the MOBY MUNGER

The official organ of the TECO Special Interest Group

Issue #3 - Part I August 1979

Contributions and correspondence should be sent to the editor:

Stanley Rabinowitz
6 Country Club Lane
Merrimack, NH 03054

or to

TECO SIG
c/o DECUS
Mail Stop MR2-3/E55
One Iron Way
Marlboro, MA 01752

Media intended for the newsletter editor should be sent to his work address:

Stanley Rabinowitz
Mail Stop MK1-2/EØ9
Digital Equipment Corporation
Continental Blvd
Merrimack, NH 03054

Camera ready copy is preferred but scribblings on the side of a napkin will be accepted. Ready to use material should be prepared on 8 1/2 x 11 inch white unlined bond paper. A one inch margin should be maintained on both sides, on the top, and on the bottom. Material should be reasonably clean, legible, and sufficiently dark copy for printing. Material intended for publication should be mailed in a large envelope so that it can be sent without the necessity of folding.

Readers are urged to submit articles, techniques, notes, hacks, ideas, comments, and bug reports to the Moby Munger for publication. This newsletter will publish on a more regular basis if more readers will submit material. It would be especially helpful if some people would volunteer to write certain columns regularly, such as the Beginner's column, or the TECO Techniques, or the NOTES FROM XXX, etc. Volunteers are urged to send their names in to the editor.

TABLE OF CONTENTS

This newsletter contains Part I, pages 1-41.
The next issue will contain Part II, pages 42-74.
The total Table of Contents appears here as a preview of more to come.

SIG NEWS

Symposia Notes:

NEW ORLEANS:

The TECO SIG was out in force, as usual, at the Spring 1979 DECUS Conference in New Orleans. A good 5Ø people showed up at the TECO SIG meeting. The discussion covered new versions of TECO, their new features, and how to get them, as well as new features in VTEDIT, obscure TECO macros, and much more. Many new members signed up.

We held two TECO tutorial sessions, the usual introductory session, and, by popular demand, a session on advanced TECO features.

SAN DIEGO:

Plans for the Fall 1979 Symposium in San Diego are shaping up very well. This symposium will be the best one yet for TECO fans. We expect to have a full day's worth of sessions, ranging from tutorials to workshops to formal papers. By popular demand, we will have a workshop on the internals of VTEDIT. There will be several tutorials on EMACS (a fantastic video editor written in TECO). We will hear about a TECO keyboard monitor and a TECO Macro library. There will be panel discussions on commenting conventions, using VTEDIT without a keypad, and structured programming in TECO. And lots more. Don't miss it!

As is our usual policy, all TECO sessions will be open to users of all computers, small, medium, and large systems. We will also have at least one birds of a feather session to discuss items that are not formally on the agenda - so bring along things to talk about.

At the time of this writing, it is not clear what hardware will be available, but it is expected that we will be able to copy media for -8's, -11's, -10's and -20's, so bring along any macros that you have that you wish to distribute to others. We will make up kits containing all contributed macros and make copies available to all who want them.

Also, the SIG will bring down the latest copies of all its supported TECOs. This will include TECO-8, TECO-11, and TECO-1Ø. PDP-11 programs will be distributed on DOS format MAGtapes. PDP-8 programs will be distributed on OS/8 floppies and/or DECtape. DECsystem-1Ø and -2Ø programs will be distributed on BACKUP INTERCHANGE MAGtapes (if a -2Ø is at the symposium) else on PDP-1Ø DECtapes.

We hope to have a TECO campground at San Diego where TECO GURUs and fans can congregate. Details should be in the symposium program. Don't forget to ask for a TECO sticker for your badge when you register.

SIG NEWS (CONT.)

How to join the TECO SIG:

Send your name and address to

DECUS
Mail Stop MR2-3/E55
One Iron Way
Marlboro, MA 01752

and say that you wish to join the TECO SIG. If you are a DECUS member, send your membership number. (If you are not a DECUS member, ask for an application blank.)

How NOT to join the TECO SIG:

Requesting SIG membership by writing to the newsletter editor will delay your request (possibly indefinitely).

SIG Activities:

Those of you who were in New Orleans saw our poster paper showing the first draft of our new compatibility level 1 TECO reference card. This is a new reference card which we are still in the process of designing. It describes all the TECOs that the SIG supports and compliments (rather than obsoletes) the TECO Pocket Guide which describes all of DEC's TECOs (and which was also produced through the efforts of the TECO SIG). When the new card is finished, it is hoped that it will be able to be distributed through DECUS. We are currently negotiating with them about this. It is expected that final copies will be available in San Diego.

Stan Rabinowitz is currently doing research into the history of TECO for a poster paper to be presented in San Diego. Readers bearing knowledge of little-known TECOs or of facets of the history of TECO should communicate the same to him. Especially wanted are old manuals which show the development of early versions of TECO.

The SIG is also preparing to submit packages of TECO macros to the DECUS Program library. So many readers have written in to tell us that they have lots of small macros that are too small to be worth submitting to DECUS. What we would like is for all these people to submit these macros (along with documentation) to the TECO SIG (c/o Stan Rabinowitz) and we will package them all into a single tape (or maybe more than one) and submit the whole as a package to DECUS. Each macro submitted should contain a comment at the top stating the name of the macro, the author's name, date, version number, and which TECO and which operating system the macro is designed to run on. A description of what the macro does and how it is invoked should also be included. If this description is lengthy, include it in a separate file with an extension of .TXT or .DOC and with name the same as the name of the TECO program.

SIG NEWS (CONT.)

Upcoming Activities:

VOLUNTEERS ARE URGENTLY NEEDED.

The TECO SIG has been run mostly by a small number of people. This is not good. Right now, if one or two key people should lose interest in TECO, the SIG would probably fold up. Also, our chairman has recently resigned. Jim Crapuchettes has agreed to act as temporary chairman until the end of the year. It is absolutely crucial that we get a new chairman at or before Fall DECUS. Note that DECUS rules prohibit a DEC employee from being SIG Chairman. Anyone who would like to be SIG Chairman should write to us immediately. The person should be dedicated to keeping the SIG active. He should be someone who expects to be at most DECUS Symposia and he (or she) should expect that he (or she) will remain interested in and actively using TECO for the next year or two. The SIG chairman handles correspondence with DECUS and answers queries from users. Most of the chairman's work can probably be performed by forwarding the appropriate correspondence to the appropriate DIGITAL counterpart.

IF THE SIG DOES NOT GET A PERMANENT CHAIRMAN, WE WILL LOSE ALL OUR FUNDING AND RECOGNITION FROM DECUS.

Keeping the SIG alive is not a one-man operation. Neither the chairman nor the newsletter editor should have to "do it all". The chairman should be able to delegate responsibility. We can use a library chairman (someone who will keep in touch with the DECUS library and report on new TECO submissions), a symposium chairman (someone who will call around and solicit papers to be given at future symposia), etc.

We also need a steering committee. This would consist of 4 or 5 users who could discuss SIG policy matters. Such things arise from time to time. For example, DECUS has recently informed us that the newsletter funding policy will change next year and that the SIG must decide how it will finance its newsletter. This will be something that will be discussed at the SIG business meeting in San Diego; but it would have been nice if the SIG steering committee could have discussed the problem amongst themselves first.

I would also like to see a sub-chairman for each operating system. The subchairman would answer all user queries having to do with the operating system he knows best (possibly by referring them to the appropriate DIGITAL counterpart). The sub-chairman would keep up with the latest information and problems with TECO on his operating system. He would probably receive the latest (pre-release) versions of SIG supported TECOs as they were produced (to field test). Write to Stan Rabinowitz if you wish to volunteer for any of the above positions, whether technical or administrative. It is critically urgent that the SIG be run and supported by users.

SIG NEWS (CONT.)

Availability of SIG TECOs.

The TECO SIG is actively pushing Compatibility Level 1 of TECO. This dialect of TECO is upward compatible with what you are familiar with and is compatible with each other. Compatibility level 1 TECO is implemented currently via three TECOs: TECO-8 which runs under OS/8, TECO-1Ø which runs under TOPS-1Ø and TOPS-2Ø, and TECO-11 which runs under RT-11, RSTS/E, RSX-11/M, RSX-11/D, IAS, and VMS. These versions of TECO are supported by the TECO SIG and have been created through the SIG's efforts.

STATUS

TECO-8 V7 is now currently available from the DECUS Program Library. Its order number is DECUS 8-913. For more information, see page 11.

TECO-11 V35 kits have been made for all PDP-11 operating systems, but as of this writing (17-August-1979) these kits have not yet been submitted to DECUS (although all the paperwork has been filled out). The kits are going through in-house field test. Also, work is still progressing on the new TECO manual. It is expected that TECO-11 will be available from DECUS within a month after you receive this issue of the Moby Munger. However, do NOT order it from DECUS until you have seen a formal announcement of its availability. You can also call Stan Rabinowitz [at (6Ø3) 884-5288 between 3 PM and 4 PM Eastern Time] to inquire about availability.

TECO-1Ø is still being field tested but should be ready for mass distribution before Fall DECUS. We will have copies there. Watch for further announcements before ordering from DECUS [or call the above phone number].

It is unfortunate that it is taking the implementors so much time to formally get these products into the DECUS Program library. Perhaps we should go back to the scheme of informal distribution by users pending official library submission. If you would be interested in field testing new TECOs as they come out and sending out copies to other users who sent you media, please contact Stan Rabinowitz. Send your name, address, phone number, hardware configuration, and operating system.

Anyone interested in implementing Compatibility Level 1 of TECO on some other operating system should contact Stan Rabinowitz. We will send you more information and encouragement.

NOTES FROM RSX

As of July 1979, the version of TECO available from the DECUS library for the RSX based systems is still V28 (under DECUS 11-333 for 11M and 11-334 for 11D and IAS). We will be releasing a new version very soon (probably V35). See page 6 for the latest information about availability of this version of TECO-11.

TECO-11 version 34 was distributed on the LUG tapes at the Spring 1979 Decus Symposium in New Orleans. It has many new features, including new control flow constructs, more VTEDIT support, and a much more sophisticated startup procedure. Details are documented in the kit.

This version, 34, has one known problem: When a text buffer is written to a file, any ESCAPE (ALTMODE) characters in the buffer act as record delimiters. Under most circumstances, this is harmless, since the record contains the ESCAPE character, and the input reader knows not to insert CRLF, so the whole operation is self-consistent. The only problem occurs when a file containing ESCAPE characters is to be used by some other program (such as a command file containing escape sequences). In this case, breaking escape sequences across record boundaries will generally cause problems. For the present, you can kludge around this bug by inserting the escape characters in the buffer as octal 233, which will not cause record breaks. This problem will be fixed in version 35 which will shortly be released to the DECUS library.

