols while a TERM file is open -- your buffer may be written to disk to make room, but nothing will be lost, and the file will still be open.

To close the file when you're done saving text, exit terminal mode with <ESC>-E, and use the WRT command. If you change your mind, and decide you don't want to keep the file after all, use the DEL command to erase the file entry and delete any text already saved in memory.

When your printer or ASCII-save buffer fills, MEX will send an X-OFF character to the remote. MEX will then wait for the remote to stop, and will save up to an additional 150 characters into an auxiliary buffer (this buffer may be increased in size by modifying MEXPATxx.OVR). When the remote stops (or the auxiliary buffer fills) MEX will write the ASCII-save buffer to disk (if ASCII-save is active) and print characters until the printer buffer is half-full (if List-copy is on). Then MEX will re-start the remote by sending an X-OFF character. However,if the MEX queue function is disabled (see the QUEUE STAT option in section 8), MEX will not save any additional characters after the X-OFF.

Note that for this scheme to work, the remote computer must respond to this X-ON/X-OFF "protocol" (most computers do, but not all!).

You can transmit a local disk file to the remote with <ESC>-T; you will be prompted for a filename. If you have the XON and and XLINE variables turned off (see section 8) MEX will ask if you want to use the character and line delays for the filesend; these delays allow slow remote computers to receive files.

The WCHAR and WLINE variables determine the time-delay values for each character and each line respectively (of course, you must have answered "Y" to the "want delays?" prompt). See section 9 for more discussion these variables.

If either of the the XLINE or XON switches is ON, the delay prompt will be skipped, and delays will not be used.

XLINE is a technique used by some mainframes: after a line is transmitted, MEX will not transmit another until receiving an X-ON character from the remote.

XON is more frequently used: when XON is active, MEX will send characters at full speed, but monitor the remote for an X-OFF character. When an X-OFF is received, MEX will pause until the remote sends an X-ON character.

The APPEND secondary option:

MEX has a special feature that allows you to maintain 'log' files

of your sessions; this is the 'A' secondary option (specified with the T, E or L primary commands when entering terminal mode IF a filename is specified).

When you specify the 'A' secondary option, MEX will search the logged area (and/or the ALT area: see the ALT command description in section 7 for more information on searching) for the file specified. If found, MEX will scan to the end of the file, and append new data on to the end, rather than querying for an erase.

If 'A' is not specified, MEX will limit its search to the currently logged DU, and, if the file is found, will ask you if it should erase the file.

In either case, if the file is NOT found, MEX will create a new file in the currently logged DU.

5) Single-character commands

The most commonly used MEX commands are implemented as single character commands, for ease of use. These are:

S: Send a file or group of files using Christensen protocol
R: Receive a file or group of files using Christensen protocol
T: Enter terminal mode (may specify an ASCII-save file argument)
E: Enter terminal mode with echo (filename argument ok)
L: Enter terminal mode with local-echo only (filename ok here)

T, E, and L are all variations of terminal mode: T sends keyboard characters to the remote and prints characters received from the remote on the console. E mode echoes received characters back to the remote ("half duplex" mode), and L echoes keyboard characters on the console before they go out to the remote ("half duplex" in the other direction).

6) Secondary commands (options)

Secondary options modify the way the single-character commands work, and are normally placed immediately after the single-character command on the command line. Following is a list of the secondary commands:

OPT	NAME	COMMANDS USED WITH	ACTION
A	Append	E,L,T	Append to a terminal-mode file
B	Batch	R and S	BATCH mode, Christensen protocol
D	Disconnect	R and S	Disconnects after a file transfer
E	Term/echo	R and S	Term-mode w/echo after transfer
L	Local	T,E	Local-character echo
Q	Quiet	R and S	Sets quiet mode for file transfer
R	Recv-view	R and S	Shows received-characters only
S	Send-view	R and S	Shows transmitted characters only
T	Terminal-mode	R and S	Enter terminal mode after xfer
V	View	R and S	Views the file transfer in ASCII
X	Exit	R and S	Disconnect and exit after xfer

Note that the options used with R and S are also effective with CIS transfers (except for the 'B' option), if set with the GLOBAL command (see the command description for "GLOBAL" in section 7).

Examples of secondary options:

```
Primary Option
|Secondary Options
|||
|||
ST   MYFILE.FOO   <<== send MYFILE.FOO, go into Term. mode after
RVT  CRACKER.JAK  <<== receive file with view, enter Term mode
RQX  MONKEY.SHN   <<== recv file quietly, disconnect/exit after
RBT               <<== recv batch files, Terminal mode after
T    DISKFILE.SAV <<== enter term. mode, allow save to disk file
                  (note that the actual copy to the file must be
                   activated with ESC-S while in terminal mode)
TA   DISKFILE.SAV <<== Enter tr area. The command
```

7) Command descriptions

The following pages detail the command set recognized by MEX. In general, the bracket characters denote optional items, and should not be entered literally. For example, the syntax descriptor

```
DIR [<du-spec>]
```

means that the du-spec is optional; hence, the command syntax may be satisfied with any of the following actual command lines:

```
DIR
DIR A3:
DIR 0:
```

Items surrouded in angle-brackets denote an item's type; the angle brackets should not be included in the actual command line, nor should the description inside the angle brackets be entered literally. In the above example, the descriptor <du-spec> is replaced with an actual drive/user specification.

We will now look at each command in detail, in roughly alphabetical order.

- The ALT command

The ALT command specifies the ALTernate drive/user area to be searched by MEX for LOAD files (.PHN and .KEY), READ files, and terminal capture files (when APPEND is on). Syntax is

```
ALT <du-spec>
```

Examples:

```
ALT A0:
ALT B3:
```

- The BELL command

The BELL command rings the bell on your terminal (regardless of the setting of the STAT BELL, which normally disables the bell). BELL takes a single argument, which specifies the number of times to ring the bell. BELL is intended for use in READ files, as a means of summoning the operator when a certain point has been reached in the READ file's execution).

- The BYE command

BYE is used to return to CP/M (does not disconnect) at the end of a MEX session.

- The CALL command

The CALL command is used to dial numbers either from the library or from the keyboard. If you specify more than one number on the command line, CALL will try each until one answers with a carrier tone. For example,

```
CALL 16165559033 SENACA
```

will dial the first number; if busy (or no answer), CALL will try the second. The special number '#' will cause the entire command line to be repeated if none of the dialed numbers responds with a carrier:

```
CAL WESTWOOD 1-616-555-2040 TCBBS #
```

will try all three numbers repeatedly until one answers. You may optionally place a limit on the number of retries by including a number after the '#':

```
CAL 555-1212 #48
```

A ^C will abort dialing at any time.

Phone-library numbers may specify an optional baud rate (see the description for the PHONE command for information on how to specify the rate). If a baud-rate is present, CALL will change the baud rate before dialing the number.

