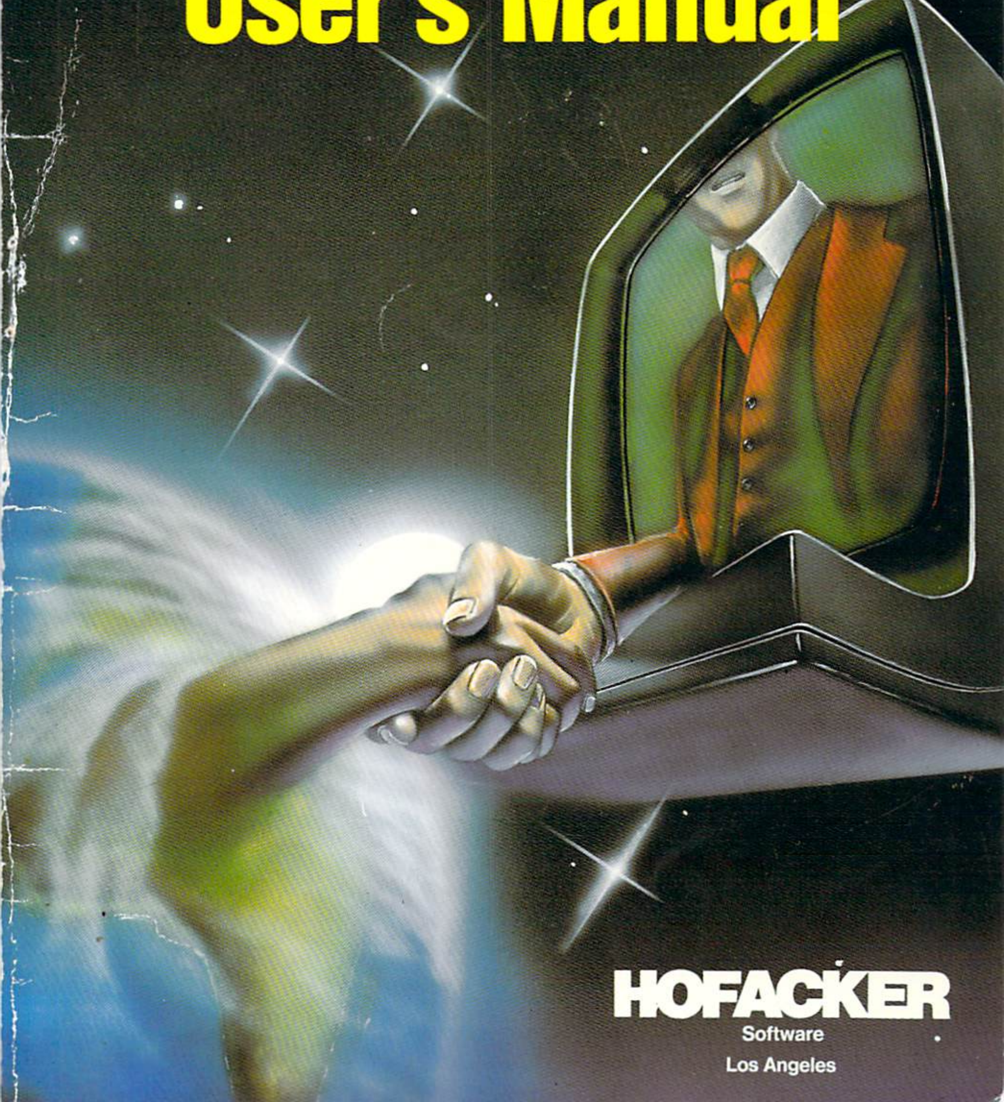
MACROFIRE

EDITOR/Assembler for the Commodore 64

HOFACKER
Software
Los Angeles

HOFACKER

Software User's Manual



HOFACKER
Software
Los Angeles

MACROFIRE - an editor assembler for the Commodore 64

USING THIS BOOK

Instructions can be dull, tedious and uninformative at times, with obscure technical language obfuscating the exposition (what?). In this book we will use explanations and examples that are easy to understand, and straight to the point. We will try to avoid the complications of programming terms, but if it becomes necessary to discuss something of a technical nature, we will attempt to simplify and explain everything each step of the way.