For those of you who have access to the RSX-11M Utility Fiche, the following is an index to the TECO source modules contained therein (as of March 1979):

Module	Fiche #	Frame #	Brief Description
TECO	43	NØ1	Text Editor and Corrector
SCREEN	43	NØ8	VT52 WATCH
.MAIN.	43	G1Ø	Prefix file for RSX-11 TECO I/O Assembly
TECOIO	43	B11	Data area for RSX-11 TECO I/O package
TECINI	43	L11	Initialization for TECO I/O
CCLCMD	43	G12	Command line processing login
GETFLS	43	N12	File opening and closing for TECO I/O
SWPFLS	43	E14	Swap primary and secondary streams
INDCLS	43	B15	Close indirect stream
INDERR	43	F15	Error in indirect file
OUTPUT	43	K15	Record I/O for TECO I/O
LISTEN	43	F16	TTY service for TECO I/O
RUBOUT	44	CØ1	RUBOUT and CTRL/U processing
TIAST	44	KØ1	Type-in AST service
TTYOUT	44	CØ2	TTY service for TECO I/O
EXSRV	44	NØ2	Random exec services for TECO I/O
EXIT	44	LØ3	Program exit for TECO I/O
ERRORS	44	FØ4	Error handling for TECO I/O

Note that TECO-11 V28 does not make use of the monitor type-ahead facilities that are available under the latest releases of the RSX-11 operating systems. (Thus running VTEDIT under those systems is not very useful.) V35 contains full type-ahead support.

Tektronix, Inc.

March 30, 1978

DECUS
129 Parker Street
PK3-1/E55
Maynard, Ma. 01754

Gentlemen;

We are using TECO version 29 under RSX-11M V3. Attached, please find a correction to module TTYOUT, which allows TECO to echo an old style ALT MODE (ASCII code 175) correctly as a $.

```
TYOUT - TTY SERVICE FOR TECOIO MACRO M11    01-FEB-78 10:00  PAGE 6

    32 000000                                       .PSECT  PURE,RO,D
    33
    34
    35                                      ;
    36                                      ; CHARACTER DISPATCH TABLE FOR TYPE OUT ROUTINE.
    37 000000  000010                       ;
    38 000002  000011                       TYTAB:: .WORD   BS
    39 000004  000012                               .WORD   TAB
    40 000006  000013                               .WORD   LF
    41 000010  000014                               .WORD   VT
    42 000012  000015                               .WORD   FF
    43 000014  000033  000175                       .WORD   AM,175          ; ECHO OLD STYLE ALT MODES ALSO
    44 000020  000007                               .WORD   BELL
    45         000022                       TYTABL  == .-TYTAB
    46
    47 000022  000304'                      TYDISP: .WORD   TORS
    48 000024  000334'                              .WORD   TOTAB
    49 000026  000262'                              .WORD   TOLF
    50 000030  000262'                              .WORD   TOVT
    51 000032  000262'                              .WORD   TOFF
    52 000034  000312'                              .WORD   TOCR
    53 000036  000320' 000320'                      .WORD   TOAM,TOAM
    54 000042  000326'                              .WORD   TOBELL
```

NOTES FROM RT-11

RT-11 V3B included TECO-11 V28 as an unsupported product. The kit also included a TECO-11 manual (for V27 TECO). Version 29 of TECO-11 was distributed at the last DECUS symposium.

In the last Moby Munger, we asked for volunteers to upgrade RT-11 TECO I/O to support V34. We received several takers, all with excellent qualifications. However, Mark Bramhall beat them all to it and RT-11 TECO V34 is now a reality. It will be released (unsupported) with RT-11 V4 which is expected to be released in early 198Ø. Copies of V34 TECO will be available at San Diego DECUS. (It will run under RT-11 V3B.)

New Features

General new features in TECO-11 V34 can be found elsewhere in this issue.

New features that are specific to RT-11 are:

(a) Auxiliary file support. TECO now permits two input streams and two output streams. EP switches to the secondary input stream and ER$ switches back to the primary input stream (the usual input stream). EA switches to the secondary output stream and EW$ switches back to the primary (usual) output stream. All the usual I/O commands, such as ERfile$, EWfile$, P, PW, A, Y, EF, EC, etc., work on the currently selected streams. For example, during the middle of an edit, if you wanted to append the contents of a small file FOO.TXT to the end of the current page (without losing your place in the current input file), you would issue the following commands:

```
EP                  switch to secondary input stream
ERFOO.TXT$          open FOO.TXT on secondary stream
A                   append text to end of text buffer
ER$                 return to primary input stream.
```

(b) Edit Indirect. The command EIfile$ tells TECO to accept additional TECO commands from the specified file instead of from the terminal. This new "input command stream" takes effect after the current TECO command string has finished executing (since the current command string up to the concluding $$ has already been read in from the terminal). The EI$ command switches the input command stream back to the terminal, as does encountering end-of-file in the indirect command file. While an EI is in progress, all immediate action commands (except $$) are suspended.

(c) Timer Support. The ↑H command now returns the number of seconds that have elapsed since midnight divided by 2. This is compatible with RSX TECO.

(d) Wildcard Support. This was described on pages 37 and 38 of issue number 2 of the Moby Munger. Briefly, ENfilespec$ sets up a wildcard file specification. Successive EN$ commands then load up pseudo-Q-register * with the name of the next file that matches the specification. All standard RT-11 wildcards are permitted. The EN$ command can be colon modified to return a success or failure indication.

(e) Private Initialization. The contents of a file called TECO.INI will be executed as TECO commands (if this file is found) each time that TECO starts up. This lets you perform custom initialization, such as flag settings. A sample TECO.INI is included in the release kit. This file also describes the meaning of returned values from TECO.INI.

(f) Customizable command line parsing. Your EDIT/TECO command line is normally parsed by a TECO macro built into TECO. However, if you have a file called TECO.TEC on SY:, then this file will be invoked to do the command line parsing. A sample TECO.TEC is included in the release kit. It is recommended that only TECO wizards use or modify this file.

(g) Immediate mode CTRL/C command. Now, you can use CTRL/C to exit TECO without having to type a double ALTMODE. Since RT-11 doesn't have a CONTINUE command, accidental exit from TECO is undesirable, so it will be necessary to type two CTRL/C's in a row to exit. Temporarily, since people need time to get used to this feature, it will be necessary (in V34) to type four successive CTRL/Cs to get out of TECO. Sometime in 1981 we will reduce this back to two CTRL/Cs to be compatible with other TECOs.

Version 4 of RT-11† will support the TECO, MAKE, and MUNG commands. These will be compatible with other PDP-11 implementations of these commands. MUNG will be slightly different from the EDIT/EXECUTE command as shown below:

Monitor command	Corresponding TECO command
EDIT file/EXECUTE:text	ERfile$YHXZHKItext$MZ$$
MUNG file,text	Itext$EIfile$$

These commands are almost identical (but not quite). Note that for your TECO program to automatically start up when invoked by a MUNG command, it must end with two ALTMODEs.

Under RT-11 V4, you will also get TECO command 'memory' as has been available under TOPS-1Ø and OS/8 (and more recently under other PDP-11 operating systems). Memory is implemented via a file called TECFØØ.TMP on DK:. Then, when you type a TECO foo.bar command, TECO will remember your filename so that later you can merely type TECO to the operating system and you will automatically begin editing foo.bar . Note that .R TECO will always enter TECO (as usual) with no memory.

† probably only the FB version.

NOTES FROM OS/8

The latest DEC-supported TECO for OS/8 is called OS/8 TECO V5 and is distributed as part of the OS/8 V3D extension kit. The PDP-8 group has no plans to re-release TECO in the near future.

The TECO SIG intends to support enhancements to OS/8 TECO. Through the aid of Stan Rabinowitz, they have created a new product, called TECO-8, which is now in the DECUS Program Library. TECO-8 is based on OS/8 TECO but is written in MACREL (rather than PAL8) and contains many new features. Any time that DEC decides that it is willing to release and support TECO-8, the SIG will be happy to give it to DEC.

TECO-8 version 6 was distributed informally at Spring DECUS in New Orleans. TECO-8 V7 is now available from the DECUS Program Library. The order number is DECUS 8-913. It is available on two media - DECtape or floppies. The media code for the DECtape is H3Ø and the media code for the two floppies is K5Ø. There are two floppies - a source floppy and a binary floppy. If you have no need for the sources, you can order just the binary floppy. Very explicitly, specify that you want the binary floppy only and send in $25 rather than $5Ø. (The DECtape kit contains both sources and binaries on one tape.) There is also a write-up (media code A2) but this is identical to material appearing on the machine-readable media, so you should not order it if you are ordering one of the kits.

Bug reports on OS/8 TECO V5 should continue to be submitted to DEC in the usual manner (with copies to the TECO SIG). Bug reports on TECO-8 V7 should be sent to the TECO SIG c/o DECUS. Do NOT send TECO-8 problem reports to DEC.

Probably the most important feature of TECO-8 is its window support (written by Jim Roth). This allows TECO to maintain a window into the text buffer on the screen of a VTØ5, VT52, or VT1ØØ terminal (similar to the VR12 window support that already existed). This allows writing true video editors in TECO and in fact, one is included in the TECO-8 kit. It is called VTEDIT version Ø and is compatible with VT52.TEC which was distributed with early versions of TECO-11. It runs on VT52's and VT1ØØs. A picture of the VTEDIT keypad layout occurs on page 9 of the last issue of the Moby Munger.

TECO-8 also supports expanded length MUNG command lines and allows an alternative way to chain to TECO so that the command line will be parsed by a TECO macro called TECO.TEC instead of by CCL. Unfortunately, both of these features requires corresponding support by CCL, which is not present in the CCL distributed by DEC. Perhaps sometime in the near future, a fix to CCL will be issued by the SIG.

The new features in TECO-8 which were not in OS/8 TECO V5 are described on the following pages.

TECO-8 V7 has the following new features:

1. Symbiont support. This allows TECO-8 to run under the OS/78 operating system while a symbiont is running.

2. V1 support. This provides a window feature into the text buffer (for use with VT52 and VT100 terminals).

3. W command. -1W updates window into text buffer. 0W resets cursor line to its default position. nW if n is positive, sets the cursor line to line n. If n is negative, nW causes the VT support to 'forget' the top -n-1 lines on the screen. These commands are only operable if VT support is enabled.

4. The (^S) immediate mode command no longer gives you an error message if used immediately after a (^S) had already been used.

5. :EK and :EB commands. These are the same as ER and EB, however, if the specified file is not found, then no error message is generated. Instead, no new file is opened and the command returns a value of 0. If the command had succeeded, then a value of -1 is returned.