Note that, for this feature to work, your hardware must be capable of changing the rate AND your overlay must implement the NEWBD vector. Most MEX overlays (overlays whose names start with "MXO-") allow this, if the hardware is capable of baud-rate change; most MDM overlays do NOT.

- The CLONE command

The CLONE command allows you to save a new copy of MEX, with the current options intact (including any defined keystrings and phone numbers). The syntax is:

```
CLONE <filename>
```

Examples:

```
CLONE MEX.COM
CLONE NEWMEX.COM
```

Clone will prompt you for an erase-file operation if the specified file already exists on the disk.

- The CLS command

CLS clears the screen on your terminal (if supported by your physical modem overlay). Handy when a burst of noise leaves the cursor atop a screenful of garbage, but requires clear-screen support in the physical overlay.

- The COLD command

The COLD command re-starts MEX and erases any defined keystrings along with the phone library. This is the only way to erase the entire phone library with one command. The COLD command also allows you to remove the CIS protocol module (for those who don't need it) ... do this:

```
STAT CIS OFF
COLD
CLONE <filename>
```

Note that this removes the CIS module altogether (you can't get it back with the STAT CIS command). This frees up roughly 1K of memory.

- The CPM command

CPM is a synonym for BYE, and returns control to the operating system without disconnecting.

- The DEL command

DEL is used to close and erase an open terminal file; use this command when you change your mind about saving a terminal file.

- The DIAL command

The DIAL command performs the same task as the CALL command, except that after calling, DIAL returns to command mode (CALL

goes to terminal mode).

The syntax for DIAL is exactly the same as the syntax for CALL (described previously), including the multiple-number feature and the repeat option.

DIAL is intended to be used in READ files, where it is desirable for the READ file to retain control after calling a number (READ files have no effect in terminal mode; thus using CALL in a READ file would pause execution of the read file when the distant end is reached).

- The DIR command

DIR works similarly to the CP/M DIR command, and displays the disk directory on the screen. DIR takes advantage of the DU specification, thus "DIR B7:" will display all of the files in user 7 on drive B. System files will be excluded if the STAT variable 'EXCLUDE' is set to ON. If EXCLUDE is OFF, all files will be displayed.

More examples:

```
DIR C3:*.?Q?
DIR LOGIN.MEX
DIR 3:
```

- The DSC command

DSC disconnects the modem from the phone line (may not be implemented in all overlays).

- The ERA command

ERA erases files, similarly to CP/M's ERA. The syntax is as follows:

ERA <filename> [V]

The filename may be ambiguous (e.g., *.ASM, BOOT?.BAK). ERA always displays a list of the files that are being erased. If you specify the optional 'V' after the filename, ERA will ask for verification before actually erasing the files (after displaying

the names).

- The GLOBAL command

The GLOBAL command allows you to set the secondary options (described previously) for the single character commands. Any options set this way will then be active for the single-character commands whether or not they are specified in the actual command line. For example, if you prefer to go directly to terminal mode after a file transfer, do:

GLOBAL T

You can also set the VIEW mode for CIS transfers by doing

GLOBAL V

(note that this is the only way to "view" a CIS file transfer.

- The EXIT command

EXIT is a synonym for BYE, and returns control to the operating system without disconnecting.

- The HELP command

HELP is used to access the on-line manual for MEX (contained in the file HELP.MEX). Syntax for the HELP command is as follows:

```
HELP                    <<--- prints help for the HELP command
HELP ?                  <<--- lists available topics
HELP <topicname>        <<--- prints help information for a topic
```

HELP.MEX must reside in the currently logged drive/user area (unless you use an ALT area, with the SEARCH variable). Once the help file is opened, you may move freely among drive and user areas without affecting the operation of HELP; MEX remembers where the HELP file is located.

Once opened, the HELP file will remain open for the duration of the session, unless a LOG command is executed.

- The ID command

The ID command allows you to configure the MEX ID string (which is printed in the prompt, and in error messages, if the STAT ERRID switch is set to ON). This is useful if you're communicating with another computer running MEX; if each computer has a different ID string, you're never in doubt as to which one you're typing commands to, or which one printed an error message.

The syntax for the ID command is

ID <string> (see "STRINGS")

The string may be a multi-line string; it's length in the standard distributed MEX may not exceed 28 characters. This length may be different if your system overlay replaces the MEX defaults (in fact, the overlay may disable this command altogether; if the "INVALID COMMAND" message appears when you attempt to use the ID command, then this is probably the case).

-The KEY command

The KEY command is used to manipulate the keystrings that are available in terminal mode. There are several forms of the command:

```
KEY                          <<= prints out all of the keystrings
KEY <keyname>=[<string>]     <<= defines a new keystring
```


Keyname is any valid ASCII character, except for the set defined as the terminal mode escape-sequence commands (see TERMINAL MODE).

In the second form, above, if <string> is omitted, then the specified keystring is erased.

Examples:

```
KEY %="ATDT 14145559932"        <<= sets up the '%' key
KEY Q=                          <<= erases the Q keystring
```

Note that KEY names that duplicate terminal-mode commands will be rejected by the KEY command, and an error message will be printed.

- The LOAD and SAVE commands

LOAD and SAVE are used to load and save phone number and key string files. Both commands take a filename argument. Examples:

```
LOAD A5:CBBSPHON.PHN          <<== loads a phone # file
LOAD ARPA.KEY                 <<== loads a keystring file
SAVE C:NEWPHONE.PHN           <<== saves phone # file
SAVE 9:COMPUSRV.KEY           <<== saves keystring file
```

Note that the filetype determines what type of file is being saved (.PHN for phone number files, .KEY for keystring files). Any other filetype will generate an argument error.

- The LOG command

The LOG command allows you to reset the drive (for switching disks) and, optionally, simultaneously change drive and/or user area. The command syntax is

LOG <DU-SPEC>

where DU-SPEC is either or both of the new drive to log into and the new user area. An optional colon may follow the DU-SPEC. Examples:

```
LOG B7:          <<= reset, log in drive B user 7
LOG 7:           <<= reset, log user 7 current drive
LOG B:           <<= reset, log drive B current user
LOG              <<= reset, retain current user/drive
```

If either a terminal file or a READ file is open, the disk-reset will be denied and the LOG command will display an error message.

If you want to change drive and/or user without a disk reset, you can do so without the LOG command: just enter the DU spec at the MEX command prompt.

- The PHONE command

The PHONE command may be used to query the phone number library, as well as for adding and deleting numbers. To add a number, use this form:

PHONE <id>=<number> [baud-rate]

Where ID is a string (up to 8 characters) that you want to use to call out the number and <number> is the telephone number. If <id> already exists, it will be replaced. <Baud-rate> is an optional rate to be associated with the number, used by the CALL command. To remove a number, do

PHONE <ID>=

To see the entire library, do

```
PHONE
```

To see a single entry, do

```
PHONE <ID>
```

Examples:

```
PHONE FONE=1-414-563-4013 1200 <<--- associates number w/FONE
PHONE FONE=1-414-563-4013      <<--- same but no baudrate chg
PHONE                          <<--- lists the phone number lib
PHONE FONE=                    <<--- removes FORTFONE from lib
PHONE FONE                     <<--- prints entry for FORTFONE
```

- The PREFIX and SUFFIX commands

PREFIX and SUFFIX are used with the SENDOUT command (described later). PREFIX is used to manipulate a special PREFIX string, and SUFFIX is used to manipulate a SUFFIX string.

