change tasks. The solution to this awkward method was to transfer the entire procedure to within the editor. Hence the editing environment, used the most in the production of a TeX document, becomes the environment that controls the entire process.

Once the basic sequence of producing, previewing, and printing a document was simplified to a few keystrokes, more sophistication was soon desired. The following is a partial list of significant features the authors believed important enough to include in the initial editing macro package.

- capability of inputting various kinds of information easily and efficiently
- instantaneous graphical display of fonts and font information
- a complete context-sensitive help system for TeX and the editor
- automatic text reformatting and TeX control code modification
- representing TeX macros as modified ASCII characters

2. Editing Environment

The text editor of choice is KEDIT, the PC version of the IBM mainframe editor XEDIT. KEDIT is an extremely powerful and versatile editor. Together with the procedural language REXX, practically any task can be simplified to the touch of a key. The principle advantage of KEDIT, because it is programmable, is that it can emulate most text editors (*not* word processors). Users of this interface will not need to learn a new editor — a fate on par with a root canal gone awry. KEDIT can be made inanely simple, such as EDLIN, or as sophisticated and complex as the user desires. Thus, the TeX interface is completely uncoupled from the editing process.

There are several features within KEDIT that enable it to be so versatile. It allows the user to create synonyms, such that any command can be called by a different name. For example, the term "translate" could be substituted for the command `move`, if that term was more comfortable to use. It can also be abbreviated to any length desired. Using the same example, the "translate" synonym could be specified as "tr", "tran", or "transl". The ability to define macros and assign them to almost any key- or user-defined command name is what makes KEDIT unarguably superior to non-programmable text editors. These macros can be simple functions used to save key strokes for frequently used commands or a technique to avoid having to go to the editor's command line or to DOS to perform a certain task. The macros are also able to call other macros, such as the TeX interface, which can all be accessed by hitting a single key.

A rudimentary example of redefining keys in KEDIT is the authors' modification of the opening and closing curly brace and square bracket keys. Whenever the user is in the editing environment, an opening square bracket `[` will return a `{` character. Likewise, a closing square bracket `]` will return a `}` character. This is useful not only for TeX, where square brackets are not generally used as control characters,[1] but also for **C** programming. To eliminate confusion, one of the authors has actually switched the keys on his keyboard to signify this modification.

Mansfield Software Group's Personal REXX is an easy to understand yet powerful procedural language written specifically for the IBM-compatible personal computer.[2] Besides an extensive array of commands for file handling, text manipulation, and parsing, the one feature that is primarily used in Personal REXX is windowing. This greatly simplifies the TeX process by using windows to display various types of helpful information or to control various options.

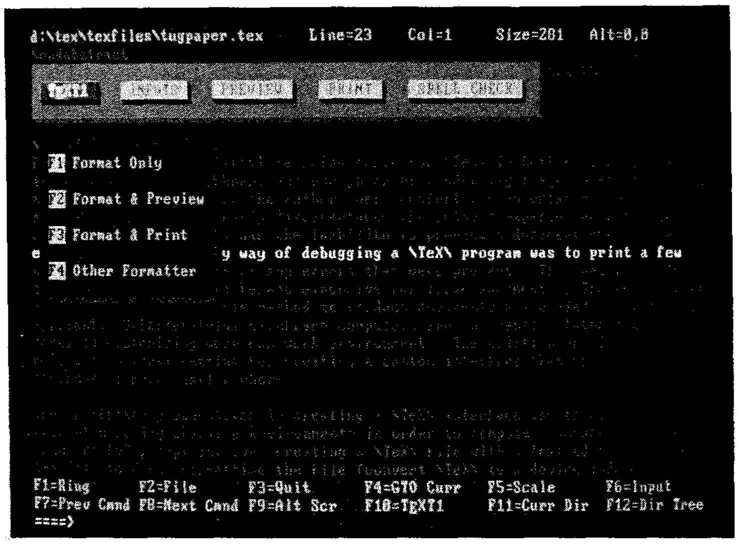
3. TeX Interface

A single keystroke invokes the TeX interface. Presently, this is reserved for the `F10` key. When this key is hit, two windows will be displayed, as shown in Figure 1.