6. The /S switch is now legal in filespecifications. For example, ERFOO.TE/S$ is permitted. This switch puts TECO in 'SUPERTECO' mode. In this mode, TECO will be able to read through end-of-files. A (^Z) in an ASCII file will not mean end-of-file. Instead, the (^Z) will be read in like any other character. (^Z) is not considered to be a line terminator or a page terminator. TECO reverts back to normal mode the next time an ER, EW, or EB command is issued and no /S switch appears within the filespecification.

7. F_ command. Similar to FS and FN but _ searching is employed. This command is affected by yank protection and the 2's bit in the ED flag (qv).

8. ^L command. This command types a form feed on the terminal when it is executed.

9. The ^A command can now be @-sign modified. The form is thus @^A/text/ where / denotes any delimiter that does not appear within the text. If the ! or 0 command is @-sign modified, unpredictible results will occur.

10. ED flag bits. Two bits have been implemented.

    1. The 1's bit (bit 11). If this bit is on (1), then up-arrows (^'s) in search strings are treated as ordinary up-arrows. This is the default. If this bit is off (0) then an ^ in a search string is a

special string build character. It takes the next character and converts it to its corresponding control character. The result is as if that control character had appeared in the string.

2. The 2's bit (bit 10). If this bit is off (0), then Yank protection is enabled. Yank protection causes the commands Y, _, and F_ to abort if there is text in the text buffer and there is an output file open. This is the default. If this bit is on (1), then Yank protection is disabled (thus all Y commands would work regardless of possible loss to data).

11. When you chain to TECO.SV, the chain argument (TECO command) can now be much larger tha before. The argument passed must be in ASCII (one character per word) beginning at location 17600. Such characters must be positive (bit 0 off). If a negative number is encountered, then this tells TECO that the command is continued at the specified address in field 1. A word of 0 terminates the command. This feature is not of much use at the present time.

12. If you are not on a PDP-12, and you do not have VT support, then you get an extra 7 characters in your search buffer (limits length of search strings).

13. Several new bits in the EI flag have been implemented:

    1. Bit 10 (value 2). This bit is initially on if you had typed a SET TTY SCOPE command to the monitor and off if you had typed SET TTY NOSCOPE. User modification of this bit has no effect.

    2. Bit 9 (value 4). This bit is initially on for compatibility with TECO-11 macros. That means that TECO will read lower case characters typed on the terminal as is, and will not convert them to upper case. User modification of this bit has no effect.

    3. Bit 6 (value 32). This bit has meaning only if VT support is present. This bit is initially off (0). If turned on, then ↑T commands read a character if one is available in the type-ahead buffer (or the terminal buffer), but if no input character is available, then this command returns a -1. This bit is cleared by TECO everytime TECO returns to prompt level.

    4. Bit 5 (value 64) is initially 0 and is not currently used by TECO-8.

    5. Bit 4 (value 128). This is now the abort on error bit. This bit is initially on. Whenever this bit

is on (1) and TECO prints an error message, control subsequently returns to OS/8. If this bit is off (0), then control returns to TECO's prompt. Whenever TECO prompts with a *, this bit is turned off.

6. Bit 3 (value 256). This bit only has meaning if VT support is available. If this bit is on, then displayed lines are truncated at the right edge of the screen. If this bit is off (default), then lines that are longer than the terminal width are continued on the next line of the display.

7. Bit 2 (value 512). This bit is on initially if you have VT support and off if you don't. User modification of this bit has no effect.

8. Bit 1 (value 1024). This bit is on initially if you have a PDP-12 and VR12 support is available. It is 0 if not. User modification of this bit has no effect.

9. Bit 0 (value 2048). This bit enables trapping of CTRL/C's. If this bit is on (1), then whenever a CTRL/C is typed, this bit is turned off and the CTRL/C is entered into TECO's type-in buffer just as any other character. It has ASCII code 3 and may then be read by the ^T command. Whenever this bit is off (0), then typing a CTRL/C has special effect as usual (causes a ?XAB error message if TECO was executing commands). This is the default. This bit only has effect if VT support is enabled.

14. VT05 scope support (at editing level) is now automatically established if you have a VT05 and you had performed a SET TTY SCOPE command to the OS/8 monitor. This support enables the CTRL/U immediate mode command to properly erase the line from the screen. It also allows rubouts to work properly on VT05s.

15. The CTRL/R character is now identical with the CTRL/Q character within search strings. It causes the next character to be interpreted literally.

16. The ?POP error message now tells you whether it was a C, R, or J command that attempted to move the pointer outside the bounds of the text buffer area. If another command (such as nA or m,nK or m,nP) tries to access a pointer position outside the text buffer, you also get a ?POP error code with a different message. Furthermore, if an nD command tries to delete non-existent character positions, no characters are deleted and you get the ?DTB (Delete Too Big) error message.

17. The "= command. It is synonomous with the "E command.

18. The error codes ?NAC, ?NAS, and ?NAY have been changed to ?NCA, ?ISA, and ?NYA respectively.

19. Type ahead. If VT support is enabled, then you can type-ahead to TECO.

20. EXIT protection. The EX, EC, and EG commands now abort (with the ?NFO error) if there is text in the text buffer and there is no output file open. This is meant to protect user's from accidental loss of data. The commands can always be executed by proceeding them with an HK.

21. Auto ? feature. If bit 9 (4's bit) of the EH flag is on (1), then TECO will simulate the action of a ? after printing an error message. This automatic action does not occur when bit 9 is off (0), the default.

22. If TECO is chained to, then before it parses your TECO command, it will look to see if you have a file called TECO.INI on SYS:. If you have one, then it will be read into Q-register W and it will be executed. If an error occurs in this file, then the TECO command that you passed to TECO will not be executed. After this command has been executed, Q-register W will be cleared and TECO will proceed to execute your command that was passed to it upon chaining (i.e. an EW command if you had invoked TECO with a MAKE CCL command). When that is done, TECO will execute Q-register X if and only if the TECO.INI program returned a non-zero value. It is up to TECO.INI to load up Q-register X and return a value (preferably 1) if this feature is to be used. A useful way to use this feature is to have TECO.INI load Q-register I with an editing macro and then put an MI command in Q-register X.

23. If TECO is chained to and location 7600 in field 1 contains a 12-bit 0, then TECO works completely different than before. In that case, TECO assumes that a CCL command (like TECO FOO/INSPECT) is being passed to it rather than a TECO command. Such a command begins at location 17601 and consists of ASCII text terminated by a fullword 0. A negative word indicates a pointer to subsequent text. TECO will then parse this CCL command by invoking SYS:TECO.TEC with the specified command left in the text buffer. TECO.TEC will execte out of Q-register V which will be cleared upon conslusion of command parsing. This feature would be useful if CCL were modified to pass TECO commands directly to TECO rather than parsing them itself. This gives the user much flexibility in adding options to his TECO commands.

24. When an illegal character occurs in a command such as Eq, if that character is a tab, TECO will print it as <TAB> rather than as <HT> in the error message.

25. If the contents of EH bits 10 and 11 is 3, then TECO is put into super extended error message mode (war and peace mode), provided you have at least 16K and no VT support or 20K with VT support. In this mode, each error message will be accompanied by an explanatory paragraph of helpful information culled from the file SYS:TECHLP.TXT which can be edited by the user to suit his own needs. (Note: always leave TECO and re-enter it after editing TECHLP.TXT since TECO notes its location upon startup and unpredictible results will occur if this file is moved during TECO's execution.)

26. With the same memory restrictions as above, if the EH help level is not set to 3, you can still get the paragraph of help information by typing / as the very first character after TECO's prompt. This will give you a detailed song and dance about your previous error.

27. TECO will turn off trace whenever it comes back to prompt level.

28. The * immediate mode command has been implemented. If typed as the very first keystroke after TECO's prompt, it will save away the previous command string in Q-register Z. It will also type a Z to inform you of this fact. It is exactly equivalent to the CTRL/S feature of OS/8 TECO V5 and is preferred since CTRL/S may go away in a subsequent release.

29. The ^_ command. n^_ returns the ones complement of the number n. Note that ^_ is a TECO command and is not a unary operator.

30. The V command. This command is identical to OTT and was put in for TECO-11 compatibility. This command currently does not accept an argument. The OTT command is preferred inside macros since it will run faster. (The V command is located in a little-used overlay.)

31. The "D and "A conditionals. These test the numeric argument to see if it is the ASCII code for a digit or alphabetic character respectively.

32. The EY command. This is equivalent to Y but is not affected by Yank protection.

33. The immediate action commands LF and BS have been added. If typed as the very first keystroke after TECO's prompt, LF will cause an effective LT command to be immediately executed and BS (Backspace or CTRL/H) will cause an immediate -LT command to be executed. No double ALTMODE is necessary.

34. Additional error messages were added to warn of commands such as [A and EP that are implemented in other TECOs but not available in TECO-8.

35. The m,n<flag> commands were added. They turn off those bits specified by m and turn on those bits specified by n.

36. Display of search string in search failure error message now shows the match control constructs as <NOT>, <ANY>, and <SEP>. <ESCAPE> is displayed as <ESC>.

NOTES FROM TOPS-10

No activity going on from DEC. TECO V24(2Ø2) is the latest released TECO. It is supported by DEC but is quite out of date.

There are many user-written upgrades to DEC's TECO. Perhaps the one that is most widely used is TEXAS/STEVENS TECO V124. For anyone who wants it, send a magtape to Clive Dawson, Computation Center, University of Texas, Austin, Texas, 78712. So far, this is the best -1Ø TECO we've seen around (not counting the SIG's new TECO-1Ø described below). Information about other TECOs is wanted. We will be happy, in this column, to disseminate information and bug reports about other TECOs.

The TECO SIG is now pushing its new product, TECO-1Ø. It was written by Andy Nourse and is based upon XTEC. It has all the known XTEC bugs fixed and is compatible with TECO-11. TECO-1Ø runs under both TOPS-1Ø and TOPS-2Ø. TECO-1Ø V3 is currently out for field test. It will be submitted to the DECUS Program Library within the next 2 months. We will have copies available for copying at San Diego (depending on hardware availability). TECO-1Ø supports the W and :W commands (as in TECO-11). These maintain a window into the text buffer on the screen of a VT52, VT1ØØ, and several other terminals. True video editors are easy to write, and in fact, two will be distributed in the TECO-1Ø kit. One is called, VTEDIT, and is compatible with the video editor (VØ.Ø) by the same name that runs on PDP-11's. Another, called VT is patterned after the TV editor that runs on the DECsystem-2Ø.

Some of the features of TECO-1Ø are: Symbolic Q-register names (6 character significance), it is a compiler rather than an interpreter, log file capability, EE command to create runnable TECO programs, scope support, :ER and :EB, EK, nV, n↑T, :Gq, support for TECO.INI, bounded searches, edit indirect, :J, :C, and :R, and much much more. More information in the next issue of the Moby Munger.

NOTES FROM TOPS-20

TECO is not a supported DEC product. DEC's TOPS-1Ø TECO will run under TOPS-2Ø, but has no scope support and does not mesh well with the operating system.