The PREFIX string is transmitted ahead of any SENDOUT string, when the SENDOUT command is used, and the SUFFIX string is transmitted after. This simplifies the construction of complex, repetitive command lines to be sent to the remote.

Either command without arguments prints the current value of the string. To change the string, enter the string on the command line after the command.

Examples:

```
PREFIX ""               <<== sets null prefix string
PREFIX "XMODEM S "      <<== sets a prefix
SUFFIX "^M"             <<== set carriage-return as suffix
```


In the second and third examples above, the subsequent SENDOUT would work like this:

```
SENDOUT FOO.BAR
```

which would actually send out "XMODEM S FOO.BAR" <CR>

- The READ command

The READ command causes MEX to take command lines from a disk file. This can be handy for such things as controlling a set of file transfers (especially when you can't be present for the entire session) and executing complicated login sequences automatically. READ, used with the EXTEND function (see the description of the EXTEND variable in section 8) provides a means of extending MEX's command set.

The syntax for READ is:

```
READ <filename> [<parm1>] [<parm2>] ...
                       \          \
                        \ _________\_____>> optional!
```

The READ file may contain any valid command EXCEPT another READ command. These commands may freely move among drives and user areas; MEX will remember where the READ file is located.

Typically, READ files are created with a text editor, and may contain STAT commands to set MEX for a particular type of connection, DIAL commands to actually make the connection, and SENDOUT commands to log in at the destination. It's possible for an entire session to take place under a READ command.

A READ file will terminate when the file ends, or when a CONTROL-C is seen at the console (aborting any command with CONTROL-C will abort the READ file).

It's important to note that, while you can use a READ file to enter the terminal mode, the READ file is not used while the terminal mode is active. When you exit terminal mode, the READ file starts again.

Normally, READ commands echo on the screen when they execute. You can inhibit this, however, by manipulating the SILENT variable (do STAT SILENT ON to inhibit the command echo).

READ parameters are very similar to CPM's SUBMIT parameters; they are called ACTUAL parameters. If they are present in the command line, they will be plugged into the FORMAL parameters in the submit file. This substitution facility provides a powerful tool for generating multiple-purpose command files.

Formal parameters occur in the READ file, and take the form {n}, where n is the parameter number; these numbers correspond sequen-

tially with the ACTUAL parameters entered on the command line. The ACTUAL parameters in the command line are substituted for the FORMAL parameters in the READ file when the file is executed. A special form of FORMAL parameter allows a default value to be used if an ACTUAL parameter is not specified on the command line; this form is {n:<text>} where n is the parameter number, and <text> is any arbitrary text to be used as the default.

If the default form of the formal parameter is not used, and no actual parameter is specified in the command line, execution will continue, but the parameter will be blank.

Normally, the parameters in the READ command line are terminated by the space between parameters (or the end-of-line); spaces can be imbedded in the ACTUAL parameter by enclosing it in braces.

I know this all sounds quite complex, but it's really quite simple, as the following example will show.

Assume a file named FILEGET.MEX contains the following lines:

```
SENDOUT XMODEM S{2} {1:SENACA.DQR}
R{2} {1:SENECA.DQR}
```

({2} and {1:SENACA.DQR} are FORMAL parameters). This file can be used in several ways:

```
READ FILEGET            will transfer SENACA.DQR to your system
READ FILEGET MEX.UPD    will transfer MEX.UPD to your system
READ FILEGET *.NEW B    will transfer all NEW files in batch mode
READ FILEGET USQ.DOC VT will transfer FILEGET, viewed,then T mode
```

The following line illustrates how to expand a parameter, using the above file, and is useful only if MEX is running on the remote end:

```
        READ FILEGET {*.BOO NEWSTUFF.DOC} BX
```

(Transfers all .BOO files and NEWSTUFF.DOC) in batch mode, then disconnects)

If you turn the command extender on (STAT EXTEND ON), the READ in the above lines can be omitted, making the READ file look like a built-in command. This affects single-character commands somewhat: the disk is searched before the the command is checked for a single; you can avoid this by prefixing single character command lines with a '*' (eg, *RQ FILE.FOO).

- The REN command

The REN command syntax is similar to the CP/M's REN, except that DU specifications may be employed in either the new OR the old filename (but not in both).

In general the syntax is

REN <newfilename>=<oldfilename>

Examples:

```
[MEX] A0>> REN NEWFILE=OLDFILE          (uses A0)
[MEX] C9>> REN B6:REPORT.OLD=REPORT.DOC (uses B6)
[MEX] B4>> REN 6:MYFILE=B:YOURFILE      (uses B6)
[MEX] D4>> REN ERROR=C9:BLUNDER         (uses C9)
```

Note that the following REN commands will produce errors:

```
[MEX] A0>> REN B3:SOMETHING=C:ELSE      (ambig drives)
[MEX] A0>> REN 6:ME=7:YOU               (ambig user #'s)
```

Also, ERA will prompt for erasure if the new name already exists. Before doing this, it checks for the existence of the old filename, and, if not found, aborts before checking for the presence of the new name.

- The SENDOUT command

SENDOUT allows you to send an arbitrary string out to the modem (see STRINGS). This is most useful in READ command files, but can often be useful in normal interactive mode.

The syntax for SENDOUT is

SENDOUT <string>

Before the specified string is transmitted, a PREFIX string, if any, is transmitted, followed by the string specified to SENDOUT, and terminated by a SUFFIX string (if any; the default suffix string is a return-code, normally terminating the line to the remote). After transmitting the string, MEX will wait for a reply from the remote, up to a pre-set number of seconds; any reply will printed on your screen.

Following are settings and variables which affect the SENDOUT command:

SUFFIX: \ Described previously
PREFIX: /

WTECHO: Manipulated by the STAT command, this switch-variable determines whether or not SENDOUT validates its transmission by waiting for characters to be echoed from the remote. If set to ON, then SENDOUT checks each printing character it transmits with the character as echoed by the remote. If a mismatch occurs, SENDOUT marks an error, sends a CANCEL character, and awaits a TRIGGER character from the remote. It then begins again, and repeats this cycle until either the transmission occurs without error, or the error count is exceeded (which aborts SENDOUT).

If you use SENDOUT with WTECHO off, you'll also likely want to set TRIGGER to null (STAT TRIGGER ""), to avoid trigger-wait.