The top window has five categories of control: **Format**, **Inputs**, **Preview**, **Print**, and **Spell Check**. The particular TeX formatter (e.g. *TeXT1*, TeX, LaTeX, μTeX. . .) which can be toggled by hitting the `F4` key, will appear in the highlighted box in the upper left; in this case, the formatter is *TeXT1*. The left window shows the function key options for format control, which are self-explanatory.

[1] Note that the authors are referring only to TeX, and not LaTeX, where the square bracket is of course as crucial as the curly brace –Ed.

[2] REXX is also available on several other small computer platforms; for descriptions, see Kubik (1989) or Tokicki (1988), both on Amiga REXX.



```
d:\tex\texfiles\tugpaper.tex     Line=23    Col=1     Size=281   Alt=0,0

 F1 Format Only
 F2 Format & Preview
 F3 Format & Print
e                    y way of debugging a \TeX\ program was to print a few
 F4 Other Formatter

F1=Ring      F2=File      F3=Quit      F4=GTO Curr   F5=Scale      F6=Input
F7=Prev Cmnd F8=Next Cmnd F9=Alt Scr   F10=TEXT1     F11=Curr Dir  F12=Dir Tree
====>
```

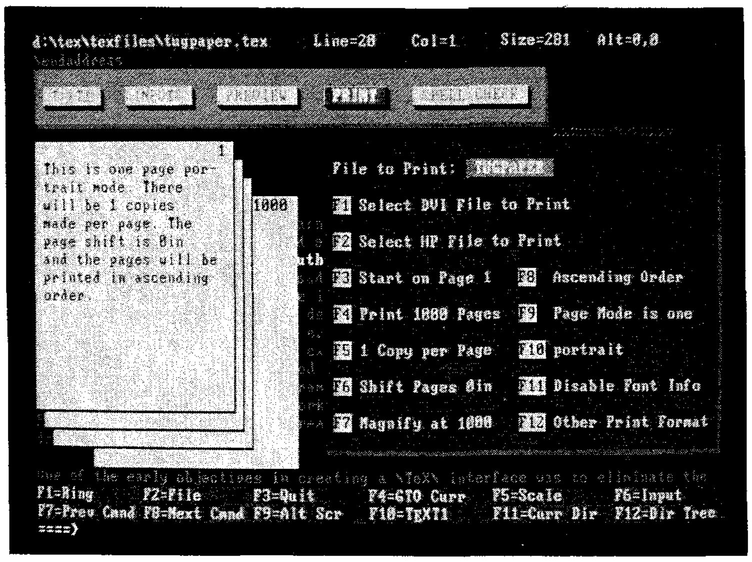
Figure 1: Screen display of the invoked TEX interface

Control is cursor-selected using the left and right arrow keys. Hitting the right arrow key will highlight the **Inputs** control option and a new lower window displaying six new function key options will appear. The function keys F1 through F6 are used for inputting *TEXT1* blocks, *TEXT1* models, font sets, math sets, specific fonts, and tables, respectively. This control option will be discussed in more detail in the next section.

The two control options **Preview** and **Print** are similar in function. The F1 key will list those files in a window that have already been formatted (dvi files) and can be cursor-selected to preview or print. The F2 key for the print control option will list those files that have previously been prepared for printing (for the authors' systems, *.hp files). The remaining function keys modify how the output will be presented. Specific to the print control option, an additional window display is located in the lower left that provides a general perspective of how the printer output will appear. The starting and ending page of the document will be shown in the upper right corners of the "pages". The orientation is also clearly displayed as being either portrait (right-side-up) or landscape (sideways). The short paragraph written on the starting page is a summary of options that cannot be easily shown in text-mode. These options include the number of copies per page, the margin offset for odd and even pages, magnification of the print, and whether font information will be echoed prior to printing. Figure 2 shows on example of the print controller option.

