TOTAL ENGINEERING SERVICES TEAM, INC.

SOFTWARE DOCUMENTATION

STANDARD TEXT EDITOR

Document No. 1100-01

P.O. Drawer 1760
671 Whitney Ave.
Gretna, La. 70054
(504) 368-6792

Contact Arthur Zatarain, PE
via www.artzat.com
For information on this document

TOTAL ENGINEERING SERVICES TEAM, INC.
SOFTWARE DOCUMENTATION
STANDARD TEXT EDITOR

1.0 TEXT FILE BASICS

Many TEST INC. programs use simple text files for many setup and data file operations. For example, both the RTU/SCADA system and the Production Allocation Program (PAP) systems use the same basic editor program. This document serves as a basic introduction to text editing as well as a reference manual on the editor operation. It is helpful for the operator to have a basic understanding of DOS text files and how they can be edited. This section will explain text file concepts, and will provide specific information on operation of TEST's internal text editor.

A text file is simply a DOS disk file that contains only text characters. Text characters are the normal printing characters like the letters A, B, and C. Text file also contain special control characters that do not appear on the screen but serve to control the layout of the text. These control characters include the Carriage Return, Line Feed, Form Feed, Tab, and other formatting characters. In any case, the text file stores these characters in a sequential manner in the same order that the characters are entered into the file. This causes the file to have the appearance of a typewritten document, which is basically how the computer (and the TEST computer program) treats the file.

By contrast, there are many other disk file types available for use on the computer. These are often called Binary files, because they contain information that only makes sense to the computer. They contain encoded digital information rather than simple text information. These files are used to hold programs as well as data, and can only be used in certain specific instances. Text files, on the other hand, are almost universally accepted by any program that can process documents. These programs could include editors, word processors, spreadsheets, and program that you are now using itself.

One tradeoff of text files is that they may take more space on the computer to store the same amount of information. A computer takes 5 storage locations (bytes) to store the text string "12345", but it would only require 2 bytes to hold this as digital information. In fact, the computer would probably have to convert from the text to the digital format before the number could be used for a calculation. Therefore, it would appear that it is better to store information in a binary format rather than in a text format.

In many cases this is true. In the case of TEST's programs, and other information related programs, it is often easier to design the program to operate in text mode rather than in binary mode. This is because the interaction of a person with the data represents most of the processing time. The actual calculation time is fairly small when compared to the data entry, viewing, and printing time. TEST's programs use this text approach so that the data is always easily available for viewing and modification by a person. Also, any editor (in addition to this internal editor) can be used on the data files. This allows operators to work on the data using any available editing program, now and in the future, because the data is stored in a simple text format.

The text information is normally entered into the computer's memory via the keyboard. As the operator types, each keystroke is stored in the next available memory location. This would be very similar to filling out a piece of paper that is printed with a grid. Each character takes the same amount of memory space (one

byte), and the characters are lined up one after another in memory. As the operator types, the computer's memory is filled up with these text characters. Remember that printing as well as control characters are stored, and that they each take one byte of space. If any characters are inserted into existing text, the characters located after the insertion are moved further down in memory to make room for the new characters. Character deletions have the opposite affect.

Eventually, the characters in memory are written out to the computer's disk to be permanently stored in a DOS file. The file will have a unique name, and will contain all of the characters (printing and non-printing) that were entered by the operator. This file can be used in many ways. The obvious use in TEST's programs is to provide setup and control operations for the various program using the editor. Another use is to have the file reloaded so that is can be modified by the same editor that created it. The file can also be printed using a variety of methods. The file can be copied to a backup disk for safe keeping of for use on another computer.

The simple text nature of the file makes it easy to use in many computer applications. The rest of this section will provide information on operation of the built-in editor.

2.0 STARTING THE EDITOR

2.1 EDITOR SELECTION

The editor is started by selecting one of the options under "EDIT" on the main program horizontal menu bar. All of these edit options eventually end up operating the same editor. The only difference in each choice is the default DOS file name presented to the operator. These options exist because normally the program can "guess" what file you want to edit based on your menu choice. For example, if you select "notebook" in PAP or RTU files in the SCADA program, the computer can use the current information to pre-select a file name for you. You can always override the computer's selection, so there is no harm in letting the system select a file name for you.