More factors affecting SENDOUT:

RETRY: Manipulated by the STAT command, this value-variable specifies the error-retry count for SENDOUT before aborting.

CANCHR: Manipulated by the STAT command, this string-variable (single character) specifies the character to be transmitted to the remote to cancel the transmitted line after an error.

SODELAY: Manipulated by the STAT command, this switch-variable ties SENDOUT strings (and terminal-mode keystrings) to the WCHAR and WLINE time-delay values, as long as WTECHO is off (if WTECHO is on, it takes precedence, and delays are not used).

TRIGGER: Manipulated by the STAT command, this string-variable (single character) is the character the SENDOUT command waits for after an error-cancel, before proceeding with the retry. If the other end doesn't echo, set TRIGGER to 0 (and WTECHO to OFF), and MEX will simply send its arguments and not worry about validating the echoed string (this is the technique you should use to send Smartmodem command strings, for example).

- The SET command

SET is a command defined entirely by the overlay; if your overlay doesn't implement the SET command, MEX will print the "invalid command" message when you try to invoke SET.

- The SSET command

SSET is similar to SET, but is included for use of the Smartmodem logical overlay. Not all Smartmodem overlays implement set, but those that do aren't guaranteed to use identical syntax.

- The SLEEP command

SLEEP is used to invoke a delay (handy sometimes in READ files). Syntax is:

SLEEP n

where n is the number of seconds to wait (a ^C from the console will abort the SLEEP command, as well as any active READ file).

SLEEP, used with the STAT CLOCK function, can be used to "tweak" the timing constants in MEX (by timing the actual SLEEP period, and adjusting the CLOCK value until the SLEEP argument agrees with the actual measured time). This is especially useful with multi-tasking operating systems, such as MP/M and TurboDOS, where background processing wreaks havoc with the internal MEX timing constants.

- The STAT command

The STAT command lets you examine certain system variables, and change certain others. In general, the syntax is as follows:

```
STAT <KEYWORD>                <<== to examine a variable
STAT <KEYWORD> <NEW-VALUE>    <<== to change a variable
```

The NEW-VALUE will depend on the context of the keyword in question. For example, most of the switch-type variables are either ON or OFF, hence,

```
STAT BAKFIL                   <<== prints out ON or OFF
STAT BAKFIL ON                <<== turns on BAKFIL
STAT BAKFIL OFF               <<== turns off BAKFIL
```

The value-type variables, on the other hand, will print the number or text associated with that variable. Setting the value requires the entry in units associated with that value. For example,

```
STAT REPLY 5     <== sets 5 seconds as SENDOUT reply time
STAT WCHAR 4     <== set 40 ms as transmit-character delay time
                     during a terminal-mode file send
```

Some variables are read-only, most notably STAT BUFFER (which prints out the save-buffer and printer-buffer statistics).

To obtain a list of all of the options that can be viewed or changed with the STAT command, do

```
STAT ?
```

Most of the STAT keywords are documented in section 8.

- The TERM command

The TERM command is used to activate a terminal-mode file, and is generally of use within READ files. TERM works exactly like the single-character T command, but does not actually enter terminal mode. TERM takes a filename as an argument.

Some examples:

```
TERM SESSION.LOG
TERM ROYALOAK.DIR
```

- The TERMA command

TERMA works exactly like TERM, but opens the file in append mode. Thus it is similar to the single-character T command, with the A secondary option ("TA").

- The TYPE command

The TYPE command prints files on the console, similarly to the CP/M TYPE command, but with pagination: The syntax is:

```
TYPE <filename> <'P'>
```

The optional 'P' will affect pagination at screen boundaries: if pagination is OFF ("STAT PAGE OFF") then 'P' will cause the file to be paged. If pagination is ON ("STAT PAGE ON"), then 'P' will inhibit pagination.

Examples:

```
A2>>COMMAND: TYPE REPORT.DOC    (paged output if PAGE ON)
A2>>COMMAND: TYPE REPORT.DOC P  (inhibits paging if PAGE ON)
```

- The SYSTEM command

SYSTEM is a synonym for BYE, and returns control to the operating system without disconnecting the modem.

- The WRT command

WRT is used to close and save an open terminal file. An automatic WRT if performed for you if you use any of the exit commands (BYE, EXIT,CPM, or SYSTEM) while a terminal-mode file is open.

8) STAT variables

This section describes the variables that may be examined or changed with the STAT command.

- The ALERT variable

ALERT is a STAT value variable; it determines the number of times the console bell will ring when a remote computer is reached with the CALL and DIAL commands. ALERT works only on calls that have taken more than one dialing attempt to reach.

ALERT is handy for dialing with the repeat option (see CALL command in section 7 for more information about the repeat option): you can enter a number (or group of numbers) to be dialed; with ALERT set non-zero, you'll have an audible signal that a call has been completed.

- The BAKFILE variable

BAKFILE is a STAT Switch variable: if ON, any command that creates a file when one of the same name already exists will, instead of erasing the old (or prompting for an erasure) rename the old with the same primary name, and a secondary name of "BAK".

- The BELL variable

BELL enables or disables the console bell throughout MEX (ie, in

terminal mode, command mode, and during file transfers).

- The BUFFER variable

BUFFER is not really a variable; use STAT BUFFER to print out the ASCII capture buffer statistics (size, amount used, amount available). In addition, STAT BUFFER displays the batch-filenames buffer size (which determines the largest number of files that may be transferred in one batch file transfer command).

- The CAPTURE variable

CAPTURE is a switch variable that may be used to enable or disable save-in-memory when a term file is open (it will refuse to change if no term file is open). CAPTURE is most useful within a READ file to allow SENDOUT replies to be saved to the capture buffer.

- The TRIGGER and CANCEL variables

TRIGGER is a STAT CHARacter variable, and specifies the character the SENDOUT command will look for before sending out its argument (a single-character prompt from the remote). To disable the trigger-wait function altogether, do

```
STAT TRIGGER ""
```

CANCEL is anoter CHAR variable, and specifies the character the SENDOUT command will send to the remote to cancel the line after an error.

- The CHAR variable

CHAR is not really a varible, but a command option to STAT that prints a list of all of the CHAR variables.

- The CHECKSUM and CRC variables

CRC and CHECKSUM are switch variables; the two are mutually exclusive (that is, turning one on turns the other off) and set the preferred type of error checking in Christensen file

transfers.

CHECKSUM is a simple sum of the outgoing record; CRC uses a more sophisticated technique employing polynomial arithmetic, and is thus a better method (more errors are detected, thus, there is less chance of an incorrect record being received as if it were valid).

The original MODEM2 protocol employed only CHECKSUM detection; however, in the past few years, most versions of Christensen exchange programs have been rewritten to accept the CRC technique as well. MEX will adapt to transmitting programs using either type of validation.

If MEX is the transmitter, it will switch modes after several failed attempts to transmit a file, thus allowing full compatibility with older modem programs.