The best way to learn to use MACROFIRE is by trying the various commands as you read about them. You will discover the best way to do the things YOU want to do by using MACROFIRE and experimenting with different combinations of commands.

If the computer is "new" to you, then we recommend that you read the C-64 manuals that came with it before you begin using MACROFIRE. This will make you more familiar with the ins and outs of MACROFIRE as we discuss the various parts of the system.

THE EDITOR

NOTATION

One of the most frustrating and confusing things about a tutorial book for a computer (this goes for the writer as well as the reader!) is the notation used to indicate what you enter (or type) into the computer, what the computer prints out on the video screen, or how we indicate in a book, a sequence of keys to press or keys that must be pressed at the same time. Here are the special notations we have used in this description along with their explanations.

<A>
A letter or word enclosed in brackets indicates one of the keys on your C-64 keyboard. In this case, the "A" key should be pressed.

<CTRL>
This refers to the "control" key, located on the left side of the keyboard. In most cases, the <CTRL> key is used in combination with another key to tell MACROFIRE to execute a special function. The "-" sign will be used to indicate that two keys are to be pressed at the same time. For example:

<CTRL>-<A>
means that you should press the <CTRL> key, and, while still holding it down, you should press the <A> key so that both keys are being held down at the same time. Note that pressing the two keys in the reverse order will not produce the same result.

<RETURN>
This key is located on the right side of the main keyboard. This key causes the cursor to "return" to the left side of the screen. It is used mainly to terminate lines on the screen, but some command sequences require that a <RETURN> be pressed at the end. When you press this key, a "control-M" is

placed in the text. Normally, all control characters except "M" and "I" are visible on the screen. These two characters are not usually visible because they are used to terminate lines and generate blank spaces in the text, as we will discuss shortly. You can see these characters by using the <CTRL>-<H> command to turn the control character display on and off, which we will also discuss in a moment.

Spaces and other special characters:

A "character" is anything that appears on your ideo screen, and some things that don't, like the control-M and control-I that we just discussed. Even spaces, which are apparently blank, are actually characters as well. These are the different types of characters that can appear on screen:

...Letter characters - are "A" through "Z" in upper case. When MACROFIRE boots up, you are in the upper case mode. To lock into uppercase letters, you have to press the <SHIFT/LOCK> key. You can return to the upper/lower case mode by pressing the <SHIFT/LOCK> key once. Be careful when pressing the <C=> key or <SHIFT> in combination with an other key and watch out for careless usage of the function keys on the right side of the keyboard.

...Number characters - are the numerals "0" through "9", located across the top row of the keyboard.

...Symbol characters - are those that appear above the number keys (like "! ", "@", and so on), as well as the "+", "-", "=", "*", and other non-letter or number keys.

...Control characters - are generated by holding down the <CTRL> key and simultaneously pressing a letter key. If a control character is printable, it appears on screen as a reverse-field upper-case letter.

"filespec"

This word is used to describe the name of a disk or cassette file. It means "file specification", and refers to the name you assigned to the text you have saved on disk. These files may contain an exact duplicate of what was displayed on the screen. Picture these files as being sheets of paper which contain your text, in file folders with names (filespecs) on them. The disk/cassette can be thought of as a filing cabinet in which the files are stored.

Refer to your C-64 manuals for a full description of the disk and cassette files; for now, we will provide only the information you need to use files with MACROFIRE.

The C-64 computer performs Input and Output operations (referred to as "I/O"), through devices such as the keyboard, video and disk drives. Each device is assigned a number. For example, all disk drives have the #8. Since you can have more than one disk drive, you can also include a drive number in the file specification. When you type in a filespec (using the Command Line commands), you must also include a device number. To write a textfile to disk you enter the command line with <CTRL>-<A>. The cursor must be positioned at the beginning of text or of the part of text you want to save.