The options used for a particular printout can be saved to file. For sets of options that are used frequently, these files can be quickly retrieved by hitting the F12 key and cursor-selecting the file that contains the desired printing options. Once selected, the short paragraph displayed on the starting page will reflect a summary of the new options.

The last control option, **Spell Check**, loads a spell checking utility into memory when the F1 key is hit. The authors have installed Webster's New World Spelling Checker on their systems. Any spell checker, however, can be used to suit the taste of a particular user. The dictionary option, which is activated by hitting the F2 key, will open a window that lists several auxiliary dictionaries that may be cursor-selected and added to the standard dictionary. This is necessary for files such as TEX, that contain numerous commands or markup which would not normally be accepted by a standard dictionary. The auxiliary dictionary may have a list of "incorrectly" spelled commands that the spell checker will assume to be correct and thereby speed up the process. Under the **Spell Check** window



```
d:\tex\texfiles\tugpaper.tex     Line=28    Col=1     Size=281    Alt=0,0

File to Print: TUGPAPER

1000    F1 Select DVI File to Print
        F2 Select HP File to Print
        F3 Start on Page 1       F8  Ascending Order
        F4 Print 1000 Pages      F9  Page Mode is one
        F5 1 Copy per Page       F10 portrait
        F6 Shift Pages 0in       F11 Disable Font Info
        F7 Magnify at 1000       F12 Other Print Format

This is one page portrait mode. There will be 1 copies made per page. The
page shift is 0in and the pages will be printed in ascending order.

One of the early objectives in creating a \TeX\ interface was to eliminate the
F1=Ring      F2=File       F3=Quit      F4=GTO Curr   F5=Scale      F6=Input
F7=Prev Cmnd F8=Next Cmnd  F9=Alt Scr   F10=TEXT1     F11=Curr Dir  F12=Dir Tree
====>
```

Figure 2: Screen display of the print controller option for the TeX interface

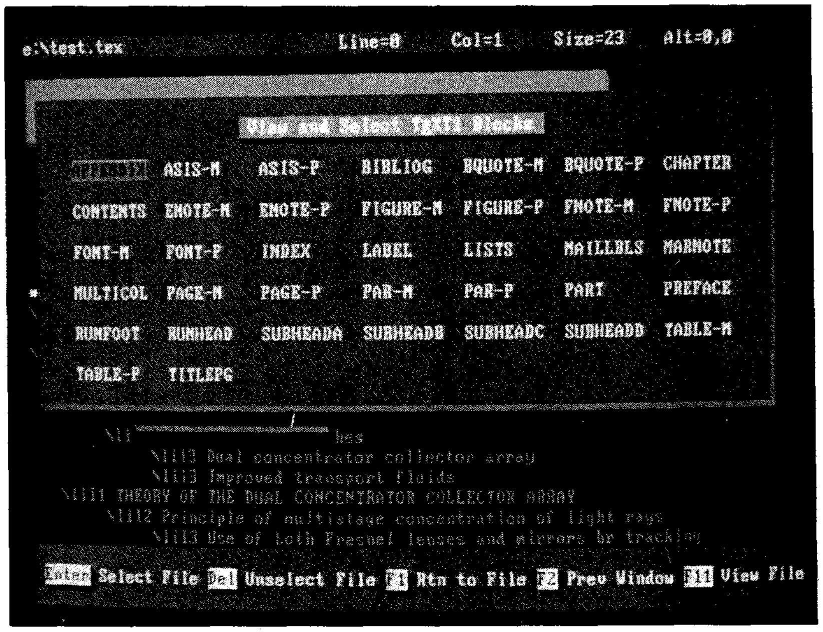
appear several other utilities to "aide" in the writing process. A dictionary and thesaurus may be loaded. Also, RightSoft's RightWriter may be invoked on the TeX source file.