2.2 FILE SELECTION

If you want the name that is suggested, simply hit [ENTER]. The specified file will be loaded and the editor is started. The editor always asks you to confirm the file name, and allows you to override the default selection in two ways. If you override the default name (by simply typing in the new name), then the editor will load your file instead. If you override the default with an ambiguous file name, then the editor presents a "pick list" of files that match your file specification. For example, if you entered "*.dat" as the file spec, then the picklist would contain all files that end in "DAT". If you are not familiar with ambiguous and unambiguous file names, refer to your DOS documentation for a thorough discussion.

Note that the file name prompt can be edited in a manner similar to text in the editor itself. This means you can move the cursor to the character to change

rather than retyping the entire name. Many of the normal editing functions are active while editing the file name field.

If you use a file name containing a wildcard character (such as "*.DAT"), a pick list is generated that indicates all files meeting the specification. Use the arrow keys to move the selection bar onto the desired file, and select with the [ENTER] key. If you have a mouse, you can use it instead by picking the file with the left button. If you press escape to exit the pick list (or the file name prompt), then the entire edit process is terminated.

When the editor starts, the entire text file is loaded into memory and the editor is set up to use the entire screen. Depending on the number of lines selected for the current CRT configuration , the editor shows 25, 43, or 50 lines of text on the screen. This includes the heading and other information about the editor at the screen's upper section. This information includes the file name, the number of bytes (characters) used, the column and line number, and various editor settings. This heading is constantly updated during the editing process.

2.3 EDITOR HEADING LINE

Within the editor, there is a top heading section that tells you several things about the state of the program and the file being edited. It contains flags for insert/overtype mode, indent mode, and other user controlled settings. It also shows the current file name as well as the line number and column number for the current cursor position. Any information prompts or error indicators used by the editor will also be placed in this top section of the screen.

3.0 EDITOR OPERATION

3.1 BASIC FUNCTIONS

Basic operation of the editor involves only a few keystrokes. For the beginner, this is fine. As the operator gets more familiar with editor operation, the more advanced functions can be used. The editor in the TEST program uses the keystroke sequences established by standard PC programs such as Dbase, Sidekick, Wordstar, Borland Products, and other editing type programs. Once you learn these codes, you can use them in many other programs. If you are already familiar with them, then you are in good shape. If not, then take a few minutes to learn the basics before you try any fancy stuff. First, lets get familiar with the basic editor commands and functions.

Some of the functions can be performed by more than one key sequence. This is a carryover from the old days (1970's) when there were no standard PC keyboard designs. Many functions can be done using control key sequences rather than special PC keys like PgUp and PgDn. The end result is the same, and you can use either method at any time. The PC keys are easier to remember when you first start, while the old style control key sequences are faster after you become familiar with the programs. Suit yourself!

Normally, when we discuss control sequences below, we will note the

keystrokes individually within brackets like this - <^K>. The keystroke noted here is a control-K. The up caret ^ means press and hold the control key while pressing the indicated key. All keys within the brackets <> represent a single key stroke. So, the sequence <PgUP><LEFT> represents two keystrokes... the page up key followed by the left arrow key. The sequence <^K><^B> represents two keystrokes, a control K and a control B. Get familiar with this type of notation. It shows up here as well as in many other computer related documents.

The most basic function is to simply type characters into the computer. Each keystroke is entered into memory and appears on the screen just as if the user were typing on a typewriter. This is actually what is done most of the time when using the editor. Just type away! The less obvious keystrokes occur when the current cursor position must be moved, when text must be inserted, or certain words must be rapidly located. Each operation builds on previous ones, so take your time and learn them in a logical sequence.

It is difficult to explain the entire editor system because it is something that has to be learned "hands-on". It is very important to understand the basic operation of what is going on. The entry of text characters into the memory buffer and then into a disk file is the most basic principle. When the file contents are in the buffer, the user can drive through the file using the arrow keys (and other keys) to position the cursor. Any text entered at the keyboard goes into the file at the cursor position. This is the basic operation of any editor, including the TEST editor. Basic operation can involve only these control keys:

Arrows	- To move around the buffer.
PgUp/PgDn	- Flip one screen at a time.
Insert	- To switch from overtype to insert mode.
Delete	- Erase characters to right of cursor.
BackSpace	- Erase characters to left of the cursor.
F2	- Save buffer to the disk file.
ESC	- Quit edit and discard changes.