TECO-1Ø (described above) will run well under TOPS-2Ø. This will be available from DECUS within the next few months. Wait for an official announcement before ordering it. We believe you will be greatly impressed by this product and we hope it will become the standard TECO used on the -2Ø since it is compatible with the other TECOs at DEC. TECO-1Ø is supported by the SIG.

A variant of TECO, called TV runs under TOPS-2Ø and will be supported by DEC. They are also working on a manual which will be out soon. More information about TV can be found elsewhere in this issue.

MIT TECO also runs under TOPS-2Ø. This is an incredible TECO, much more powerful than anything at DEC. It, together with EMACS, can be obtained by sending a magtape to Richard Stallman, MIT Artifical Intelligence Laboratory, 545 Tech Square, Cambridge, MA 02139.

NOTES FROM VAX/VMS

The version of TECO-11 distributed with VAX/VMS release 1 was V29. Some of you may have gotten V34, which was distributed at the Spring 1979 DECUS Symposium in New Orleans. It has the same problem converning ESCAPEs as record terminators as the version of TECO (V34) that was distributed for RSX (see NOTES FROM RSX in this newsletter).

No documentation or help file was included for TECO in VMS release 1. Many of you have already found or figured out how to implement conveniently the usual TECO commands. For those of you that haven't, the following describes how:

In your login command file, define three symbols for the three TECO commands:

```
$ TECO :== "$TEC TECO "
$ MAKE :== "$TEC MAKE "
$ MUNG :== "$TEC MUNG "  .
```

You can then invoke TECO with the TECO, MAKE, or MUNG commands as on other operating systems.

It is expected that TECO will continue to be released with the subsequent releases of VMS. TECO V35 will shortly be submitted to DECUS. See page 6 for details about availability. V35 will also be available at San Diego for copying.

New I/O Features:

The /B2 switch is now recognized within an EB, ER, or EW command. It has the same meaning as the corresponding switch in RSTS/E TECO.

Print format files (FD.PRN) can now be read (but not EI'd or written).

CTRL/C handling has been fixed. Two CTRL/C's in a row will cause TECO to cleanly exit. (Three CTRL/Z's in a row will do the same thing. This helps the termination of TECO in batch streams.)

TECO.INI is now supported. This provides a clean interface for user initialization. If a file by this name is located in your top level directory, it will be executed when TECO starts up. It can contain flag initialization commands, or any other TECO commands that you desire. It could be used to load up Q-registers, etc. TECO.INI can also return a value. The meaning of the bits in the returned value are described in the sample TECO.INI file included in the kit. Use of TECO.INI to perform initialization is strongly preferred over using a private command decoder (TECO.TEC). TECO.TEC is intended to be used only by GURUs who wish to change the way TECO parses its command line. TECO.INI is intended to be used by the average person who wishes private initialization.

On the following page is a listing of DIRECT.TEC, a program to let you get directories with embedded wildcards, to be used until VMS version 2 comes out (which has this support).

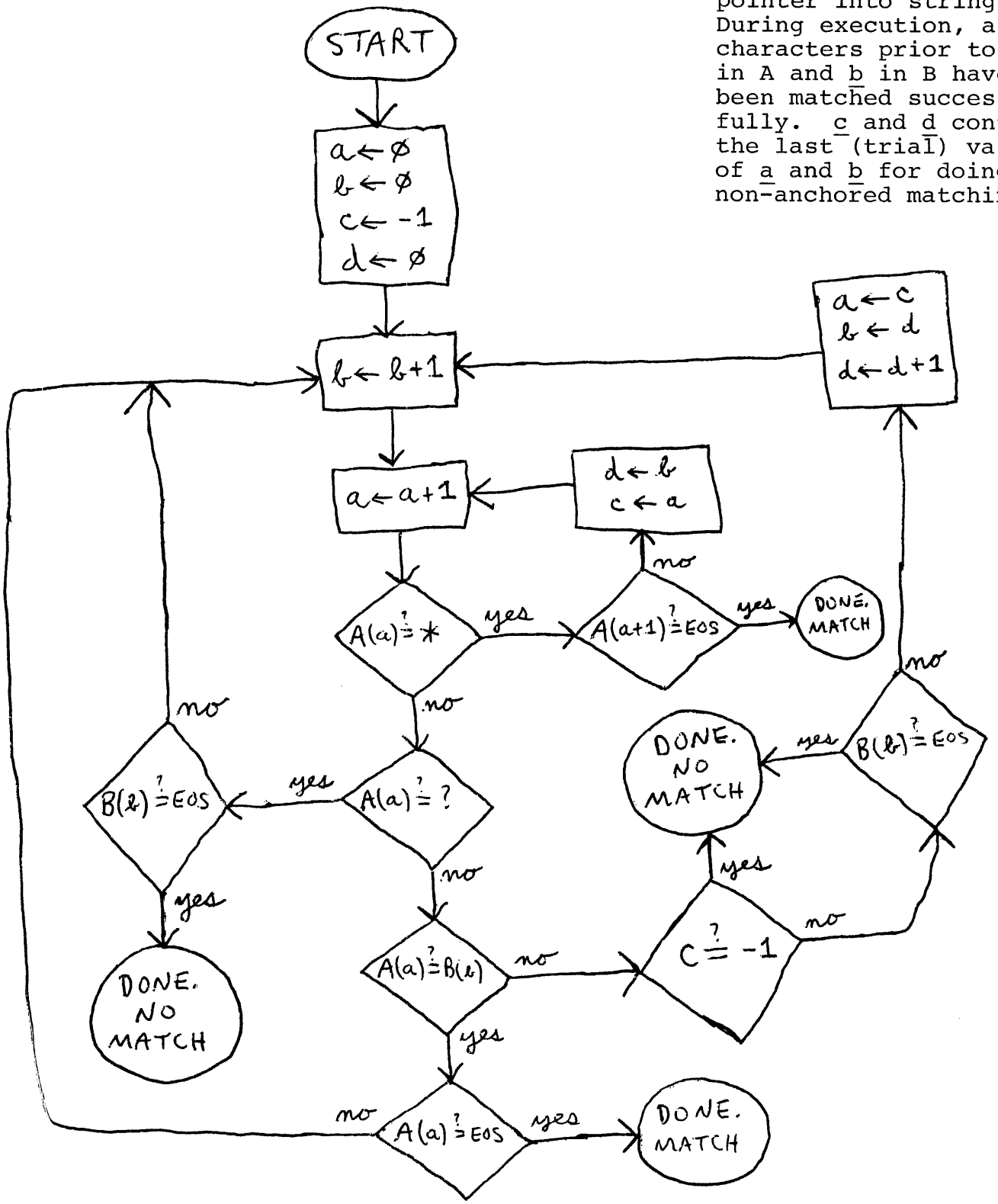
```
! DIRECT.TEC Version 2.0                    22-July-1979                    !
! Author: Herb Jacobs                                                        !
! Modifications by: Stan Rabinowitz                                          !

!--------------------------------------------------------------------------!
! Load Q-reg S with macro that compares a string to see                      !
! if it matches another string with wildcards.                               !
! This part of algorithm by David Spector.                                   !
@^US\J L .-1U1 -1U2 -1U3 0U5 J
1<%2AU6 !LOOP1! %1AU7
Q7-^^*"E Q1+1A-13"E 1U5 0;' Q2U4 G1U3 @LOOP1@'
Q7-^^%"E Q6-13"E 0;' F< '
Q7-Q6"N Q3"L 0;' G6-13"E 0;' Q3U1 %4-1U2 F< '
Q7-13"N F< ' 1U5 > HK 0S\
!--------------------------------------------------------------------------!

@^UC\
\ !                                        Load CRLF into Q-reg C          !
J<S S;-D> !                                Remove all spaces               !
<FS?S%S;> !                                Allow ? as synonym for %        !
<FS<S[S;> <FS>S]S;> !                      Allow <dir> as well as [dir]    !
J:S/S"T ^AOptions not allowed^A HKEX' !    Don't allow switches            !
0UV J:S:S"T 0XV 0K 1UV' !                  V contains device name          !
0UW J:S]S"T 0XW 0K 1U<' !                  W contains directory            !
J 2@:fs/./;/S J:S,S"T !                    allow .ver as well as ;ver      !
^AComma list not allowed^A HKEX' !         No comma list permitted         !
0UZ J:S;S"T P XZ K 1UZ' !                  Z contains version no.          !
1UY J:S.S"F ZJ I.*S P 0UY' P XY K !        Y contains extension (filetype) !
1UX Z"E I*S 0UX' HXX HK !                  X contains filename             !
GW 0UB J:S*S"T 1UB' J:S%S"T 1UB' QBUW !    W=1 if wildcards in directory   !
QW"N  J:S.S"S
^AWildcards in subdirs not allowed^A  ! Illegal wildcard in subdirectory!
HKEX ' '
HK QY"N GY J:S*S"T 1UY' J:S%S"T 1UY'' ! Y=1 if wildcards in ext (type)  !
HK QX"N GX J:S*S"T 1UX' J:S%S"T 1UX'' ! X=1 if wildcards in name        !
HK GV GW GX GY GZ

! Change partial wildcards to full ones.                                  !

J<FS%S*S;> <JFS^ER*S*S;> <JFS*^ERS*S;> <JFS**S*S;>

^UU*S JIEMS ZJ 27IS HXRMB !                Do a general wildcard lookup    !

<:ENS; HK G* J S:S 0K S]S 0XD 0K S.S R 0XF 0K S;S P 0XT 0K XV HK 1UT

! Get the next matching filespec from VMS.                                 !
! Delete device, Put dir in D, filename in F, type in T, version in V      !
! Q-reg U contains name of current directory (initially has dummy name)    !
! Numeric part of Q-reg C is column counter (4 columns per line)           !

QW"N GD GC GW GC ^SUT' QT"E F< ' !          Skip if directory doesn't match !
QX"N GF GC GX GC ^SUT' QT"E F< ' !          Skip if filename  doesn't match !
QY"N GT GC GY GC ^SUT' QT"E F< ' !          Skip if filetype  doesn't match !
QZ"N GV GC GZ GC ^SUT' QT"E F< ' !          Skip if version   doesn't match !
G* J:S^EQUS"F !                            It matched, prepare to type it  !
:QU-1"N 3<:GC> ' !                         Don't advance first time        !
S]S -1UC 0XU :GU :GC ' !                   Type directory (if new)         !
JS]S 0K %C"E -4UC 0UB :GC ' !              Reset column counter if needed  !
QB<^A ^A> HT 20-ZUB > :GC !                Use 20 column fields            !
HKEX !                                     Exit back to VMS                !
$$
```

Wildcard Searching Algorithm (in Q-reg S)
used by DIRECT.TEC

End of string (EOS) is carriage return.
A is input pattern (2nd line in buffer).
B is current trial string (first line in buffer). Q-reg 1 points into A.
Q-reg 2 points into B.