To write a textfile to disk, you enter the commandline with <CTRL>-<A>. Again, the cursor must be positioned at the beginning of text or of the part of text you want to save.

```
      +------ enter commandline
      :
      :   +----- write command
      :   :
      :   :+------ device number
      :   ::
      :   ::+------- drive number (0-4)
      :   :::
      :   :::   +----- name of your file
      :   :::   :
      :   :::   :  +----- sequential
      :   :::   :  :
      :   :::   :  : +------ write
      :   :::   :  : :
      :   :::   :  : :           +---- finish and leave commandline
      :   :::   :  : :           :
      :   :::   :  : :      +----+----+
      :   :::   :  : :      :         :
<CTRL>-<A>W80:NAME,S,W<CTRL>-<A><CTRL>-<A>
```

If the file already exists, you can overwrite, inserting a @ after the number 8.

```
<CTRL>-<A>W8@0:NAME,S,W<CTRL>-<A><CTRL>-<A>
```

To write a textfile to cassette, you also enter the commandline with <CTRL>-<A>

```
      +------ enter commandline
      :
      :   +----- write command
      :   :
      :   :+------ device number
      :   ::
      :   :: +------- name of your file
      :   :: :
      :   :: :           +---- finish and leave commandline
      :   :: :           :
      :   :: :      +----+----+
      :   :: :      :         :
<CTRL>-<A>W1NAME<CTRL>-<A><CTRL>-<A>
```

IMPORTANT NOTICE !!!

IF NOTHING WORKS AND YOU THINK YOUR PROGRAM IS LOCKED: HIT THE <RESTORE> KEY, AND AFTER THE MESSAGE, ANY KEY.

SYSTEM HARDWARE REQUIREMENTS

MACROFIRE comes on cassette or disk

The minimum hardware required is:

1) Commodore 64
2) c-64 compatible printer (optional)
3) Disk drive 1541
4) Commodore Datasette

LOADING INSTRUCTIONS

CASSETTE VERSION

Turn the computer and all peripherals OFF. Put the MACROFIRE tape in your Datasette. Make sure the tape is completely re- wound. Switch your C-64 on and load the cassette with

```
        LOAD "MACROFIRE" <RETURN>
```

MACROFIRE will now load. Start MACROFIRE with RUN.

DISKETTE VERSION

Turn the computer OFF and turn disk drive ON. Wait for the "busy" light to turn OFF, then open the door on the disk drive and gently insert the MACROFIRE disk with the label side up. Close the door on the disk drive. Turn the computer ON. Type LOAD "MACROFIRE",8,1 Start MACROFIRE with RUN.

THE EDITOR

When MACROFIRE is started, the copyright message is displayed. Press any key to begin text entry. When the editing screen appears, you will see a line at the top of the screen containing groups of letters and numbers. This line is called the "status line. " It contains information about the status of the MACROFIRE system. Reading from left to right, you will see P:, T:, and C:, each followed by a number.

P: The number of characters preceeding the cursor (all the way to the beginning of the text) is indicated after the colon.

T: Indicates how many characters can still be typed in without exceeding memory limitations. The number will vary depending on how much text you have already typed in and how much memory you have installed in your computer.

C: Shows how much space is left in the "copy buffer." A maximum of 1024 (1K) characters can be copied at one time.

At the right side of the status line is the word OK. This indicates that you are in the normal operating mode. This is one of several two-letter messages that may appear at this position on the screen. This "message window" will inform you of errors, let you know when the copy buffer is open, and do other things which we will discuss later.

At the bottom of the screen is a line of dots, known as the "Command Line". This line is used to enter special instructions for MACROFIRE to execute, such as loading or saving text.

Below the status line, you will see a white block character at the upper left side of the screen (this corner is called the "home" position). The white block is called the "cursor". It shows you where the next character you type will appear on the screen.

To begin entering your text, just start typing. As you type, the cursor will move from left to right until it reaches the 40th character position on the screen. If you type more than 40 characters the cursor will disappear but the characters will always be inserted. To see these characters up to the 80th position type <CTRL>-<V>. The line will now be folded. To return to normal mode, type <CTRL>-<V> again (on/off).

If you hold a key down for more than 1/2 second or so, it will begin to repeat on the screen, and will keep repeating until you release the key.

For practice, type in the following words:

A COMPUTER WITHOUT(cursor here)

MACROFIRE provides you with a method to tailor the length of your lines in memory for easy management of text on the screen. When you get to the point where you would like to finish the current line, press the <RETURN> key and a new line will be started.

THE COMMANDS

There are two levels of commands in MACROFIRE: Controls, which execute immediately, and Statements, which are executed from the Command Line at the bottom of the screen. Once you have learned the various commands, it becomes almost second nature to move back and forth between the two levels.