3.2 CURSOR MOVEMENT

Cursor movements are fairly easy to use. The PC keyboard has arrow keys that are used to move the current cursor in 4 directions, one character at a time. The movement can be accelerated by holding the control key while pressing the arrows. This causes movements to be one word at a time rather than one character. The Page Up and Down (<PgUP> and <PgDN>) are used to flip whole screens.

3.3 DELETE AND INSERT

The delete key <DEL> is used to remove characters to the right of the current cursor position. The backspace key <BKSP> (big arrow pointing to the left) removes characters to the left. The insert key <INS> switches the editor in and out of "insert mode", where typed text is added to what is already on screen. In "overtype" mode, the newly typed characters replace those on the screen. Insert mode causes existing characters to move over and make room for the new ones. The insert key acts as a "toggle", where each keypress changes the mode

back and forth.

3.4 BLOCK MARKING

Text editors can manipulate a block of text by letting the user mark the block and then perform the operations. Typical functions include block copy, move, and erase. All block functions start with the same procedure, while the final result may require slightly different steps. Mark the beginning of the block with F7, and the end with F8.

Some block commands require two keystrokes for each step. These block operations start with <^K> (hold down the control key while pressing K). The next key just after the <^K> is the block function. To mark a block, place the cursor at the starting position and press <^K><^B>. The end of the block is similarly marked with the sequence <^K><^K>. This works the same as the F7 and F8 keys mentioned above. This seems strange, but can be easily remembered by looking at the word **B**loc**K**. It starts with a **B**, and ends with a **K**, and so does the marking sequence. Start with <^K><^**B**>, and end with <^K><^K>. All block commands start with <^K> and are followed by an additional control code. A marked block will appear in a different color than the unmarked text so that the blocked area can be verified.

3.5 BLOCK PROCEDURES

After marking a block, the operator can do a variety of things with it. If the procedure involves moving or copying the block, than the cursor must be moved to the new location before going to the next step. If the operation is not cursor position related, then the cursor can be left where it is. For example, to move the block, reposition the cursor and enter <^K><^V>. So, <^V> is the block command code for move. These will allow you to "block" off areas of the buffer so they can be copied, moved, printed, or deleted. The steps here are always the same: Mark the beginning of the block, then end of the block, move the cursor (if necessary), then do the operation. Always do the steps in that order and you will always get the expected results. Some of these operations actually require more than one keystroke in sequence, such as ^K^P to print a block,. This means press control-K then control-P. Some of the block keys are as follows:

	PRIMARY	SECONDARY
Save file under new name	<^K><^N>	
Save file and exit	<^K><^X>	
Abandon file	<^K><^Q>	<Esc>
Save File & Continue	<^K><^D>	<F2>
Mark start of block	<^K><^B>	<F7>
Mark end of block	<^K><^K>	<F8>
Mark current word as block	<^K><^T>	
Toggle block display	<^K><^H>	
Copy block	<^K><^C>	
Delete block	<^K><^Y>	
Move block	<^K><^V>	
Read block	<^K><^R>	
Write block	<^K><^W>	

Print block	<^K><^P>
Indent marked block	<^K><^I>
Unindent marked block	<^K><^U>
Toggle marker display	<^K><^M>

3.6 PLACE MARKERS

Sometimes it is handy to set place markers in a file so that you can go off and look at another section and quickly return to the original spot. This can be done with 4 separate markers, each set with a two stroke sequence starting with control-K followed by a number 1 through 4. For example, pressing ^K2 sets place marker number 2. To go to a marked point in the file at any time, simply press control-Q followed by the number 1-4. To go to the spot marked above, the keystrokes required would be ^Q2.

3.7 TEXT SEARCH AND REPLACE

It is often handy to quickly locate a certain word or phrase, and possibly change the phrase to another one. The editor uses the standard <^Q> "quick" functions to perform these tasks. The find function is done with <^Q><^F> sequence. The editor will ask for the text to find, and will then allow search several options to be entered. These options are single letter codes that control the manner in which the search is made. The codes can be used together to tailor the way the editor views your text during the search. The codes are:

W	Whole Words Only
U	Ignore upper and lower case differences
G	Globally search entire document
N	Don't ask before changing text

So, specifying **UW** will cause the editor to search for your text regardless of the case, and to look for whole words only. If you had specified TO as the search phrase, then the editor would match only the word **TO**, but not the word **INTO** which of course contains the word TO. The editor searches for the next occurrence of the specified phrase, and stops with the cursor at the right place. If the phrase is not found, then the editor tells you so and leaves the cursor where it was. If a match is made, then the next occurrence of the same search can be done with <^L> (for Last search).