NOTES

a contains a position pointer into string A. b contains a position pointer into string B. During execution, all characters prior to a in A and b in B have been matched successfully. c and d contain the last (trial) values of a and b for doing non-anchored matching.



Continued on page 49.

The Moby Munger
c/o Stanley Rabinowitz
6 Country Club Lane
Merrimack, NH 03054

Dear Sir:

I have recently installed TECO-11 V34 on our VAX system from the distribution tape dated 21 April 1979. When using V34, I have noticed an inconsistency between it and V29 in the handling of <ESC>'s in a file. The attached print-out shows a series of edits for a file used to send an escape sequence to a VT100.

The original file contained the single line: $WRITE SYS$OUTPUT "<ESC>H<ESC>J". Note that passing this file through V29 leaves it unchanged. However, if this file is passed through V34, the <ESC>'s are treated as line terminators and 3 records are produced. This trick is particularly confusing since when examined with V34 the file appears unchanged.

The problem was solved here by modifying the routines GETBUF and PUTBUF in GETPUT.MAC. In GETBUF, the dispatch table GETDSP was modified to dispatch to ADCRLF when an <ESC> is encountered. This causes an <ESC> at the end of a record to be treated as an <ESC><CR><LF> sequence. In PUTBUF, the three instructions beginning with CMPB R3, #AM were deleted. This prevents <ESC> from being treated as a line terminator. This modification can introduce a problem if you try to edit a file containing <ESC>'s which was previously editted with V34. The old records terminated by <ESC> pick up a <CR><LF> when editted with the new version. The spurious <CR><LF>'s must be deleted to correct this file.

One other annoyance I have found is the naming of the TECO memory file. On the VAX, the memory file is always TECFOO.TMP in the user's highest level directory. However, the convention at our site is for several related users to share a common highest level directory. Thus, the contents of the memory file changes as other users do their editing. The current fix is to use TECO.INI to strip the directory name from the memory file spec causing the file to go to the current default directory. This is less than desireable since it leaves memory files lying around in various subdirectories. A better solution would probably involve generating the file name to contain the creator's process number or name.

I hope these comments are of some interest to you or other members of the TECO SIG.

Sincerely,

Gary L. Grebus
Information Systems, Modeling and
Applied Statistics Section

GLG/ff

Editor's comments: Under Hybrid TECO, the memory information is stored in a process logical name called TEC$MEMORY so that each user will have his own memory. The ESCAPE problem has been fixed in V35. More info on (new) hybrid TECO in next issue.

NOTES FROM RSTS/E

TECO-11 was released with RSTS/E V6C as an unsupported product. V34 was distributed at the last DECUS Symposium. V35 will shortly be submitted to the DECUS Program Library. See page 6 for details of availability of this release of TECO-11. Currently, in the DECUS Library, is DECUS-RSTS-11-1Ø5 which contains an old version of TECO-11 (version 15, not version 24 as reported in the last issue of the Moby Munger). This package is useful because it contains several manuals (not written by members of the TECO SIG): A reference manual, a beginner's manual, and a system manager's guide. All are machine readable.

New features in RSTS/E I/O module:

If running under RSTS/E V7.0:

(a) Tentative files are used

(b) Wild card PPN's are available with EN

(c) No supersede mode is recognized

The following sequential disk file formats are handled:

ASCII stream

Fixed

Variable

Variable with fixed control

FORTRAN Carriage Control

Implied Carriage Control

Print format control

The following ANSI magtape file formats are handled:

Fixed format

Variable format

ASCII stream

FORTRAN carriage control

Implied carriage control

Embedded carriage control

On output, if the magtape is an ANSI magtape, the format written is variable (format "D") with implied carriage control (modifier " ") and a default block size of 2Ø48 bytes per block. The block size can be explicitly set with the /CLUSTERSIZE:n switch.

RSTS/E :EG - TECO special function processing

:EG is implemented as follows:

:EGRTS$	switch to system default RTS
:EGRTS foo$	switch to RTS "foo"
:EGFSS string$	file string scan "string"
:EGCCL cmd$	try "cmd" as a CCL command
:EGRUN file$	try to run "file"
:EGRUN file=xx$	try to run "file" with "xx" placed in core common
:EGEMT$	issue a monitor directive; the FIRQB is loaded from Q-regs A through P and the XRB is loaded from Q-regs Q through w; the low byte of the value in Q-reg A is the monitor EMT code to issue; if the high byte of the value in Q-reg A is >0 then the text part of Q-reg A is put into the XRB for a 'write' (XRLEN= size of A, XRBC=size of A, XRLOC->A); if Q-reg A high byte is <0 then the text part of Q-reg A is put into the XRB for a 'read' (XRLEN=size of A, XRBC=0, XRLOC->A)

Returned value is -1 for success, 0 for unrecognized command, or >0 for the RSTS/E error code

The FIRQB is placed in the numeric part of Q-regs A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P

The XRB is placed in the numeric part of Q-regs Q, R, S, T, U, V, W

CDL TECO

TO: the TECO world
FROM: a TECO fanatic

Budapest, May 2, 1979.

A TECO V3c written in CDL is ready to use!

A runnable TECO can be generated for every machine that has a CDL compiler. For example: IBM 360/370, PDP 11, CII 10010 (a french one), Z80 (yes, microprocessors too!) and in the very near future for the INTEL 8080 and MOTOROLA 6800.

CDL is a very high level language. (Its name is the acronym of Compiler Descriptor Language.) So our TECO is structured, modular, easy-to-modify and easy-to-upgrade.

At the present we have TECO V3c alive on the Z80. She is 99% the same as the PDP-8 TECO V3.

Differences:

12 bit arith.	16 bit arith.	
EB	none	(edit with backup)
EG	none	(edit-and-go)
none	V	(verify line)
^S	*	(*Z at later TECOs)
^F	^F	(no switch register - always 0)
none	nGq	(inserts n times the Q-register)
=	=	(always unsigned).

This TECO will not be submitted to DECUS because it is in fact a machine-independent software product rather than a DEC-oriented one. It is available costless, on program exchange basis. Modifications or upgrades are possible if requested.

Contact with

Nagy András

Polytechnical University of Budapest
Dept. of Instr. & Meas.
H 1521 Budapest, XI.
Muegyetem rkp. 9.
HUNGARY, EUROPE

PROBLEM OF THE MONTH

This column will present interesting and/or unusual problems for the reader to solve. Solutions and problem proposals for future columns should be sent to the editor: Stanley Rabinowitz, 6 Country Club Lane, Merrimack, NH 03054. Solutions to problem 2 should be postmarked by Dec. 15, 1979.

Problem 2: Write a TECO macro that locates itself.

Details and testing procedure:

I will take the macro that you submit and load it into one of the alphabetic Q-registers (A through Z). I will then load up the other Q-registers with any text that I feel like, with the one proviso that I will not load up another Q-register with an exact copy of your macro. However, I may load up other Q-registers with near copies, or with any other text that I feel might 'fool' your macro. Then I will start up your macro with the appropriate M command. Your macro must print out the letter of the name of the Q-register in which it is running. For example, if I put your macro into Q-register H, then the MH command should print the letter "H". Your macro should then return to TECO command level with no other output.

Unlike the last problem of the month, we are not concerned with how quickly you can solve this problem, or with how small a solution you can find, but only with "can you solve the problem at all". Although we would like your solution to run under all DEC TECOs, use of minor incompatibilities will not necessarily disqualify your solution, if the editor can fix up the incompatibilities easily.

Refer to the TECO Pocket Guide for system compatible commands. Do not use any features of TECO that are extremely peculiar to one particular operating system. Note that if your macro desires to use a Q-register as temporary work storage, it should probably use one of the Q-registers Ø-9, since writing into one of Q-registers A-Z might destroy the Q-register from which your macro is executing - a circumstance which may cause unpredictable results.

The problem editor reserves the right to reject any submission which is too large to type in (say over 250 characters) unless it is submitted in machine-readable form. (The author has a solution that will fit on 3 lines of this magazine and runs under all DEC TECOs.)

Winners' names will be printed in the first issue of the Moby Munger following the conclusion of this contest.

*** challenging problems for future issues are wanted ***

SOLUTION OF THE MONTH

Our first problem, given in the last issue of the Moby Munger, was to write a minimal macro that would type out the upper case letters A to Z. We received many machine-dependent solutions of the form

```
26<%A+64 ↑T >
```

which had to be disqualified because one of the conditions of the problem was that no character may be repeated, and in these solutions, the character '6' is used twice.

There were two classes of competition - Class A: machine independent and class B: machine dependent. At first, I was a bit worried about the problem, because I thought that it was too dull and too restrictive - I expected everyone to find the minimal machine independent 12-character solution:

```
26<Z+ ↑↑ AI$>HT
```

and, in fact, several people (Martin Pring, Doug Hosking, Prof. P. L. Balise, Prof. Colin Daly, and B. Perrette) found this solution or its equivalent, such as 26<9Ø-ZI$R>T .

Boy was I surprised when an 11-character solution came in! It was hard to believe. Not only that, but three correct 11-character class A solutions came in, and they were all different. The grand winners in class A (listed in order of receipt of entry) together with their solutions are listed below:

Alan Frantz,	DEC	91<.I$>65JT
Chester Wilson,	Australia	91<ZI$>26RT
Doug Hosking,	Lynn, MA	91<ZI$>65JT

Class A winners - 11 character machine-independent solutions

My worries that the problem was dull were over quickly as mail started to pile up. Over 4Ø people have submitted solutions to the problem. Your interest in TECO is very gratifying. Typical of the comments I have received about this contest is the remarks from Profs. Peter L. Balise and C.H. Daly of the University of Washington:

> *We put these solutions together shortly after reading the problem. We worried about the problem further for a few more days, and then abandoned it since we had one or two other things to do. Now we hasten to send you our solutions so we'll be eligible to win the trip to Hawaii. Actually, we are not too delighted with our solutions because they seem fairly obvious. We suspect there is some clever way to do it with fewer characters.... We are deeply worried that our solutions don't utilize the rule that lower case is to be considered different from upper case.... Most of our effort was to get around the fiendish rule prohibiting duplicate digits.*

Class B competition was just as keen. The earliest winning solution was only 1Ø characters long and was submitted by Patrick Keogh of Wagga Wagga, New South Wales. His solution is

```
26<%1+↑↑@↑T>
```

This solution works on TECO-8 and TECO-11 but is not valid on TOPS-1Ø TECO because that TECO does not support the n↑T command (which types out the character whose ASCII code is n). That solution, or a similar one, such as ↑↑↑Z<%A+64↑T>, was also submitted by several other readers:

Michael N. Levine,	China Lake, CA
Eric Leventhal,	New York, NY
Profs. Balise & Daly	University of Washington
T.F. Runge & R.F. Davis	Sandia Labs, Albuquerque, NM
Peter Balkus, *et. al.*	Gen Rad, Concord, MA

(Other winning solutions came in after the deadline date.)