Be sure you know which command level you are operating on so that you do not accidentally alter the text. For instance, typing <CTRL>-<K> will erase the contents of the copy buffer, while typing <CTRL>-<A> <K> <CTRL>-<A> <CTRL>-<A> (which appears as "K#") on the Command Line will delete the entire text from the computer's memory! Therefore, be cautious and double-check every statement you write on the Command Line before you execute it.

CONTROL COMMANDS

A few of the control commands are executed by first holding down the <CTRL> key and then pressing one of the keys indicated below, while still holding down the <CTRL> key. Here is a summary of the commands:

<CTRL>-<A>	Open and close command line
<CTRL>-<D>	Show directory drive 0
<CTRL>-<F>	Close copybuffer
<CTRL>-<G>	Repeat command line
<CTRL>-<H>	Display control characters (on/off)
<CTRL>-<I>	Move cursor to next TAB position
<CTRL>-<J>	Insert contents of copy buffer at current cursor position
<CTRL>-<K>	Erase copy buffer
<CTRL>-<L>	will enter form feed in text

<CTRL>-<P>	Jump into Butterfields Supermon 64
<CTRL>-<Q>	Same as cursor down
<CTRL>-<R>	Open copy buffer
<CTRL>-<S>	Move cursor to beginning of text same as HOME
<CTRL>-<T>	Same as delete backwards
<CTRL>-<V>	Show lines folded (80 characters) (on/off)
<CTRL>-<X>	Delete last line
<CTRL>-<Y>	Start MACROFIRE assembler
<CTRL>-<Z>	MACROFIRE assembler endtoken
<CTRL>-<↑>	Show actual diskstatus of drive 0
<CURSOR>-<UP>	Move cursor to beginning of preceeding line
<CURSOR>-<DOWN>	Move cursor to beginning of next line
<CURSOR>-<LEFT>	Move cursor one character backwards
<CURSOR>-<RIGHT>	Move cursor one character forwards
<INST/DEL>	Cursor backward delete
<SHIFT>-<DEL>	One character forward delete
<HOME>	Move cursor to beginning of text
<SHIFT>-<HOME>	Move cursor to end of text

These control commands can be executed with the cursor located anywhere in the text. If any character other than those shown in this list is typed, it will be displayed in the text as a control character.

Here is a more detailed look at the <CTRL> key functions, grouped into categories of action:

CURSOR MOVEMENT (non-destructive)

Controls in this group move the cursor within the text, without causing any changes to the existing text. We'll use the sample text we have just entered to illustrate what happens with each of these commands.

<SHIFT>-<CLR/HOME> - Move cursor to end of text

This command places the cursor at the end of all text in memory. If you press this combination now, the cursor will go back to the bottom of the text on screen.

<HOME> or <CTRL>-<S> - Move cursor to beginning of text

This command places the cursor prior to the first entry in the text. Press this combination and you will see the cursor jump to the top line of text.

<CURSOR>-<down arrow> or <CTRL>-<Q> - Move cursor to beginning of next line

This command places the cursor at the left of the screen, at the beginning of the next line of text, as delimited by <RETURN>. If the cursor is on the 16th line of the screen, the text is scrolled upward to display the next line. Enter this command now and watch the cursor move down the display, pushing the line it is currently on to the right by one space.

<CURSOR>-<up arrow> Move cursor to beginning of preceeding line

This command positions the cursor at the absolute beginning of the previous line, at the left side of the screen. Try this command on the sample text and watch the cursor move backward through the copy.

<CURSOR>-<left arrow> - Move cursor on character backwards

This command slides the cursor one position left, while moving the character previously in that position one place to the right.

<CURSOR>-<right arrow> - Move cursor one character forwards

This command moves the cursor to the right (or down to the next line if at the end of a line), by one position, and slides the character in that position to the left.

CURSOR MOVEMENT (changes text)

This group of controls move the cursor, and also alters the text in some way as the cursor is moved.

<INST/DEL> or <CTRL>-<T> - Delete last character

This command deletes the character to the left of the cursor and moves the text following the cursor back to the left one position.

<CTRL>-<I> - Insert Tab

This command moves the cursor to the next TAB position. If any text follows the cursor, it is moved along with the cursor. This actually places a <CTRL><I> into the text. If control character display is activated (<CTRL>-<H>), a reverse-field "I" appears, and the cursor moves one position to the right. When the control character display is off, the <CTRL><I> "expands" into the number of spaces for which the tab function is set. Number of spaces set for the tab is displayed.

