*Editor for Wang 2200 Assembly Language*

## 1.0 Introduction

The EDITOR is a primative program designed to edit raw text and feed the text to a contiguous disk file. This file may then be entered as a source file to the ASSEMBLE program, which in turn, will convert the text to object code format.

EDITOR was patterned after a common character oriented text editor, and permits short programs to be written and modified prior to assembly stage.

The output of the EDITOR program is a BA style file on the selected disk. Note that the EDITOR is full of limitations, and the user is pre-warned to allow the EDITOR to have as much room as possible for a partition size.

Though the EDITOR will work in almost any size partition, large files will not be handled correctly. However, for almost all the work that the author has done to date, a 51k partition is adequate for execution of fairly large programs.

It may well be advised that the structure of the program should be rewritten if one is to make use of this seminar. The disk structure is not one that is conducive to repetative editing of files.

Since this is a character oriented editor, the input of commands should only be done once the operation is fully understood. We hope that this does not intimidate the novice, but is placed here to help, not hinder program development.

## 2.0 Text Pointers

The nature of any character Editor is to always reference the current location relative to the start of the text in number of characters. All commands to the Editor assume that the user knows where they are in the text buffer.

There are a limited number of commands available to the user that will alter the character pointer (CP). These commands are listed as follows:

*CP altering Commands*

| | |
|---|---|
| *B* | *Beginning. Sets the CP to 1, which is the top of the text buffer, position 1. If insertions are done, the inserted text always precedes the CP.* |
| *Z* | *End. Sets the CP to the last position in the character buffer + 1. This allows text to be appended to the current buffer.* |
| *EZ* | *Clear. Removes all currently entered text in the text buffer, and sets the CP = 1.* |
| *nL* | *Line move. Moves over n of text in the buffer, forward or backward. A line of text is defined by the 0D code. Therefore, this command will count the number of 0D (CR codes) bytes to pass over. 23L would pass over 23 lines of code.* |
| *nM* | *Character Move. Moves over n , forward or backward. -5M would move the character pointer backwards. 1200M relocates the CP 1200 characters forward.* |

## *3.0 Status Reporting*

*With any character editor, it is imperative that one knows where he is within the Text buffer. Therefore, the following series of commands were enabled to report various position reports:*

| | |
|---|---|
| *H* | *Reports the amount of room left in the current text buffer.* |
| *:* | *Reports the current number being processed.* |
| *.* | *Reports the current CP.* |
| *nT* | *Types the contents of the text buffer from the current CP to n CR codes. 1T would print the current line.* |
| *nP* | *Prints the contents of the text buffer from the current CP to n CR codes on the local printer, device code 204.* |

## *4.0 Entering Text*

*New text may be entered into the text buffer by positioning the CP to the location to be inserted, the typing the I command. All text following the I command will be inserted into the text buffer preceding the CP. The CP will then be adjusted to the character position the newly entered text.*

*Each line of code must be preceded by the I code. Failure to do this will result in the editor assuming that text is really commands, and procede to execute this with sometimes disastrous results.*

*To append text to the end of the buffer, first enter the Z command to position the CP to the end of the buffer. Successive I commands will then insert text at the end.*

*Examples:*

| | | |
|---|---|---|
| *BEZ* | | *Clears text buffer* |
| *ITEST* | *SET R0 < 4* | *Insertes the string* |
| | | *TEST SET R0 < 4* |
| | | *at position 1. Note that a CR code was appended.* |
| *BIS/* | | *Inserts the character S in front of TEST.* |
| *B1T* | | *Would report:* |
| | | *STEST SET R0 < 4* |

## *5.0 Removing text from buffer*

*There are several ways to remove either characters or text from the buffer. These are:*

| | |
|---|---|
| *nK* | *Kill lines. Removes n from the current CP to n number of 0D (CR) codes. The CP is positioned after the last 0D code deleted.* |
| *nD* | *Delete Characters. Removes n from the current CP. The CP is positioned after the last character deleted.* |
| *nUtext/* | *Deletes characters from the current CP to the nth occurance of the text following the U command to the delimiter, '/'.* |

## 6.0 Changing text

There are several occasions where just changing the text is required. Generally, only small sections of data need to be changed. To change text, the C command is used.

Coldtext/newtext/

To change text, type C followed by the old text string, exactly, that one wishes to change. Follow the old text by the delimiter, defaulted to '/', and enter what the new text is supposed to be, again followed by the delimiter, '/'.

The text buffer is searched, starting at the current CP, for the first occurance of the old text string. When found, the text is first deleted, then the new text is inserted. When completed, the CP should point to the of the new text.

If one is daring, one might want to change the 2nd, 3rd etc. occurance of a string. nColdtext/newtext/ will bypass n-1 occurances of the the old text, and then change the nth occurance of the old text to the new text.

If the old text is not found, the CP remains unchanged, and the message 'No Matches Found' will be displayed.

## 7.0 Finding and Searching strings in text.

Two ways are implemented to find text strings within the text buffer. These two commands position the CP differently, so be sure which one you wish to use:

Ftext/ — Looks for the occurance of the text specified prior to the delimiter, and if found, positions the CP of the found text.

Stext/ — Looks for the occurance of the text specified prior to the delimiter, and if found, positions the CP the found text.