Another valid 1Ø-character solution was submitted by Robin Miller of Derry, NH. Her solution runs only under version 26 of TECO-11. It is

```
↑V<65+ZI$>V
```

which relies on the fact that the version number command, CTRL/V will evaluate to 26 under that version of TECO. Also, for those of you not familiar with the V command, it is equivalent to ØTT .

Several solutions were disqualified because they required additional set-up, not specified by the testing procedure that accompanied the statement of the problem. For example, Robin Miller submitted the solution ↑F<65+ZI$>V which requires that the computer's switch register previously be set to 26.

Stan Rabinowitz submitted the 6-character solution: ERA$YT which relied on the fact that he had a file on his disk called A which just happened to contain the letters A through Z. Stan claims that this file permanently resides on his disk. He says he never knows when it will come in handy. He also has a file called B that contains the text of the Declaration of Independence and a file called C that contains precisely 4Ø96 copies of the word "SYZYGY", and many other files, too many to mention. We had to disqualify him too.

David Morrison (from the University of Newcastle, New South Wales) submitted a 4-character solution: EGX$ which runs under his RSTS system. His system contains a CCL command "X" which invokes a program to print the required output. (Under RSTS TECO-11, the EGcmd$ TECO command exits TECO and causes the operating system to execute the specified command.) DISQUALIFIED.

Mario DeNobili (of Liechtenstein) submitted a one character solution, namely: ? . He claims he modified his version of TECO so that the ? command printed A-Z. He claims the old use of this command (enter trace mode) was useless to him since he never makes mistakes.

COMMENTS FROM READERS

We received much too much correspondence from our readers to publish it all. We print below some of the more interesting remarks. Be assured, however, that all suggestions have been passed on to the appropriate people and is very much appreciated.

Notes from Eric Leventhal:

The newsletter is a nice one, but I'd rather have one that's less thick and more frequent.

Sorry, but this is the best I can do. If someone else would like to take over editing this letter (or parts of it) that would be fine with me. In fact, any help at all would be appreciated. If people would send me finished articles (ready to print) such as BEGINNER'S Column, macro of the Month, Techniques, etc., that would make my job that much easier.

There should be gosub-return in TECO (not just macros). Some mode for no deblocking or conversions by TECO would be nice. This would cause A, Y, P, etc. to bring in one cluster or block and P, PW, etc. to write only full blocks. No conversions or dropping of nulls would be done. This would be very useful for special applications.
There should be a command to swap the top of the stack and a Q-register.
Some possible nice functions: swap byte, insert ASCII of radix-50, compare values of two top entries on stack, kill file, close input file, rename file, a case-checking command, a /LIBRARY switch that causes TECO.TEC to search for a macro library and load up Q-registers.

Notes from H. S. Hopkins Jr.

To help study TECO, I found some simple macros in the OS/8 manual and tried to use them [under CTS-3ØØ]. I was not able to find even one macro that worked directly, but they did serve my study purpose. What I wanted was: to modify VEG and VT52 to better suit my purpose, a justification macro, a cleaner exit from VT52.TEC editing, a comment column alignment macro. After writing these, and making sure they worked properly, I was a confirmed TECO believer. I'm still not sure whether these macros are of general value, and ask your opinion as to their usefulness for publication.

These are indeed useful. I urge you to submit them to the Decus library.

A copy of his text justification macro (untested) is reproduced on the top of the next page. Readers are urged to produce more sophisticated versions and submit them to the Decus program library.

Hopkins' text justification macro:

```
ETUA$1ET$
^AJustify Right Column at ^A$(^T-48)*10UZ$(^T-48)%Z$
^A,
Don't touch lines less than Column ^A$(^T-48)*10UY$(^T-48)%Y$
^A......Macro in process.........^A$
QAET
J !1! QUN QUS!
!<QNA-32"E 1%S$'!
!QNA-13"E QJUSTIFY$'!
!1%N$>$$!
!!JUSTIFY! QN-QY"G!
!QZ-QN-QS"G QS<S $I $S^N $ -1^W>!
!OL QS%N$ QS%S$ QJUSTIFY$'!
!QZ-QN"G QZ-QN<S $I $S^N $-1^W>''!
!L Z-."GO1$'$$
```

This macro has the shortcoming that characters past the right margin 'set' will not move from line to line.

Notes from Nick Carr:

I can think of several things that would be most useful to us learners:

1) A bigger beginner column with plenty of examples
2) Listings of some useful macros, especially the VT52 scope macros and information on how best to implement them
3) Some general 'internals' on how TECO actually works (such as how it works as a Run Time System with RSTS/E).

Notes from Charles South III:

Fixing RUNOFF files can be done more simply than you indicated by the following:

```
TECO file/CR=file/-CR
*EX$$
```

We've been using TECO for this purpose for some time now and it works fine.
(I'm not sure this works for very large files, especially ones that you wish to pass to a source compare program. In cases of doubt, use the two pass method given in the last issue.)

I am in full agreement with the *Z limited usefulness commented on by Eric Osman. TENEX TECO indeed uses exactly the 15 char minimum length criteria recommended by Eric and we have found it works quite well in practice.
A long-standing gripe I have had about TECO is that the horizontal tab character is interpreted as an insert command. That destroys the user's ability to properly comment his macro, unless you preprocess the file to remove such tabs later. Personally, I consider that the horizontal tab should be totally ignored (just like the space). Failing this, at least restrict the meaning so that the insert is only implied if the tab is the first character of the current TECO command string.

Notes from Alan Lehotsky (DEC):

If someone with a lot of time and energy were to re-implement TECO in Common BLISS, it would be infinitely simpler to guarantee compatibility on every architecture except the PDP-8. Using the technique that PDP-11 TECO follows with respect to isolating the "system" interface from the rest of TECO would allow DEC-Standard TECO to be easily transported among all major hardware-software systems, including RSX-11(M,D), IAS, RT-11, VMS, TOPS-1Ø, TOPS-2Ø, UNIX, ...

Notes from Chester Wilson:

I have an RT-11 sort-of-batch system which allows use of TECO through indirect files. If this interests anyone, I'll send you more details. (His address is: 71 Galatea St., Charleville Australia 447Ø)

Notes from Thomas W. Burtnett (Dickinson College Computer Center):

One of the redeeming characteristics of TECO has been its consistency, sometimes elegance in syntax. For example, once m,nT is mastered, it is a short step to m,nK or m,nP. But then along came m,nS. Ugh! Then along came m,nFB. Yeah! But why couldn't m,nS do what m,nFB does? Find some other construct for the m,nS function. For the good of the language, change it!

Reply from Stan Rabinowitz: I agree that m,nS is pretty ugly, however it does not contradict the TECO syntax rules. The main TECO rule that applies is as follows:

> *Any command that takes a single numeric argument that represents the number of lines over which a command is to act, may also be invoked with two numeric arguments. In that case, the two numeric arguments represent the buffer position bounds between which the command is to act.*

For example, nT types n lines, so m,nT would type between pointer positions m and n. The same applies to K, FB, FC, and X. The argument to S does not represent a number of lines. nS means search for the nth occurrence. Hence the above syntax rule does not apply. In cases where the above rule does not apply, the two argument form of the command can have any meaning that is convenient. If I were redesigning TECO, I would make m,nS equivalent to m,nFB (as in STEVENS TECO). However, it is too late now to change TECO-11. Many macros already use m,nS. Also, this construct includes capabilities that are not easy to simulate by other commands. The TECO SIG recommends that the FB command be used to do all normal bounded searches in the future. TECO-10 supports FB too.

MACRO OF THE MONTH

This macro was written by Andy Goldstein and Stan Rabinowitz. It is useful on certain PDP-11 operating systems to create printable copies of TECO macros. (Some PDP-11 operating systems cannot easily print arbitrary TECO programs because their line printer driver does not print ESCAPEs as $ and do not have an up-arrow mode for printing control characters.)

```
! TECPRT V2 6-June-1979 !

! Program to create printable copies of TECO macros. !

!*************************************************************!
!                                                             !
!       Calling sequence:                                     !
!                                                             !
!       MUNG TECPRT,filename                                  !
!                                                             !
!*************************************************************!

!
! J IER$ !                          Create an ER command
! :S.S"F ZJI.TEC$ ' !               Make .TEC the default extension
! ZJ 27I$ !
! HXA MA !                         Execute the ER command to open the input file
! J S.$ RK I.LST$ !                Change the extension to .LST
! ^ACreating file ^A !             Tell user what file is being created
  J2C -DI*$ T 13^T 10^T !         after creating appropriate EW command
! ZJ 27I$ !
! HXE MR EKY !                     Execute the EW command to open the output file
!

<
    <.-Z;
    0A-31"G C F< '
    0A-09"E C F< '
    0A-10"E C F< '
    0A-11"E D4<10I$> F< '
    0A-12"E D8<101$> F< '
    0A-13"E C F< '
    0A-27"E D1$$ F< '
    0AU1 DI^$Q1+64I$
    >
^N^_; F>
^ADone^A EX
$$
```

REMARKS ON SOME NON-STANDARD FEATURES:

For those of you not familiar with PDP-11 TECO implementations, here is an explanation of some of the non-standard TECO features used in the above program:
The MUNG monitor command is used to run a TECO program. The command MUNG MACRO,text invokes TECO with the initial TECO command being Itext$EIMACRO.TEC$ which has the effect of executing the commands in the file MACRO.TEC with "text" initially residing in the text buffer.
The command n↑T types out the character whose ASCII code is n.
The command F< is a fairly new command that flows back to the first character in the beginning of the current iteration.

THE VAX UNIT PROCESSING EDITOR

David Leblang

The Unit Processing Editor (UPE) is a VT52/VT100 oriented word processing editor. It runs on the VAX as a TECO-11 macro package. Other macro packages exist (and are distributed with TECO) but UPE differs significantly in that it takes a very different view of editing and editors. UPE has a formal command syntax which is given later on in this article. An informal description of typical commands and a demonstration of UPE's power are given below.

Informally, UPE performs some action on some region of the current buffer. The action may be to 'uppercase' all the characters in the region, or to 'delete' all of the text in the region, or it may be any of several other GENERIC ACTIONS that UPE supports. A list of currently supported generic actions is given in the formal definition.

The region on which a generic action is a performed is established by the user through the use of a UNIT, a basic direction, and an ITERATION. The UNIT may be 'word', 'sentence', 'line', or any of several other units; the ITERATION is some signed number, like '4' or '-15'. The basic direction is determined by the ACTION's default basic direction and the sense (positive or negative) of the ITERATION.