4. Inputting Information

The authors learned TeX and how to use the *TEXT1* macros at Washington State University (Pullman, WA), where the *TEXT1* macros were developed. The TeX interface at WSU (on an IBM mainframe) was what the authors first modeled their original interface after. The *TEXT1* macros have the somewhat unique ability to change the global format from within the TeX source file, see *TEXT1* (1987) or Riley (1989). This can best be accomplished by loading basic document control files (blocks) from disk into the TeX file. The document control blocks are simple ASCII files that contain the necessary *TEXT1* markup to alter particular formats. This allows document processing to be greatly simplified. From within the editor, it is possible to open a window, display the 37 document component blocks (by name) and cursor-select any formatting block that will automatically be loaded into the file being edited. Figure 3 shows the screen after the window has been opened, displaying the *TEXT1* document control blocks.

For example, if a document is being created with a non-standard paper size and margin widths, loading the `page.blk` block into the current file (`page-p` in Figure 3)

```
% Default page dimensions and margins
\pageformat{\pagelength{11in}    % 792pt = 11in
            \pagewidth{8.5in}    % 612pt = 8.5in
            \topmargin{1in}      % 72pt = 1in
            \bottommargin{1in}
            \leftmargin{1.2in}   % 86pt = 1.2in
            \rightmargin{1in}
            \bindingadjust{0in}
}% end pageformat
\normalbottom                    % text height will be the same for each
                                 % page. Bottom lines will be even.
```



```
e:\test.tex                 Line=0    Col=1    Size=23    Alt=0,0

                     View and Select TEXT1 Blocks

  [illegible] ASIS-M   ASIS-P   BIBLIOG  BQUOTE-M BQUOTE-P CHAPTER
  CONTENTS ENOTE-M  ENOTE-P  FIGURE-M FIGURE-P FNOTE-M  FNOTE-P
  FONT-M   FONT-P   INDEX    LABEL    LISTS    MAILLBLS MARNOTE
* MULTICOL PAGE-M   PAGE-P   PAR-M    PAR-P    PART     PREFACE
  RUNFOOT  RUNHEAD  SUBHEADA SUBHEADB SUBHEADC SUBHEADD TABLE-M
  TABLE-P  TITLEPG

     \ii                  hes
          \iii3 Dual concentrator collector array
          \iii3 Improved transport fluids
  \iii1 THEORY OF THE DUAL CONCENTRATOR COLLECTOR ARRAY
      \iii2 Principle of multistage concentration of light rays
          \iii3 Use of both Fresnel lenses and mirrors by tracking

Enter Select File Del Unselect File F1 Rtn to File F2 Prev Window F11 View File
```

Figure 3: Sample window display of *TEXT1* control blocks that may be selected for insertion

and changing the dimensions of interest (substituting values within the appropriate { } braces), will easily adjust the page format. Markup does not have to be remembered, nor syntax for typing the markup, and time is saved with respect to loading the same file from DOS. The ability to also include document models or style sheets at the touch of a key is equally simple. It is this process that makes creating TEX documents a much easier task in comparison to manual insertion of TEX control codes.

The window interface allows the user to display and easily select any font available to TEX. This is accomplished by displaying all files with an extension of `*.tfm` into a window (a scrollable window since there are usually so many files) and then cursor-selecting as many of the font names that are necessary. The macro will read the name, perform a decimal-to-roman numeral conversion on the font size within the file name, and then insert a line into the TEX file that correctly loads the font for TEX. For example, to load the font `cmssi17`, a window is displayed with all the font names, much like Figure 3, and the user cursor-selects the `cmssi17` font by hitting the `Enter` key. Upon leaving the window, the current line in the editor will have the following line added after it: `\font\cmssixvii = cmssi17 at 17pt` (this can be seen in the second line of Figure 5). This not only saves time and frustration looking up what fonts are available, but also introduces a consistent font nomenclature. The authors also like to keep the following types of files located in this input window environment:

- *TEXT1* blocks
- *TEXT1* models/style sheets
- *TEXT1* font sets
- *TEXT1* math sets
- multiple ruled and aligned table formats
- graphical input files (e.g., clip art and scanned images)
- TEX Font Metric files (`tfm` files)
  - 75 standard Computer Modern (CM) fonts
  - 63 *TEXT1* fonts
  - resident and cartridge printer fonts
  - down-loadable soft fonts
  - PostScript fonts

Whenever the highlighted section or "cursor" is located on a file name that can be selected for input to the current file, hitting the `F11` key will open another window that contains the contents of that file for immediate viewing purposes. This feature is used throughout the interface as well as other editing utilities.