- The CIS variable

CIS is a STAT switch: when ON, then Compuserve protcol file transfers are allowed while in terminal mode. If OFF, Compuserve transfer sequences from the remote end are ignored altogether (although they are printed on the screen as "normal" terminal-mode characters).

CIS transfers always take place while in terminal mode; there is no command to send or receive a file using CIS protocol, since the remote Compuserve end must initiate the transfer.

Note that if you use the COLD command to restart MEX while the CIS switch is set to OFF, then the Compuserve file transfer module will be removed altogether from the running copy of MEX. If you subsequently use the CLONE command to create a new executable MEX.COM, the additional space formerly used by the CIS module (about 1000 bytes) will be available instead for your buffers. You will not be able to use STAT to turn the CIS switch back on (since there is no longer a CIS module in the system). This feature is intended for those users who do not need the ability to do Compuserve-protocol transfers, and would rather not waste space on an unneeded feature.

- The CLOCK variable

CLOCK is a STAT value variable; you may set this variable to any

value between 1 and 255. The value is the CPU clock speed in tens of Megahertz (hence, the speed may vary between .1 and 25.5 Mhz). You can then use the CLONE command to make this change permanent.

CLOCK is useful for "tweaking" MEX under varying conditions of load when running under multi-tasking operating systems, such as MP/M and TurboDOS. When system load is heavy, you can decrease the CLOCK value, causing MEX to spend less time in its internal timing loops.

- The DEBUG variable

DEBUG is a STAT switch that affects the Terminal mode in MEX. If DEBUG is ON, then characters received in Terminal mode are displayed in a form similar to CP/M's DDT dump (D) command: hex values on the left side of the screen and the ASCII block (as the output progresses) on the right.

You'll note that the ASCII right-side display is "buffered" until 16 characters have been received. After the 16th, the ASCII right-side is printed. Therefore, if the remote's output pauses, you'll see only the hex values until the remote sends more output. Also, exiting Terminal mode will display any buffered ASCII.

DEBUG is useful whenever it's necessary to know exactly what the remote is sending (for example, if you're connected to a time-share network, and can't transfer files, a DEBUG session can prove illuminating in determining just what is happening).

In DEBUG mode, the FILTER flag is ignored, allowing all characters to reach the screen. Additionally, the CIS protocol is disabled while DEBUG is ON.

- The HEX and DECIMAL variables

HEX and DECIMAL are STAT switch variables; they are mutually exclusive (ie, turning one ON turns the other OFF. These variables specify the default input radix of commands that take numeric arguments (e.g., SLEEP, STAT VAL <#>, the '#' spec in CALL commands, etc). If HEX mode is ON, then these numbers are considered Hexadecimal; if DECIMAL mode is on (MEX is distributed with

DECIMAL ON), then they are considered decimal values.

You can force a decimal number, regardless of the HEX or DECIMAL mode, by preceeding the number with a '$' character; similarly, the '#' character implies a decimal number.

HEX mode has a side effect: it turns on HEX record count report ing in file transfers.

- The ERRID variable

ERRID is a STAT switch that enables and disables the printing of the MEX ID string in error messages.

The ID string is useful in applications where you're communicating with a remote computer running MEX (if each end has a different ID code, there is no problem confusing error messages), but if you typically use MEX only with RCPM systems or timesharing systems that don't run MEX, you'll likely want to turn off the ERRID switch, and shorten the error messages.

- The EXCLUDE variable

EXCLUDE is a STAT switch-type variable. It modifies the action of commands that take multiple-filename arguments (such as DIR and the batch-transmit SB).

If EXCLUDE is ON, then SYS files are skipped over by these commands.

If EXCLUDE is OFF, then SYS files are included.

Thus, you can exclude SYS files from a batchfile transmission by STATting the EXCLUDE switch ON. These files will also be exclude from DIR listings while EXCLUDE is ON.

- The EXTEND variable

EXTEND is a STAT switch variable. When turned on, it modifies the way MEX's command decoding works: if a command is not found in MEX's command table, MEX will pass the entire command line on to READ, causing the command to look like a READ command (without READ being present on the command line).

When using EXTEND, you may notice that the single-character

commands are searched as disk files before being checked as built-in commands; this is due to parsing constraints. To avoid the disk search when using single-character commands with EXTEND active, prefix the command withe a '*' (eg, *RQ FILE.FOO).

- The ESC variable

ESC is a STAT CHAR variable, and specifies your terminal-mode escape character, used to activate terminal-mode functions (such as capture and printer buffer on/off, etc). The argument is a single character string. Example:

```
STAT ESC "^I"
```

changes your escape character to the TAB key.

- The FILTER variable

FILTER is a STAT switch variable; when ON, then abnormal control characters are ignored when in terminal mode (with the exception of carriage-return, linefeed, backspace and tab). If OFF, then all characters from the remote will be displayed on the screen (and included in the ASCII-capture file, if active). FILTER also screens nulls and characters with the high-bit set.

FILTER ON is useful if you're working with a noisy connection, and random garbage characters on the line are erasing or mangling your screen display.

If you're using a video-oriented text editor at the remote, you'll want to be sure FILTER is set to OFF, in order to allow the video-control characters to be accepted by MEX.

Note that the CP/M end-of-file mark is specifically excluded from ASCII-capture files regardless of the setting of FILTER (although FILTER ON allows this character to go to the screen).

- The INITFILE variable

INITFILE is a STAT switch variable. It determines whether or not MEX will look for the start-up file INI.MEX when it is first started.

INITFILE is intended to be used prior to running CLONE, to prevent the cloned MEX from looking for INI.MEX.

For more information about the CLONE command, see its command description in section 7.

- The LF variable

LF is a switch variable: if ON, it affects terminal-mode file transmission by adding a linefeed after every carriage return.

- The LIST variable

LIST is a Switch variable, and enables or disables the LIST device.

- The PAGE variable

PAGE is a STAT value variable, and sets the number of lines to be displayed on the screen, for commands like TYPE, HELP and STATUS, which paginate their output. Setting the PAGE variable to 0 disables pagination by these commands.

- The PRECHO variable

PRECHO is a STAT switch variable that enables and disables the logging of incoming characters to the printer. It is similar to the <ESCAPE>-P printer toggle command used in terminal mode, and is intended for use in READ files.

- The REPLY variable

REPLY is a STAT value variable, and specifies the amount of time (in seconds) that the SENDOUT command will wait, after sending a string, for the remote end to send a reply. Any such reply will be displayed on your terminal screen; if a terminal-file is active and CAPTURE is on, then the reply also goes to the term-file. If PRECHO is active, the reply will go the list device as well.

To query the REPLY constant, do: STAT REPLY

To set the REPLY time, do: STAT REPLY <value>

where <value> is the number of seconds to wait, between 0 and 255.

- The RETRY variable

RETRY is a STAT value variable, and specifies the number of times the SENDOUT command will attempt to send a string before aborting. Note that if the Wait-For-Echo switch (WTECHO) is off, RETRY will have no effect, since no error can occur.