Consider the following sequence of examples where a command is expressed as a triple whose elements are: { ITERATION, UNIT, ACTION }. The examples show a continuing sequence of commands applied to the text buffer. The initial text buffer is:

```
+-------------------------------------------------------------------+
| Let this be the text buffer.  And let the char slashed be the     |
| char the cursor is positioned on at the time this window          |
| is viewed.  The result of the command is also shown.  In UPE      |
| the cursor (not a slash) is actually on top of a character.       |
+-------------------------------------------------------------------+
```

Example 1: Command { 2, sentence, uppercase }

Starting from the cursor, and looking to the right (which is the default direction for 'sentence'), UPE finds the 2nd end of a sentence (a sentence is defined in the documentation). UPE establishes a region between the original cursor position and the point just found. With the region established, the action is performed on all of the contained text yielding the following buffer:

```
+-------------------------------------------------------------------+
| Let this be the text buffer.  And let the char slashed be the     |
| char THE CURSOR IS POSITIONED ON AT THE TIME THIS WINDOW          |
| IS VIEWED.  THE RESULT OF THE COMMAND IS ALSO SHOWN./ In UPE      |
| the cursor (not a slash) is actually on top of a character.       |
+-------------------------------------------------------------------+
```

Example 2: Command { 4, word, delete left }

The basic direction for 'delete left' is to the left, so UPE finds the 4th start of a word looking back from the cursor. Looking forward, UPE looks for the end of a UNIT while looking back it looks for the beginning of a UNIT. In this case, the region is established and all of the text within the region is deleted. The following buffer results:

```
+-------------------------------------------------------------------+
| Let this be the text buffer.  And let the char slashed be the     |
| char THE CURSOR IS POSITIONED ON AT THE TIME THIS WINDOW           |
| IS VIEWED.  THE RESULT OF THE / In UPE                            |
| the cursor (not a slash) is actually on top of a character.       |
+-------------------------------------------------------------------+
```

Example 3: Command { -1, sentence, lowercase }

The basic direction for 'lowercase' is to the right but, since a negative ITERATION was given, the region is established by looking back to the left for the first beginning of a sentence. All actions have a basic direction. In addition to the "forward to the end of" and "backwards to the beginning of" basic directions already shown, there is a "forward to the beginning of next" and a "backwards to beginning of previous" basic direction.

```
+-------------------------------------------------------------------+
| Let this be the text buffer.  And let the char slashed be the     |
| char THE CURSOR IS POSITIONED ON AT THE TIME THIS WINDOW           |
| IS VIEWED.  the result of the / In UPE                            |
| the cursor (not a slash) is actually on top of a character.       |
+-------------------------------------------------------------------+
```

DEFAULT UNITS AND NON-GENERIC COMMANDS

A short sequence of keystrokes is desirable. For this reason, the ITERATION may be omitted, in which case it defaults to 1, and the UNIT may be omitted, in which case it defaults to the default UNIT of the generic action performed. For example, the default unit for 'uppercase' is 'word', the default unit for 'delete' is 'character'. The UPE command { 9, , 'delete left' } deletes 9 characters, the UPE command { , , 'uppercase' } uppercases a word.

Some actions do not involve a UNIT, rather they prompt the user for information and perform an action based on the information supplied by the user. Actions of this type are called PROMPT ACTIONS. The format of a prompt action command is a triple: { ITERATION, PROMPT ACTION, PROMPT RESPONSE }. Before the user enters a response to the prompt, a small window at the top of the screen is opened and the prompt message is typed by UPE. The user then responds

Example 4: Command { 3, find substitute, 'thier$their' }

The prompt " 3*Find Substitute: " is typed in the window, the user responds as shown above, and UPE changes the first 3 occurrences of 'thier' to 'their' starting from the cursor looking forward. If 3 occurrences cannot be found, an error message is typed in the window (Failing Search for "thier"). Those occurrences encountered before the failure are replaced, and the cursor is left at the end of the last occurrence found and replaced.

Example 5: Command { -4, search, 'bufptr' }

The prompt " 4*Reverse search: " is typed in the window and the user responds as shown above. UPE searches for the fourth occurrence of 'bufptr' looking backward from the cursor. Note that the default direction for searches is forward and that a negative ITERATION (-4) causes the search to proceed in the opposite direction.

OTHER ASPECTS OF UPE

Some actions cause changes in the editing environment rather than in the text buffer. These actions may be PROMPT-ACTIONS or SIMPLE-ACTIONS (actions that do not involve UNITs or PROMPTS). These classes of actions affect the major modes or minor modes of UPE. Mode settings change things like terminal width and length or whether ordinarily non-printing characters are displayed.

The 'Learning Sequence' minor mode:

When UPE is "learning" every character typed by the user is saved by UPE until the learning minor mode is exited. The user can then have this 'learned sequence' repeated ITERATION number of times. The sequence remains around until the user requests another learning sequence. A UPE learning sequence is like a TECO macro except the commands are high level commands. Common learning sequences can be loaded from a file as well.

The 'BLISS' major mode:

In BLISS mode the definition of 'sentence' changes to mean statements ending with ';' and 'paragraphs' are defined by "BEGIN" and "END". The CRLF is interpreted to mean 'CRLF then indent to previous level'. The tab key indents by logical, rather than physical tabs.

IMPLEMENTATION ISSUES IN UPE

Since UPE and other editing packages run in a limited address space as a TECO-11 macro package, it is important to keep the macro package small. UPE uses two techniques to achieve a moderate size. The first is in the general organization of the editor: there is a UNIT definition section that establishes the region on which the action will be performed by looking at the current UNIT and ITERATION. The UPE generic action section then performs the chosen action on the region established. This means that UPE can perform any action on any region. If a new type of unit were desired, UPE would add code to establish a region for { ITERATION, new unit }. Once UPE can establish a region for the new unit, all generic actions can be applied to that unit.

Another technique used is to keep parts of UPE that do not require fast responses in an auxiliary file. UPE directs TECO to execute from an auxiliary file when one of these commands is typed. The auxiliary file system is invisible to the user and provides for features like an extensive help facility, while taking no memory away from the user.

UPE FORMALLY

This article is primarily about the philosophy of UPE so there is no discussion of which keystrokes cause what to happen . However, there is a UPE wall chart summary of commands, and will soon be available a UPE User's Guide. Contact David Leblang *(c/o the TECO SIG)* for more information. Below is a formal definition of UPE.

UPE is defined using the following notation:

X ::= Y	There exists a production X into Y
A ¦ B	A or B
[A]	A is optional
A (B ¦ C)	A followed by B, or A followed by C
'Function'	Name of a keyboard function terminal
<'Function'>	Some small set of related functions

UPE is defined as:

```
UPE COMMAND ::=   [ITERATION] [UNIT] GENERIC-ACTION
                ¦ [ITERATION] PROMPT-ACTION [user response]
                ¦ [ITERATION] SIMPLE-ACTION

ITERATION ::=   [+ ¦ -] DIGIT-STRING

UNIT      ::=     'Word' ¦ 'Line' ¦ 'Sentence'  ¦ 'Character'
                ¦ 'Paragraph' ¦ 'Page' ¦ 'File' ¦ 'Region'

GENERIC-ACTION ::=
                  'Delete Left' ¦ 'Delete Right' ¦ 'Fill Text'
                ¦ 'Lowercase' ¦ 'Uppercase' ¦ 'Capitalize'
                ¦ 'Indent' ¦ 'Undent' ¦ 'Copy Text into Save Area'
                ¦ 'Move to Previous' ¦ 'Move to Next'
                ¦ 'Move to End of ¦ 'Move to Start of'

PROMPT-ACTION ::=
                  'Execute Teco Command' ¦ 'Abort Edit'
                ¦ 'Display and Set Major Modes'
                ¦ 'Search' ('Bounded' ¦ 'Unbounded')
                ¦ 'Find Substitute' ('Bounded' ¦ 'Unbounded')
                ¦ 'Load File into' ('Save Area' ¦ 'Current Position')
                ¦ 'Load File into' ('Learning Sequence' ¦ 'Teco Seq.')
                ¦ 'Read and Insert Character verbatum'
                ¦ 'Read and Ignore Characters until LF'

SIMPLE-ACTION ::=
                  <'Quick Motions'> ¦ <'Quick Line Actions'>
                ¦ <'Screen Redrawings'> ¦ <'Help Commands'>
                ¦ 'Repeat Search' ¦ 'Repeat Teco command'
                ¦ 'Set Mark for Region' ¦ <'Set Minor Modes'>
                ¦ 'Start/Stop/Execute Learning Sequence'
                ¦ 'Write File/Exit to Monitor' ¦ 'Exit to Teco'
                ¦ 'Insert last deleted or saved text at cursor'
                ¦ 'Abort Current Command'
```

LANGUAGE ORIENTED EDITING

David Leblang

TECO-11 macro packages, like UPE (described in the last article), have provided an implicit experiment in editors. Conclusions drawn from these editors and from the success and failure of other editors give indication that time might be well spent developing an editor that is very different from any of the editors currently around.

These macro-package editors were gradually developed in a software engineering environment and given instant feedback from the engineers using them. Limited by a small address space, and unhindered by product commitments, features were removed from the editors to literally make room for other, more important features. The resulting editors reflect many of the needs and concerns of software engineers in the currently existing development environment.

Comparing two of these editors, VTEDIT and UPE, many similarities were noted, the most interesting being the commands that attempt to tailor the editor towards a particular language. VTEDIT has a normal English-text set of definitions for words, a BASIC-PLUS set of definitions, and the ability to define the set of word delimiters. UPE has a normal English mode and a BLISS mode for which UPE provides alternate definitions and auto-indentation.

The users of these macro-package editors as well as the users of the MIT EMACS editor (which runs on TOPS-20 and has PL/1, LISP, and TEXT modes) find their editor more functional because it knows about the underlying syntax of the language being edited. The success of word processing systems, which know about the underlying language (English), is an additional indication that a language-based editing system can be very useful. Despite improvements brought about by the macro package editors and EMACS, there is still a long way to go toward making editors more language-oriented, so as to make the user's job easier and his work more productive.

In the DEC Software Engineering environment, a highly interactive, transportable, multi-language editor would provide a significant link in the development of high quality software. Interactive capabilities have been shown by word processing systems to provide increased productivity. Transportability of the editor would pave the way for multi-system availability as well as avoid programmer retraining and errors due to switching between editors having different characteristics. The need for a multi-language editor can best be shown by the flurry of requests to add new languages to DEBUG-VAX which currently supports MACRO, FORTRAN, COBOL-74, and BLISS.

DIFFERENCES BETWEEN TECO-11 V33 AND V34:

TECO-11 V34 has the following new features:

1. Many new flow control commands have been added. These allow you to write more structured code. It also reduces the need for certain types of GOTOs that are very slow in TECO.