4.1 Graphical Help Facility

The ability to easily display font files in a window for automatic selection led to the desire to have a graphical display of the fonts. This would give the user an idea of what the fonts would look like on an output device without having to use a previewing program or a sample printout. It could also be used to show what default sizes and magnifications were available, and how to invoke them. Two commercially available software packages, ZSoft's Publisher's Paintbrush and PCX Programmer's Toolkit, were used to create this graphical font help system.

The graphics format used is ZSoft's `pcx` format, a pseudo-standard in the PC arena for graphics. The latest version of ZSoft's Paintbrush package includes a utility called `hp2pcx.exe`. It converts files (both graphics *and* text) produced for the HP laser printer to the `pcx` format. Thus, it is a trivial process to convert TEX-generated laser printer output to the `pcx` format.[3] Another utility, called PCX Programmer's Toolkit from Genus Microcomputing, is needed however, to quickly display the graphical file within the text-mode editing environment. The tool kit contains several useful PCX utilities; one will take a group of `pcx` files and load them into a library and another will instantaneously display the `pcx` file to the screen from a library.[4] This provides sample font files for all of the TEX fonts to be stored in one common library and displayed at the touch of a key from within the editing environment.

The process of generating the standard 75 TEX font files (plus as many as needed for specific resident printer fonts, soft fonts and the like) was simplified by creating a database of the font names, sample output text, and the sizes of available fonts. TEX could then produce the entire set automatically. This font database was formatted with *TEXT1* using one driver file that contained the necessary `\halign` commands and markup to produce the `*.dvi` files. A DOS batch file would convert all the `*.dvi` files (using `\magnification=473`) to a temporary `*.hp` file (HP LaserJet format, 300 dpi) and then convert the files to the `pcx` format. The files are loaded into a library and are available for display within the editor. Figure 4 is an example of a graphical font display file.

Besides the graphical font display, the authors have found that other information is easier to comprehend by means of graphical output rather than by using just standard descriptive ASCII text. For example, when loading font sets with *TEXT1*, a graphical display file of the particular family shows which faces are available. Also, general TEX help files are available in graphics format to illustrate quote marks, dashes, special characters, and ruled tables.