The Search and Replace function is an extension of the Find function, and the editor will request information as to the replacement phrase as well as the search phrase. The sequence here is <^Q><^A>. If you specify that the phrase "TOTAL B" is to be changed to "TOTAL A", and the options specified are "UGWN", then all occurrences of exactly "TOTAL A" will be changed to "TOTAL B" without any confirmation by the operator. If you deleted the letter N from the options, than each occurrence will require confirmation before the substitution is made.

4 TYPICAL QUESTIONS

4.1. HOW CAN I PRINT MY FILE?

Good question! First, you must understand that the text you have entered can either be in the memory buffer, or it can be stored in a disk file. To print from the memory buffer while editing, you can use the block print function. To print an entire file from outside the editor, you can use another editor, the DOS PRINT command, or some other printing utility. AS a quick answer, we will print from the editor with these steps:

1- Position the cursor at the beginning of the area to be printed.
2- Mark the block start with F7 or <^K><^B>. Either will work.
3- Move the cursor to the end of the area to be printed.
4- Mark the block end with F8 or <^K><^K>.
5- Print the marked area with <^K><^P>.
6- After printing, remove markings with <^K><^H> if desired.

4.2 HOW CAN I DUPLICATE A SECTION OF MY FILE?

Another good question. In programs that repeat text often, being able to copy a block is very handy and time saving. This operation is very similar to the print question above, which is another block operation. The steps to copy are as follows:

1- Position the cursor at the beginning of the area to be copied.
2- Mark the block start with F7 or <^K><^B>. Either will work.
3- Move the cursor to the end of the area to be copied.
4- Mark the block end with F8 or <^K><^K>.
5- Move the cursor to the start of the new copy of the text.
6- Press <^K><^C> to make a copy of the marked text.
7- After copying, remove markings with <^K><^H> if desired.

Do you see how these block operations are very similar?

4.3 WHEN I TYPE NEW STUFF IT OVERWRITES THE STUFF ALREADY THERE!

You have discovered overtype mode! This means that any text entered replaces the existing text, rather than moving it over. To switch to insert mode, simply press the <INSERT> key, or <^V>. Either will work the same.

4.4 HOW CAN I COPY A SECTION OF MY FILE TO A NEW FILE?

This is yet another block operation. You must mark the block, then tell the program to send the block to a new file. The steps are:

1- Position the cursor at the beginning of the area to be written.
2- Mark the block start with F7 or <^K><^B>. Either will work.
3- Move the cursor to the end of the area to be written.
4- Mark the block end with F8 or <^K><^K>.
5- Start the write by pressing <^K><^W>.
6- Tell the editor the name for the new file.
7- After writing, remove markings with <^K><^H> if desired.

4.5 HOW CAN I READ IN A DOS FILE INTO THE FILE BEING EDITED?

You can easily tell the editor to grab a file and stick it in the file being edited.

The entire file must be read in, and it goes into the edited file at the current cursor position. The steps here are :

1- Position the cursor where you want the text to be inserted.
2- Press <^K><^R> to start the block read.
3- Tell the editor the name of the file to read.
4- If the text is highlighted, press <^K>^<H> to turn this off.

4.6 **HOW CAN I COPY A BLOCK OF TEXT TO ANOTHER FILE?**

This is a little more tricky than the above two examples because it involves both of them. First, write the text to be moved to a temporary DOS file with any name you want, say "X.X" for example. This will be done with the block write function explained above. Then, quit the current edit and start editing the file to receive the block. Then, read in the temporary file using the block read function.