   (a) F> flows to the > at the end of the current iteration. The iteration count is bumped and the iteration is re-executed if necessary.

   (b) F< flows to the < at the beginning of the current iteration. The iteration count is not affected. If this command is issued when you are not in an iteration, flow transfers to the beginning of the current macro level.

   (c) F| flows to the else clause of the current conditional. *(What a useless command!)*

   (d) ↑C↑C causes an unconditional exit from TECO. The second ↑C must be a real CTRL/C.

   (e) $$ if executed as a TECO command causes a branch out of the current macro level. The second $ must be a real ALTMODE (as opposed to an ↑[).

   (f) nOtag1,tag2,tag3,...$ is a computed GOTO. Control branches to the nth tag specified. If n is out of range, control continues in-line. None of the tags used may contain embedded commas.

   (g) :; is used to exit an iteration. It is the opposite of ; that is, n:; exits the current iteration if n<Ø. A :; may also immediately follow a search command in an iteration, in which case the iteration is terminated if the search succeeded.

   (h) F' flows to the ' at the end of the current conditional.

2. m,nUq is equivalent to nUqm .

3. FB and FC were implemented as in Harvard TECO. FB is a bounded search and FC is a bounded search and replace (Find and Change). Each takes an argument (or argument pair) as in the T, X, and K commands, to specify the range of characters in which the search is to proceed.

4. EY is the same as Y except that there is no Yank protection. E_string$ is like _string$ but with no Yank protection.

5. :P, :Y, :EY, and :A now return a value, Ø if eof, else -1.

6. n:A reads in n lines and returns a value (Ø if eof, else -1). If the argument is omitted, :A is like the Append command (A) and not like the 1:A command.

7. All flag commands (like ET, ED, EU, etc.) now have a two argument form. m,n{flag} turns on those bits specified by n and turns off those bits specified by m. Thus, for example, if you want to turn on the 'read with no wait' bit, you would issue the Ø,32ET command.

(continued on bottom of next page)

DIFFERENCES BETWEEN TECO-11 V3Ø AND V33:

TECO-11 V33 has the following new features:

1. The ↑R radix command has been implemented. ↑R returns the current radix and n↑R sets the radix to n. Currently, the only legal radices are octal, decimal, and hexadecimal. TECO's prevailing radix affects the interpretation of digit strings and the action of the \ and n\ commands.

2. n=== and n:=== are used to print the value of n in hexadecimal.

3. If LF is the first keystroke after TECO's prompt, an effective LT command is executed. If BS is the first keystroke after TECO's prompt, an effective -LT command is executed. *(In V34, these commands have been changed to have better interaction with EV processing. Also, BS is a NOP if .=Ø .)*

4. EI processing has been changed so that execution of a TECO file looks more like execution of a macro. TECO used to treat CTRL/U's in an indirect file as the immediate mode command (erase current line), etc. In V33, all immediate mode commands (except $$) are suppressed during execution of an indirect command file.

5. 2EJ returns your PPN or UIC. -1EJ returns an operating system-dependent value. TECO programs can test this value to see what computer and what operating system they are running on.

6. The :EGcmd$ command has been implemented. It performs an operating system-dependent function and returns a value.

7. Bit value 4 of the ED flag now means "no arbitrary memory expansions". In this mode, a Y, P, or N command will not expand memory. The A command will, however, still expand memory if there are not 256 free bytes in the text buffer.

8. A whole slew of new W and :W commands have been implemented. These implement new scope modes and allow support for new terminals. Some of these commands can be used to speed up video editor macros.

(continued from previous page)

8. Bit value 16 of the ED flag means "do not reset 'dot' to zero on search failure".

9. "= is a synonym for "E .

1Ø. CTRL/R is the same as the string build CTRL/Q command.

11. The CTRL/EUq string build construct uses the character whose ASCII code is in the numeric portion of Q-register q.

12. Match control construct CTRL/EB is the same as CTRL/S.

13. Several new error messages: ?IRA - illegal radix argument, ?NYA - numeric arg with Y, ?IPA - negative or Ø arg to P, ?IUC - illegal character after up-arrow.

14. nA now returns a -1 if position accessed is outside buffer.

EG does not insert a 3-digit number with leap year information, but rather a blank line.

EO (set dumped on tape bit) does not exist.

EQ (create link) does not exist.

<n>EP is the same as EP.

EX<file>$
if a file is open for output, does EE<file>$, then instructs the EXEC to repeat the last CCL type command (load, execute, compile, debug).

FS CCL FNAME$
a string, in the same format as FS D FILE$, of the jfn given in AC1 if TECO was started at the CCL entry point; or zero if it was not or the filename has already been read.

FS D VERSION$
-1 has its usual meaning, since all versions are numeric.

FS FDCONVERT$
when given two args, will use the first as the ODTIM format. When give no arg, will do an IDTIM from the buffer, and can thus parse formats other than those created by FS FDCONV$ with an arg.

FS HELP MAC$
there is no way to input a HELP on 20X.

FS OS TECO$
returns the operating system TECO is running on, 0 for ITS, 1 for 20X.

FS UPTIME$
returns its value in milliseconds, rather than 30ths of seconds.

SIXBIT quantities.

commands which return a SIXBIT word as a value, actually return string pointers on twenex. The F6 commands do not convert strings to sixbit, but just pass strings through, so that the difference can be made transparent by using them after any command that returns a SIXBIT quantity.

filenames.

TECO attempts to convert as much as possible ITS style filenames to 20X style, including quoting special characters, to allow many more macros to work without conversion.

†TOPS-2Ø

DIFFERENCES BETWEEN TEXAS TECO V124 and V124A

In the last issue of the Moby Munger, we described Texas/Stevens TECO V124(33Ø). This TECO has been upgraded by Clive Dawson at the University of Texas. The current version, V124A(347), EO level 3, is available from Clive at the Computation Center, University of Texas, Austin, Texas 78712 for a nominal postage and handling fee. If you want a copy, send Clive a magtape. The new features of V124A are described below:

1. MAKE file1=file2 has been implemented to be compatible with DEC's TOPS-1Ø TECO V24.

2. nV has been implemented to be compatible with TECO-11 V28. That is, nV is identical to (1-n)TnT . V is the same as ØTT .

3. The /DEFAULT switch can now be used on a filespec to cause any previous "sticky" defaulting to be canceled.

4. m,nUq is the same as nUq but returns the value m. Thus, UiUj is useful at the start of a macro to save away its two arguments. *(TECO-11 liked this command and will be in V34.)*

5. The [command now allows one or two arguments. n[q is equivalent to [qnUq and m,n[q is equivalent to [qm,nUq . *(This is useful when passing arguments to macros. I feel that the TECO-11 method is more general. In their scheme, m,n[q is equivalent to [qm,n that is, arguments can pass through PUSHES and POPs.)*

6. :nA appends n lines from the input file to the end of the text buffer. :A is the same as :1A .
(TECO-11 implementors got complaints when they implemented this, and later versions of TECO-11 changed :A to be the same as A but returned a value of Ø or -1 . This is more consistent with the rest of TECO.)

7. nA now returns the ASCII value of the nth character to the right of the pointer, that is, the character at position .+n-1 in the buffer. This is the (.+n)th character in the buffer.
(This is very controversial. Note that it is one off from how nA in OS/8 TECO and TECO-11 work. The reason for this is that PDP-1Ø users have gotten into the habit of using 1A to access the current character, rather than ØA . Comments regarding this problem are hereby solicited.)

8. 2ET now causes a more literal typeout mode to occur than 1ET . In 2ET mode, tabs, vertical tabs, form feeds, etc., are sent directly to the terminal rather than simulating them with spaces and line feeds.

9. Search speed has been improved by an order of magnitude via use of the Boyer-Moore fast string search algorithm which was implemented by Jim Thomas.

1Ø. Searches in iterations no longer return values unless they are colon modified.

11. Five new terminal types are now supported: Micro-term ACT-IV and V, Lear Siegler ADM-3A, Hazeltine 15ØØ, and Hewlett Packard 264Ø.

12. The defaulting scheme for the EI and EP commands has been changed, but I won't go into the details here.

13. A TECO or MAKE command can now be followed by a dollar sign ($) and then followed by an initial TECO command to be executed by TECO. For example,

        TECO TEST.RNO$1ØØ<A>5T

    will cause the given commands to be performed after TE$T.RNO is opened for editing.

14. EWfilespec/APPEND$ is now equivalent to EAfilespec$.

15. EBfilespec/READONLY$ is equivalent to ERfilespec$.

16. A warning message is now issued for an ER or EB command whenever the file is found in a directory other than the one specified (due to a /LIB or /SCAN switch).

17. The SUPERSEDING warning message will no longer be given if a /INPLACE switch is used with an EB or EW command. (The switch is ignored by an EW except for this purpose.)
    (I would also like to see a /NOSUPERSEDE switch that would cause an error rather than a warning if the file already exists.)

18. The 2EO command brings back several features from earlier versions of STEVENS TECO.

19. TECO now recognizes a core argument on a monitor run command, e.g. R TECO 5ØP .

2Ø. An S$ command which uses a previous search string will now remember the proper setting of the "exact search" flag if it had been implicitly set by the presence of case control constructs within the previous string.

21. Several bugs have been fixed having to do with line sequence numbers, EN command, backward bounded searches occurring after 'dot' and interaction of /INPLACE and /SCAN or /SYS switches.

Texas TECO also comes with many other goodies such as a 25 page manual, a 22 page complete command summary, and a large collection of macros collected by Clive, some of which are described below:

```
ACOM   Aligns comments in MACRO programs
CLERK  Creates repeated blocks of text
COMENT Adds comments in MACRO programs
DATE Returns date in Q-register Ø
DECIDE performs text substitutions with user interaction
DOW    Returns day of week
FORCOM Adds comments to a FORTRAN program
GETNAM Returns user's name
GIVMFC Displays amount of monitor free core in use
HELP   On-line help system
LINE   Returns current line number
PGMFMT Formats MACRO programs
PPN    Returns job's PPN in Q-register Ø
RUNCON Generates table of contents for RUNOFF output
SIGl   Computes PI (from Moby Munger issue #1)
SYSTEC Types system status
TIME   Returns time in Q-register Ø
UNDER  Makes underscored text readable on a CRT
```

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MR2-3/E55
MARLBORO, MASSACHUSETTS 01752

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to guarantee continuing receipt of DECUS literature. Allow up to six weeks for change to take effect.

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: _____

Name: _____

Company: _____

Address: _____

_____

State/Country: _____

Zip/Postal Code: _____

Mail to: DECUS - ATT: Membership
One Iron Way, MR2-3
Marlboro, Massachusetts 01752 USA

Affix mailing label here. If label is not available, print old address here. Include name of installation, company, university, etc.