<SHIFT>-<INSTL/DEL> - Delete next character

Erases the character to the right of the cursor, and moves the text to the left.

<CTRL>-<X> - Delete last line

COPY REGISTER CONTROLS

Earlier, we used the copy buffer to temporarily hold the text used for the examples. Now, let's take a more detailed look at the use of this buffer. The following group of controls access the copy buffer. When you copy material into the buffer the text is not changed.

Open copy buffer

<CTRL>-<R>

Opens copy buffer. Position the cursor at the END of the text you wish to duplicate in the copy buffer, and press this key combination. You may now move the cursor to the HEAD of the text to be copied (using any of the cursor movement commands), and press <CTRL>-<F> to close the buffer.

Close copy buffer

<CTRL>-<F>

Closes the copy buffer to further input. Use this before moving the cursor to a new location, after filling the copy buffer.

Insert contents of copy buffer into text

<CTRL>-<J>

Inserts the contents of the copy buffer at the current cursor position. Move the cursor to the position you wish to place the copy of the buffer contents, and press the key. The contents of the copy buffer are duplicated in the text, and anything after the current cursor position is moved to the end of the block of text duplicated.

Delete copy buffer

<CTRL>-<K>

This empties the copy buffer, and restores the memory counter.

To use the copy buffer, place the cursor at the END of the text to be duplicated. Open the buffer with the <CTRL>-<R>, and use the cursor movement controls, <Up arrow> (move to beginning of last line), <Right arrow> (move one character backward) or <HOME> (cursor to beginning of text), to place the cursor at the START of the area to be copied, then close the buffer with a <CTRL>-<F>.

To eliminate text you will duplicate elsewhere, it is necessary to erase the original text. If you are trying to accomplish a block move of text, you can fill the buffer and erase the original lines as you go, using the <INST/DEL> (delete last character), <CTRL>-<X> (delete last line)

OTHER CONTROLS

PRINTER control switch

Pressing <STOP> during output to a printer will stop the output

Display control codes

<CTRL>-<H>

This toggles the display of control characters on and off. When display of control codes is on, tabs and carriage returns will be displayed. When off, tabs appear as the corresponding number of blanks on screen, and carriage returns are invisibe

Disk control codes

<CTRL>-<D>

Shows directory of the disk in drive 0. Return to the editor by pressing any key.

<CTRL>-<^>

Gives you the status of the error channel of the disk. ERROR NO, ERROR MESSAGE, TRACK NO, SECTOR NUMBER Hit any key to return to the editor.

Assembler and machine monitor control codes

<CTRL>-<Y>

Starts the assembler. The assembler translates the code in the text buffer from the top of the text till the end of the text or till a <CTRL>-<Z>.

<CTRL>-<P>

Jumps into the built in monitor (Supermon by J. Butterfield). See monitor description for the several commands, except the X-command which causes the monitor to jump back to the editor via the copyright message (hit any key) instead of jumping to BASIC.

COMMAND LINE CONTROLS

These two controls enter, or execute, the command line.

<CTRL>-<A> - Open/close command line.

<CTRL>-<A> opens, closes, and executes the command line, as we shall see in a moment.

<CTRL>-<G> - Execute command line

This command executes the last command line entry again. This is useful for repeating a command line that encountered an error, after correcting the error condition.

THE COMMAND LINE

Inputs are made on the command line by first pressing the <CTRL>-<A> keys. This causes the first dot on the command line to be replaced by a dollar sign ("$"). You can then enter multiple commands, separated by <CTRL>-<A> (which displays the "$"). To execute the command line, just press <CTRL>-<A> <CTRL>-<A>. This displays a $ symbol at the end of the line, and after executing the commands you have entered, a "number sign" ("#") will be displayed.

The command line is used for such functions, as printing formatted text, retrieving files from a disk, invoking certain editing functions, and using Commodore's disk I/O-commands.