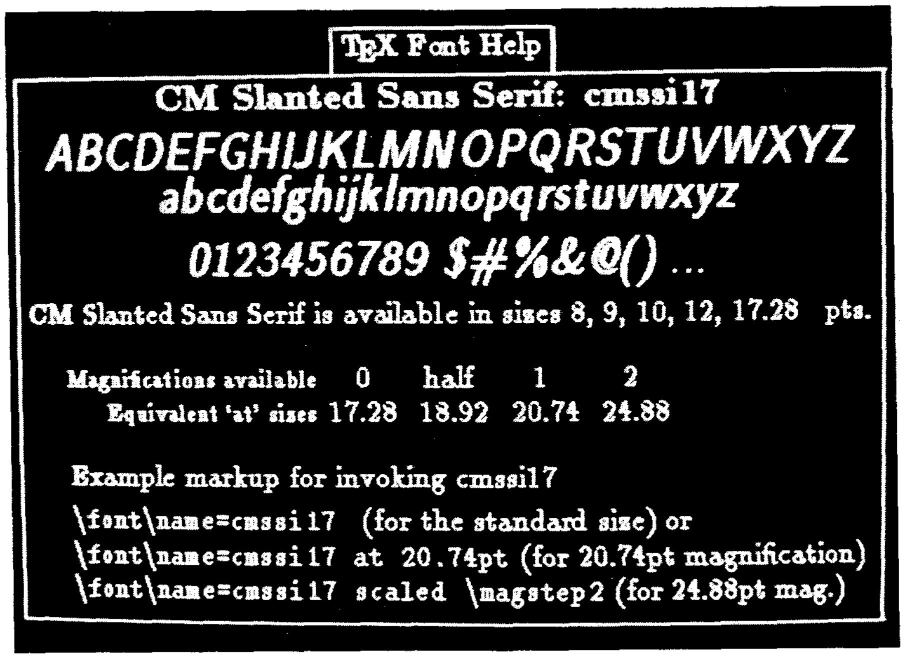
5. General Help Facilities

Several help facilities have been written to assist the user not only with the TEX interface but also within the editor and KEDIT/REXX macros. For each screen or window, the function keys (`F1--F12`) are usually displayed along the bottom of the screen (see Figure 1) with an abbreviated word or phrase describing their function. Complete and separate help menus for key combinations involving the `Alt` and `Ctrl` keys will be displayed by hitting the key combinations `Alt-h` and `Ctrl-h`, respectively. Particular to TEX files (any file with an extension of `*.tex`), `Ctrl-\` will determine if the cursor is on a TEX command; if so, an ASCII help file describing that command will be shown in a window that can be scrolled forward and backward using the `PgDn` and `PgUp` keys. When the `Alt-\` combination is hit, a listing of TEX commands, their correct abbreviations, and short synopsis will be shown in a window. The same help information as for the `Ctrl-\` key combination can then be accessed by hitting the **Enter** key when the highlighted line is located on a particular command.

---

[3] That is, it is the TEX output, consisting of the **METAFONT**-produced down-loaded *.pk font files in HP's pcl format, that gets converted to the Paintbrush *.pcx format. This process is *not* available with most graphics conversion utilities, e.g. Hijaak or IMSI's Graphics Transformer. However, hp2pcx has never disappointed the authors.

[4] The utility supports standard graphics cards (CGA, EGA, VGA, and Hercules), as well as Super VGA cards (800×600 modes provided by Tseng, Paradise, and Video Seven). The automatic display mode can easily be over-ridden to force an image onto the screen in any desired mode.


TeX Font Help

CM Slanted Sans Serif: cmssi17

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 $#%&@() ...

CM Slanted Sans Serif is available in sizes 8, 9, 10, 12, 17.28 pts.

Magnifications available	0	half	1	2
Equivalent 'at' sizes	17.28	18.92	20.74	24.88

Example markup for invoking cmssil7

```
\font\name=cmssi17 (for the standard size) or
\font\name=cmssi17 at 20.74pt (for 20.74pt magnification)
\font\name=cmssi17 scaled \magstep2 (for 24.88pt mag.)
```

Figure 4: Typical TeX graphical font help display file

5.1 Example of a REXX Editing Macro for *TeXT1* Conversion

There is a feature that eases the creation of TeX documents that can serve as an example illustrating the inter-relationship between the editor and REXX, the procedural language. Displaying the text on the monitor in a form that resembles the format after TeX formats the text is a desirable way to edit and view files. For example, a TeX file could have one line of text that will produce three lines of centered text when formatted and printed. It would be more readable to have three centered lines appear in the editor.