To query the RETRY constant, do: STAT RETRY

To set the RETRY constant, do: STAT RETRY <value>

where <value> is the number of retries, between 1 and 255.

- The RTIME varible

RTIME is a STAT value variable, and affects Christensen-protocol file transfers.

RTIME is the amount of time (in seconds) that MEX will wait for a character from the remote end before declaring a timeout, and initiating a record-retransmission sequence.

RTIME is set for one second in the standard distribution MEX; this is more than adequate for RCPM work, and MEX-to-MEX file transfers where each end is running under a single user operating system. But when working with a mainframe-type connection (and sometimes when a remote MEX is running under MP/M), especially when a packet-switched network is part of the connection, character delays can easily exceed one second. In these environments, you'll want to set RTIME to a higher value (16 seconds is recommended; the maximum is 255).

- The RUB variable

RUB is a STAT switch variable and affects terminal mode: if ON, then the local backspace key is converted to RUB when the character is transmitted to the remote.

- The SEARCH varible

SEARCH is a STAT variable that may take on the values 0, 1, 2 or 3. SEARCH specifies a search path for READ files, phone libraries, keystring files, and (if the append secondary option, 'A' is specified) terminal-mode ASCII capture files. These files normally are taken only from the currently logged DU; by modifying SEARCH, you can change the way MEX finds these files.

Two drive/user (DU) areas are possible: the currently logged DU and the Alternate DU (see section 7 for information on changing the alternate DU).

If SEARCH=0, then only the currently logged DU is searched.
If SEARCH=1, then only the ALT DU is searched.
If SEARCH=2, then the currently logged is searched; if the file is not found, then the ALT DU is searched.
If SEARCH=3, then the ALT DU is searched; if the file is not found, then the currently logged DU is searched.

- The SODELAY variable

SODELAY is a switch variable; when ON, SENDOUT arguments and keystrings sent from terminal mode are tied to the WCHAR and WLINE delay variables normally used to transmit a file in terminal mode. This provides a conventient means of transmitting passwords, etc. to the remote from within a READ file (you don't have to force MEX to wait for an echo that never comes, yet you don't run the risk of overrunning the remote input buffer).

Note that, for SODELAY to have any effect, the WTECHO switch variable must be OFF (WTECHO takes preference if both are active at the same time).

- The SPLIT variable

SPLIT is a STAT switch variable that affects the screen display of the phone library (done from the PHONE command or the CALL command).

This variable is set ON in the distribution version of MEX, but may be turned OFF if you're using a terminal with an extra-wide display, to effect a more compact phone library display using two entries per screen line. This compact display does not, however, provide a number's optional baud-rate (as does the display when SPLIT set to ON).

- The SILENT variable

SILENT is a STAT switch variable, and, when ON, inhibits the echo of command lines in READ files.

- The SWITCH variable

SWITCH is not really a varible; it requests the STAT command to list out all of the switch-type variable.

- The TAB variable

TAB is a switch variable that can be used to enable or disable the internal tab expander within MEX; this is sometimes necessary when using certain terminals with remote screen editors.

Note that TAB is set to ON in the distribution version of MEX.

- The VAL variable

VAL in not really a variable; it requests the STAT command to list out all of the value-type variables.

- The WCHAR and WLINE variables

WCHAR and WLINE are value variables; they specify delay times in simple file transmissions done within terminal mode (with the <ESC>- T command). When you transfer a file, MEX will ask you whether you want to use these delays.

WCHAR is the delay to use between characters.
(0-9, in 10's of milliseconds)
WLINE is the delay to use between lines
(0-9, in 100's of milliseconds)

WCHAR and WLINE delays are often needed for timesharing main frames and BBS's written in BASIC, to allow the slower remote end to catch the entire file. These variables are also using in keystrings and SENDOUT arguments when SODELAY is ON and WTECHO is OFF.

- The WECHO variable

WECHO is a value variable that affects the use of the SENDOUT command. WECHO is the time, in seconds, that SENDOUT will wait between characters during a reply from the remote, before considering the reply ended. This should not be confused with the REPLY variable, which is the maximum amount of time to wait, after sending a string, before considering that no reply is forthcoming.

- The WTECHO variable

WTECHO is a STAT switch variable, and determines whether or not keystrings and SENDOUT command lines transmitted to the remote

are validated by comparison with their echoed characters.

When WTECHO is ON, all printing characters transmitted to the remote must echo correctly back from the host; this effectively halves the transmission rate, but provides error detection feedback for the SENDOUT command, allowing it to cancel a line and retry.

If you're sending command strings out to a line that doesn't echo the characters, you'll want to set WTECHO off. Additionally, if you're using the SENDOUT facility, you'll want to also remove the trigger string (you can do this with STAT TRIGGER "", which effectively removes the trigger character.

- The SEP variable

SEP is a STAT character variable, and allows you to change the character used to separate commands when using multiple commands on a line. The argument is a single character string. Example:

```
STAT SEP "$"
```

changes your command separator to the "$" character.

- The XLINE variable

XLINE is a STAT switch that affects terminal-mode file transfers. If XLINE is set to ON, then the file will be transmitted, one line at a time; each succeeding line will not be transmitted until an X-ON character is received from the remote.

XLINE is used with a very fewe timeshare and network computers, almost never with RCPM and BBS systems.

- The XON variable

XON is a STAT switch variable that affects the terminal-mode file send (see TERMINAL MODE for a full description of terminal-mode file send).

When XON is set to ON, MEX will send all characters to the remote at full speed, but will monitor for an X-OFF character from the remote. When MEX sees the X-OFF, it will pause, allowing the remote to read all of the input MEX has transmitted. When MEX

sees an X-ON character from the remote, it will resume the file transmission.

9) About the source code (and other stuff).

Some users will note that MEX is distributed without source code; this is not an oversight. MEX has required a very substantial investment in development time, and I've become very possessive of the program (especially in light of what's happened with so many other programs appearing in source form on various RCPM's and through user groups -- they tend to get modified to death by people who do not properly consider the effects of their changes). For this reason, I will not be releasing source code for MEX in any form, beyond the source for the overlay files.

I do intend to maintain the program as responsibly as is possible for any non-funded project. In addition, a number of enhancements are planned for this fall, to culminate in a 2.0 release that will offer significant enhancements (including a simpler overlay structure, a far-more powerful READ command processor, including nested reads and conditional execution, smart-terminal emulation, and the capability of MEX to act as a remote terminal server, similar to the public domain BYE program). It's not likely that MEX 2.0 will be distributed without charge (as MEX 1.0 is); you can, however, be assured that it will be one of the cheapest terminal programs available for CP/M and as well supported as any commercial product available.

10) Support

MEX has been beta-tested among a small group over the last few months; however, the nature of software development implies the existance of a microscopic fissure through which program bugs will invariably ooze, escaping detection by all but the omnipotent (I am, sadly, not among that group).