4.7 **I MADE CHANGES AND QUIT THE EDITOR, BUT THE CHANGES ARE NOT THERE. WHAT HAPPENED?**

It is important to understand that the text entered while editing a file are not permanent until the file is saved to a DOS file. This is done with the F2 key, or with <^K><^S>. If you have made changes, and do not save the file, then the file remains as it was before the edit. You can quit an edit without saving changes by pressing <ESC> rather than F2, and you may want to do this under certain circumstances. Accidentally quitting without saving is hard to do because the editor will prompt you before throwing away your changes, but it can happen if you are in a hurry and are not careful.

4.8 **I MADE CHANGES AND THEY ARE NOT WANT I WANTED. HOW CAN I GET MY OLD FILE BACK?**

Every time you save the edit file to disk, a backup copy of your file is made. This file has the same name as the original but has a file type of ".BAK". If you want to recover this file, exit to DOS and copy this backup file over on top of the current file. This effectively takes you one step back in your editing process. The file will be the version just before your last save operation.

5.0 ONLINE HELP SUMMARY

The editor can display online help by pressing the F1 key anytime during the edit process. A simple text file is displayed showing a keystroke summary as shown below:

```
Cursor up                       <Up>            <^E>
Cursor down                     <Down>          <^X>
Cursor left                     <Left>          <^S>
Cursor right                    <Right>         <^D>
Cursor to start of line         <Home>          <^Q><^S>
Cursor to end of line           <End>           <^Q><^D>
Cursor left one word            <^Left>         <^A>
Cursor right one word           <^Right>        <^F>
-------------------------------------------------------
Scroll up one page              <PgUp>          <^R>
Scroll down one page            <PgDn>          <^C>
Scroll window up                <^W>
Scroll window down              <^Z>
Top of file                     <^PgUp>         <^Q><^R>
End of file                     <^PgDn>         <^Q><^C>
Top of screen                   <^Home>         <^Q><^E>
Bottom of screen                <^End>          <^Q><^X>
Cursor to prev position         <^Q><^P>
Jump to line                    <^J><^L>
-------------------------------------------------------
New line                        <Enter>
Insert line                     <^N>
Toggle insert mode              <Ins>           <^V>
Insert control character        <^P>
Insert tab                      <Tab>
Restore line                    <^Q><^L>
-------------------------------------------------------
Delete char at cursor           <Del>           <^G>
Delete previous char            <BkSp>          <^BkSp>
Delete word                     <^T>
Delete line                     <^Y>
Delete to end of line           <^Q><^Y>
-------------------------------------------------------
Mouse select                    <ClkLeft>
Request help                    <F1>            <ClkBoth>
-------------------------------------------------------
Save file                       <F2>            <^K><^S>
Read new file                   <F3>
Save file under new name        <^K><^N>
Save file and exit              <^K><^X>
Abandon file                    <^K><^Q>        <Esc>
Save and switch files           <^K><^D>
-------------------------------------------------------
Search                          <^Q><^F>
Search and replace              <^Q><^A>
Search again                    <^L>
-------------------------------------------------------
```

Mark start of block	<F7>	<^K><^B>
Mark end of block	<F8>	<^K><^K>
Mark current word as block	<^K><^T>	
Toggle block display	<^K><^H>	
Jump to block begin	<^Q><^B>	
Jump to block end	<^Q><^K>	
Copy block	<^K><^C>	
Delete block	<^K><^Y>	
Move block	<^K><^V>	
Read block	<^K><^R>	
Write block	<^K><^W>	
Print block	<^K><^P>	
Change block to upper case	<^O><^U>	
Change block to lower case	<^O><^V>	
Toggle case of block	<^O><^O>	
Indent marked block	<^K><^I>	
Unindent marked block	<^K><^U>	
Set level for block indent	<^O><^B>	
Set marker 0	<^K><0>	
Set marker 1	<^K><1>	
Set marker 2	<^K><2>	
Set marker 3	<^K><3>	
Jump to marker 0	<^Q><0>	
Jump to marker 1	<^Q><1>	
Jump to marker 2	<^Q><2>	
Jump to marker 3	<^Q><3>	
Toggle marker display	<^K><^M>	
Reformat paragraph	<^B>	
Global reformat	<^K><^G>	
Center line	<^O><^C>	
Toggle smart/fixed tabs	<^O><^F>	
Set size of fixed tabs	<^O><^T>	
Set right margin	<^O><^R>	
Toggle indent mode	<^O><^I>	
Toggle word wrap	<^O><^W>	

End of document

AMZ/nt texted.doc