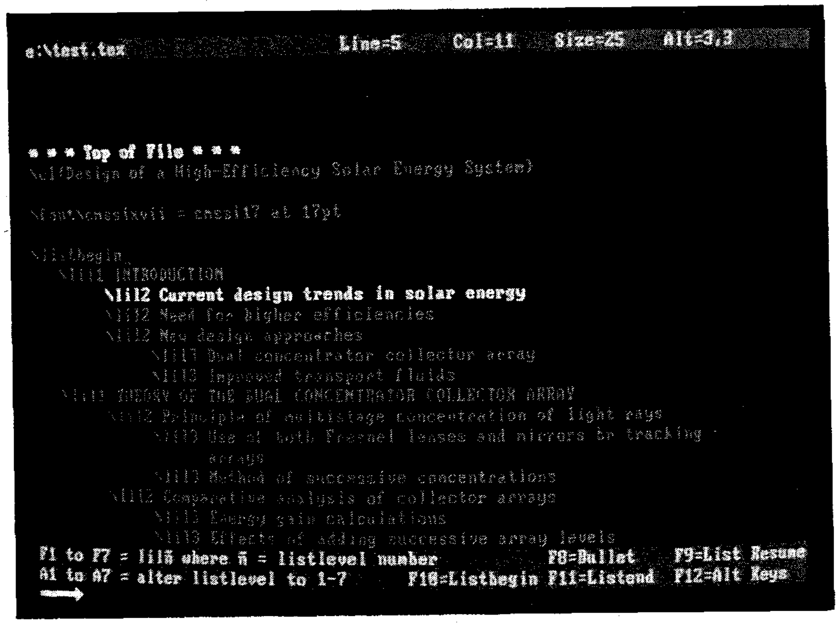
Taking this one step further, and using outlines as an example, it would be suitable to have text appear in outline format on the screen with text incrementally indented for each outline level. Also, for editing purposes, it would be advantageous to allow the particular outline level to be easily changed to another level. In one operation, the text will be reformatted for the screen display and the TeX control sequences that format the text will change to alter the final TeX output. This is performed within the editor by hitting one key, `Alt-L`, that loads the key definitions for list levels and displays the definitions along the bottom of the screen, as shown in Figure 5.

Keys `F1` through `F7` will create up to seven levels of outline lists, while `Alt-F1` through `Alt-F7` will, if the cursor is anywhere within an outline level, alter the current indent level. For example, if the outline had a section in the third indent level and the user wanted to alter this to a second level, then hitting `Alt-F2` would change `\lil3` to `\lil2`[5] and re-format the text with appropriate indentation. This feature allows quick and painless editing of list levels for *TeXT1*.

6. Problems and Idiosyncrasies

No system is without its flaws. There are some recommendations on the use of the TeX interface that significantly increase its performance. The way REXX is located in memory, applications should not be made resident while the user is in the KEDIT environment unless they are removed from memory before exiting the editing environment. The availability of LIM Expanded Memory Specification (EMS memory) alleviates the problem of overloading DOS with large macros or window information. To make the system run faster, the macros should be placed in a virtual disk. A significant amount of time is

[5] Here, `\lil3` and `\lil2` stand for `\listlevel3` and `\listlevel2`, the automated list macro for the *TeXT1* macros.



```
e:\test.tex                  Line=5    Col=11   Size=25   Alt=3,3



* * * Top of File * * *
\tl{Design of a High-Efficiency Solar Energy System}

\font\cmssixvii = cmssi17 at 17pt

\listbegin
   \lil1 INTRODUCTION
      \lil2 Current design trends in solar energy
      \lil2 Need for higher efficiencies
      \lil2 New design approaches
         \lil3 Dual concentrator collector array
         \lil3 Improved transport fluids
   \lil1 THEORY OF THE DUAL CONCENTRATOR COLLECTOR ARRAY
      \lil2 Principle of multistage concentration of light rays
         \lil3 Use of both Fresnel lenses and mirrors in tracking
               arrays
         \lil3 Method of successive concentrations
      \lil2 Comparative analysis of collector arrays
         \lil3 Energy gain calculations
         \lil3 Effects of adding successive array levels
F1 to F7 = lilñ where ñ = listlevel number          F8=Bullet    F9=List Resume
A1 to A7 = alter listlevel to 1-7        F10=Listbegin F11=Listend  F12=Alt Keys
⟶
```

Figure 5: Screen display of outline list file in list editing mode.

taken if macros have to be constantly read from the hard drive. The TEX interface has been designed assuming the user has an extended keyboard; the standard keyboard restricts a portion of the features of the interface from being applicable. At this time, there is no elegant accommodation for the standard keyboard.