So if you detect errors or bugs in MEX, I would like to know about it. You can reach me through the following avenues:

Arpanet:	Send mail to RFOWLER@SIMTEL20.ARPA
RCPM:	Fort Fone File Folder, Fort Atkinson, WI (owned and operated by Al Jewer). (414) 563-9932, running a Compupro 40 MB hard disk and Frank Wancho's RBBS4 message system.
US Mail:	My mailing address is Route 1, Box 7, Fort Atkinson, Wi.; be aware that I am very slothful about answering non-electronic mail (ie, you may never get an answer).
Phone:	None. Please do not call me at home; this raises hell with my family, and I'm not normally able to respond in "real-time" in any case.

If you report a bug, please be as explicit as possible, detailing any unusual overlay configurations, STAT variables, and any other conditions you feel are pertinent. Bug reports like

"Terminal mode doesn't work right"

are totally useless; all I can do is ignore such reports.

Feature requests for future releases of MEX are welcomed.

11) Credits

I'd like to thank the following individuals for their participation in the development of MEX, which included many suggestions for features, beta site testing and helpful feedback (and, in the cases of Bob Plouff and Frank Wancho, code examples from their own communicatons programs):

Bob Plouffe	Dick Mead	Keith Petersen	Al Jewer
Frank Wancho	Sigi Kluger	David Sternlight	Shawn Everson
Dave Kozinn	Charlie Strom	Eric Stork	

Special thanks must go here also to Ward Christensen, who wrote the original MODEM program from which all others have descended, Mark Zeiger, who developed the batch-file transfer protocol extension to Ward's original MODEM2 protocol, Irv Hoff, whose work with MDM7 provided a foundation for a common overlay format between MEX and MDM7, and the dozens of others who have contributed to the development of the many versions of MODEM2, MODEM7 and MDM (from which MEX began with a healthy advance along the

learning curve).

11) Distribution and a warning to illicit profit-takers

Legitimate commercial interests interested in custom versions of MEX, for distribution as a for-profit product, should contact the author for rates, royalty information and sample contracts.

Ronald G. Fowler
Fort Atkinson, WI 53538
August 20, 1984

Appendix 1: Buffer Modification

This section explains the buffering scheme used in MEX, and how you can change it using the MEXPAT overlay file.

MEX employs a number of dynamically-allocated buffers for such things as terminal-file storage, printer buffer, keystrings,

modem pre-read queue, and the phone library. The size of these buffers are defined in a fixed area of MEX; you can change their values by editing and assembling MEXPATxx.ASM, then using MLOAD to bind in the new values:

```
ASM MEXPAT11
  MLOAD MEXNEW.COM=MEX.COM,MEXPAT10
```

The variable labeled PSIZE in the MEXPAT file defines the number of Kbytes to use for the printer buffer. This can be set as low as 0, or as arbitrarily large as you'd like. The variable labeled ASIZE performs a similar function for the terminal-mode file capture buffer, but must be set to 2 or greater, to reserve the minimum 2K space for file transfers.

One of the two (and only one) must have the value 255 (0FFH); this defines a particular buffer as the "top" buffer ... ie, the buffer allocated after all other space has been allocated, and extends to the top of the TPA. This is usually the largest buffer, and I recommend that this be the terminal-mode capture buffer, since that is the most frequently used buffer.

The PHSIZE label defines the maximum size of the phone library, in entries. Currently an entry is 43 bytes long, so the "standard" value for PHSIZE of 30 results in a phone library consuming almost 1300 bytes.

Similarly, the KYSIZE variable defines the amount of space (in bytes) reserved for terminal-modem keystrings. The "standard" value is 400 bytes.

If you don't use phone libraries or keystrings, either variable may be set to zero, and the resulting space reclaimed for the "top" buffer.

Another buffer variable is the NSIZE label: this defines the size of the file-transfer batch filenames buffer (in Kbytes). For each 1K allocated to this buffer, you can transmit 85 files at a time. Hence, if 85 files are not enough, you can set NSIZE to 2 and transmit 170 files at a time. If you don't use batch file transfers, you can set this variable to 0, and reclaim the space for the "top" buffer.

The XSIZE label defines the number of Kbytes for the file transfer disk buffer; note that this is not an allocated buffer, but resides within the terminal-mode file buffer. XSIZE is used to

restrict the actual size of the buffer, and should be less than or equal to ASIZE. This restriction is necessary in systems with extremely slow floppy disks, since there is the possiblity of the remote end timing out while MEX flushes its disk buffer.

PQSIZE defines the size of the queue used to service the modem at certain times (such as after sending an X-OFF to the remote when the terminal-file buffer fills up, during the overflow wait time SENDOUT strings, and within certain long loops, when there is a possibility of losing characters). This buffer must be a minimum of 2 bytes, and may be as arbitrarily large as you'd like. There is little to gain in increasing the size, however, unless you consistently run at speeds greater than 1200 baud.

You may at some time see the message "Not enough memory for MEX!" minimum buffer requirements. Normally, this will only happen in small memory segments (when you're running a small CP/M system, for example, or running in a small MP/M segment), but you'll also see it if you've increased MEX's buffers too much.

The only recovery possible is to reduce some of the buffers (or obtain more memory).

Appendix 2: Alternate Long Distance Service numbers (ALDS):

This appendix details the use of Alternate Long Distance Service (ALDS) numbers with MEX.

You may have two ALDS numbers defined; simply enter them as you would any other number, but give them a name of '>' or '<' (normal delay characters, passwords, etc may be included). Then, if you have a number you'd like to route through your ALDS service, simply prefix it with the associated '>' or '<'. An example should clarify this:

You have MCI service, your password is 98765, and it takes 2-4 seconds to connect after the number is dialed. You also have Sprint (you cover all your bases, don't you?), the password is 12345, and it sometimes takes 6 seconds to reach the number after it is dialed. Finally, you have a Hayes Smartmodem; a comma in the dialing string is a 2-second pause (is it really? I don't have a Hayes, so let's pretend).

In order to use both services, we'll put one number on the > key:

```
[MEX] A0>>PHONE >=555-9122,,98765          <<--- MCI
```

note the four second delay with the two commas, then the password.

Now Sprint:

```
[MEX] A0>>PHONE <=555-8144,,,12345
```

<longer delay, different password>.

Now RBBS Rockhead is a long, long distance call; it's available only through Sprint (and, of course, Ma Bell). We decide that if we can't make it through Sprint, we don't want to call RBBS Rockhead. Here's how we enter the number:

```
[MEX] A0>>PHONE ROCKHEAD=<202-555-1414
```

Now RBBS Aristocrat is our favorite BBS; if Sprint is jammed up, we'd like the option of dialing it over (ugh) Ma Bell lines. So we define it without an ALDS marker, like this:

```
[MEX] A0>>PHONE ARISTOCRAT=202-555-2222
```

Now notice that we can still call Aristocrat through Sprint or MCI with:

```
[MEX] A0>>CALL <ARISTOCRAT              <<--- Sprint
[MEX] A0>>CALL >ARISTOCRAT              <<--- MCI
```

But we must explicitly enter the ALDS symbol in the CALL command.