From an aesthetic viewpoint, the standard PC graphics cards (e.g., VGA and EGA) include the option of setting the number of lines that are displayed on the screen. The authors prefer 28-line mode for several reasons. Many of the utility macros, although still functional in other line modes, simply look best in 28-line mode. Also, the authors have modified many of the default ASCII characters (those above decimal 128) to letters and shapes that are useful for display solely in this mode. These modified characters can be made for other line modes, but, although not a difficult task, it is very time consuming. The entire Greek alphabet, including upper-case letters, has been installed in a modified character set. These Greek letters, which represent simple ASCII numerals, are macros that will print their corresponding character. This "substitution" is useful for typing mathematical equations. Not only will the equation appear more representative of what will be printed, but will also shorten the typed length in the file, thus making it easier to read and debug. The authors have also added character shapes that permit two types of three-dimensional border effects and a descending capital E used in a text-mode TEX, are shown in Figure 1.

7. Portability

The practicality of the TEX interface would not be appreciated if it could not be easily transferred to other PCs with a wide variety of associated hardware and support software. The first attempt to copy the TEX interface to another computer proved to be awkward, because the authors had written into the macros several commands that were specific to the directory setup and hardware of the host computer. Some major modifications were immediately implemented. All paths and file identifications were removed from the macros and condensed into a single file, called `config.kex`, from which each macro then calls and retrieves particular information. Therefore, the only file that needs to be altered when copying the KEDIT and REXX files is `config.kex`, which will "personalize" the TEX interface for individual PCs. In addition, there are REXX functions located in several utility macros that are able to distinguish the hardware setup of the user's computer. For example, the REXX `pcfloppy` command will return the number of floppy disks available to the system. This command is useful for a PCTOOLS-type of utility that enables the user to easily move to and scan other directories.

8. Conclusions

The TEX interface has served the intended purpose the authors were originally trying to achieve: to speed up the creation and modification of TEX documents and to bypass the need to memorize markup. In the process of creating this interface, many additional macros, not necessarily related to TEX, were implemented to complement the editing software. There are a number of projects the authors feel would be extremely useful for the interface but have yet to be accomplished or finished.

1. The TEX formatter, Arbortxt's *μTEX*, when encountering an error, will not return the line number of the error back to the calling routine. It would be convenient to immediately return to the location of the error in the editing environment so that it can quickly be corrected. However, there is a roundabout solution to this problem. The error line number is written to the `*.log` file that is created when the TEX file is formatted. It is possible to read from this `log` file and retrieve the line number, but this method is exceedingly inefficient. The authors will wait for the next version of μTEX to see if this problem is addressed. Hopefully, by means of setting an environment variable.
2. An example of the auto-reformatting of text has been presented in this paper for *TEXT1*'s list level markup. Similar auto-reformatting will include block quotes, labels, centerlines, hanging paragraphs, justification (right and left), and subheadings.
3. Many of the *TEXT1* font sets are incomplete. For example, a majority of the font sets omit the bold italic font. In the future, all the font sets will contain the six standard text faces: roman, **bold**, *italic*, `typewriter`, SMALL CAPS, and ***bold italic***. These extra faces will be created using PC-**METAFONT**.
4. An interesting addition to the input control option of the interface, besides those already mentioned, would include clip art. The integration of graphic pictures and figures that could be cursor-selected and viewed (similar to the font sets) would be very useful. This addition is already in progress.

There is virtually no limit to the TEX interface, other than given that it is a text-mode only editing environment. The foundation has already been made; all new ideas are simply "tacked" onto the option windows and given new function keys to implement them.

Bibliography

Kubik, Kim. "AmigaTEX . . . or How Envy Was Resisted and Knowledge Found on the Road to Ööç." *TUGboat* 10:65–67, 1989.

Riley, Don L. *Using TEXT1: A Set of TEX Macros at Sandia.* Livermore, CA: Sandia National Laboratories, SAND89-8238, 1989.

Rokicki, Tomas. "The Commodore Amiga: A Magic TEX Machine." *TUGboat* 9:40–41, 1988.

TEXT1 : Reference Manual. Computing Service Center, Washington State University, 1987.