Since Rockhead is defined with a leading '<', it will always go through MCI; we don't have to supply an ALDS symbol in the CALL command (we can switch to the other ALDS number, however, by specifying the other ALDS symbol in the CALL command; eg,"CALL >ROCKHEAD" will switch to MCI even though we've defined Sprint as Rockhead's ALDS number).

In short, the left or right arrow specification is treated as if its ALDS number were part of the number being dialed.

Appendix 3: Hints on using SENDOUT and keystrings

There have been some queries regarding keystrings, and some problems reported in using them on RCPM systems. I'd like to clarify their use a little.

First, there is a variable that may be changed with the STAT command that directly affects keystrings: this varible is WTECHO. When this switch-variable is turned ON, MEX will send each character, then wait for it to echo from the remote end. If you're sending into a system that allows type-ahead (most time-share computers, Compuserve, Arpanet, some MP/M and TurboDOS systems, etc), it is best to turn this variable OFF; the keystring will be transmitted a lot faster. If you're dealing with an RCPM system, however, you might well overrun the receiving end with the keystring (especially if you're transmitting into a BASIC program, such as RBBS). For such systems, you should turn WTECHO ON; MEX will then wait for each character to be echoed from the remote.

The rest of this discussion addresses the same topic with respect to the SENDOUT command.

WTECHO also affects the SENDOUT command in the same way; there is a difference between SENDOUT and keystrings, however, that you must be aware of: SENDOUT tries its best to send the string correctly when WTECHO is on; if an echoed character is different than the transmitted character, SENDOUT will send a cancel character and try again (up to a retry-limit).

The retry-limit and the trigger and cancel characters are all STAT variables, allowing them to be changed with the STAT command. The trigger-character in the distributed MEX is the right-arrow ('>'); this is most handy for RCPM systems (which prompt each command with a right-arrow). If you want to use SENDOUT with non-CP/M systems (or with RCPM programs that use a different prompt), you'll need to change the trigger character. For example, if you're sending commands to a remote PIP, you'd want to use PIP's asterisk ('*') as the trigger, so you'd do:

```
STAT TRIGGER "*"
```

For sending commands to a Smartmodem (with the modem set to non-echo, which is the default for US Robotics), you'll want to set WTECHO to OFF and TRIGGER to 0.

Appendix 4: High-speed transfers:

I've been using MEX locally for transfers between two computers connected through an RS-232 link. File transfers at speeds of 9600 and 19200 are possible, with the following guidelines noted:

1) If one computer has a faster CPU clock speed than the other, it can receive at 9600 or 19200 without problems (assuming no extraordinary delay in the overlay modem I/O routines).
2) The receiver should use the 'Q' mode ('quiet': no status messages, but more relevantly, no console-status checking) if the transmitter is at an equal or greater clock speed.

3) It's not generally a good idea to view either received or transmitted characters at the receiving end, regardless of clock speed (i.e., avoid use of the V, S and R secondary commands at the receiving end).

4) Batch file transfers will work much better with the above speed patch.

Appendix 5: What's new in MEX114:

Release 1.14 of the MEX Modem EXecutive communications program adds support for 1k XMODEM file transfer packets.

Why 1k packets?

With the current proliferation of 2400 baud modems, it has become obvious that throughput (i.e., efficiency) of file transfers could be higher if more data could be added to the fundamental unit of exchange (i.e., the "packet"). The reason for this is essentially the "stop and wait" nature of the Christensen protocol: send a packet, wait for an acknowledgement, send a packet, wait, etc. When the packet size is relatively small, as it has always been with Christensen protocol, this "turn-around" time becomes a significant portion of the total time necessary to transfer a file. If the medium through which the transfer is taking place exhibits its own delay, the problem is compounded (all transfer media -- even hardwired RS232 connections -- have some media delay; this delay is much more pronounced in satellite telephone connections and packet-switched networks, such as Arpanet and Compuserve).

Conversely, using a large packet size with an inherently noisy medium can not only destroy any gains realized by using a the larger packet, but can actually increase file transfer time, because retransmission of a large packet takes longer than retransmission of a small packet.

So it seems logical that any large-packet protocol must also have the ability to "fall back", in the face of line noise, to the small packets that are so much more efficient in the noisy environment.

MEX 1.14 implements this fallback feature; it uses nearly the same algorthm employed by Paul Homchick in his 1k-packet modifications to the public domain XMODEM program (version 10.8 at the time of this writing). Further, the 1K packet option is entirely user-selectable; if you don't want to use large packets, simply continue using MEX as you've always used it; there's no penalty for not using large packets.

If you prefer the higher efficiency (and noisy lines are not a problem for you), simply append a "K" to the the "T" command when you're SENDING a file with MEX 1.14. In fact, you can make this change permanent by entering the command "GLOBAL K", then using the CLONE command to save your modified MEX 1.14 to disk (be advised, however, that if you do this, you run the risk of not being able to exchange files with versions of XMODEM or MEX that do not have the 1k packet capability, without expressly turning off the GLOBAL K).

MEX 1.14, when receiving, is always prepared to receive 1k packets, in any mixture with 128-byte packets. Thus, when you're preparing MEX 1.14 to receive a file, you need take no special action (in fact, the 'K' option, while accepted, is ignored in a file receive).

MEX, when transmitting, will adjust for line noise; after the third (not necessarily consecutive) error has occurred, MEX will calculate the ratio of errors to "virtual" 128-byte packets. If this ratio exceeds 1 error per each six 128-byte "virtual" packets, MEX will switch to 128-byte mode. Note that MEX will NOT switch to 128-byte mode until the next successive packet, however. Thus, once a packet has started as a 1k packet, it must finish as a 1k packet (otherwise, certain combinations of noise could cause the transfer to appear correct, but be received incorrectly). If you're using the batch option, MEX will always switch back to 1k packets at the beginning of the next file.

Note that MEX 1.14 is fully compatible with the emerging YMODEM specification authored by Chuck Forsberg of Omen Technology, insofar as 1K blocks are conerned (MEX does not "round up" an outgoing file to 1K, however -- it switches to 128-byte mode when the remaining outstanding byte count is less than 1024. This is permitted by the YMODEM specification).

Progress reporting

You'll notice that while transferring files in 1k mode, MEX will print "logical" record numbers on the screen (actually the starting and ending record numbers of the 1k packet being sent or received). Note that this is the 128-byte record number, and bears to relation to the packet number, which is part of the packet "envelope", t increments by 1 for each 1k packet, and is of no consequence to the user.

→→→→→→→→→→→→→→→→→→→→→→→→→→→→