The following commands are available from the command line:

<@>	set tab position
<B>	one character backward
<D>	delete preceeding character
<F>	one character forward
<G>	insert contents of copy register
<H>	insert hex byte
<I>	insert string
<J>	repeat command line
<K>	delete text buffer (careful!!)
<L>	list text to a device
<M>	list text to a Commodore printer
<P>	send command(s) to disk
<R>	read file
<S>	search for string
<T>	delete next line
<U>	jump to userprogram
<W>	write file

Some of these commands can be executed several times by entering a number from 2 through 255 in front of the command. You also can repeat the whole line using the <CTRL>-<J> command, which brings you back to the beginning of the line. This is possible only if the command line was executed without error. If an error occurred the program jumps back to the control mode.

Here is a description of the parameters used with command line instuctions:

<@> set tab position
Format: @<n> <CTRL>-<A> (where n = 1 thru 9)
Example: $@5$#

This command sets a tab every five spaces. Pressing <CTRL>-<I> causes the cursor to advance to the next tab position. If the display of control characters is suppressed (<CTRL>-<H>, you will see the number of spaces set for the tab width each time you enter the <CTRL>-<I> command. If the display of control characters is not suppressed, you will see a reverse-field "I" each time you tab over, but the cursor will advance only one space on the screen. During printout, tabs are ignored if print formatting instructions are entered. In an unformatted printout, the tabs will be as seen in the text, with control characters suppressed. Tabs are stored as control characters in the text file, and take up less room than an equivalent number of spaces would. Tab positions are only multiples of "n".

<H> insert hex byte
Format: H<byte>ESC

Example: $H1D$#

This inserts the hex value $1D at the current cursor position. (The $1D value represents an end-of-file marker for a disk/cassette file.) This command can be repeated up to 255 times by inserting the number of bytes desired prior to the "H", as in $25H20$#.

<I> insert string
Format: I<string>ESC
Example: $IHello$#

This inserts the word specified string at the current cursor position. Repeated inserts can be done by typing the desired number of repeats prior to typing the "I" statement.

<J>

Jump to beginning of command line. This command is very useful if you want to change or delete characters or words in the text and you don't know how much to change. You jump to the beginning of the command line and repeatedly execute the command. The command "J" will be executed until an error occurs or until all characters or words are changed. You don't have to press <CTRL>-<G>. Do not use it, if an error cannot occur. For example: $sTEXT$4ditext$j$# will change all "TEXT" in "text" through the whole textbuffer.

<L>

To list a file to any device, except the keyboard! Sends official ASCII code. This includes Carriage Return and Line Feed (0D followed by 0A). This is the command you use to print your text. It does not work with your Commodore printer. If you have a Commodore printer, please refer to the M-command.

1. List on cassette

```
      +---- open command line
      :
      :   +----- start list command
      :   :
      :   :+-------- device number (cass. = 1)
      :   ::
      :   ::+------ secondary address ( 1=without eot )
      :   :::
      :   :::  +----- file name
      :   :::  :
      :   :::  :         :--- close commandline
      :   :::  :       +--+------+
      :   :::  :       :         :
<CTRL>-<A>L11NAME<CTRL>-<A><CTRL>-<A>
```

2. List on RS232 via USER PORT

How to connect and wire - see the Reference Manual.

```
+---- open command line
:
:   +----- start list command
:   :
:   :+-------- device number (cass. = 1)
:   ::
:   ::+------ secondary address ( 1=without eot )
:   :::
:   :::+----- control register byte in hex (*)
:   ::::
:   :::: +----- command register byte in hex (*)                    DO THIS
:   :::: :                                                     BEFORE RUNNING
:   :::: :          :--- close commandline                        MACROFIRE
:   :::: :        +--+------+                                        (4800)
:   :::::::   .   :         :
<CTRL>-<A>L208610<CTRL>-<A><CTRL>-<A>
     ^A L208010 ^A^A      <-- OPEN 2,2,0, CHR$(0)+CHR$(1)+CHR$(6)+CHR$(0)
```

(*) The examples above cause the RS232C interface to work with the following settings.

a) Control Register byte

```
    8   :   6
+-+-+-+-:-+-+-+-+
:1:0:0:0:0:1:1:0:
+-+-+-+-+-+-+-+-+
 : : : u    :            u = unused
 : +++      :
 :  :       :
 :  +---+   +--- 300 baud
 :      :
2 stop   8 bits
 bits
```

See your Commodore Programmer's Reference Guide on page 350.

b. Command Register Byte

```
    1   :   0
+-+-+-+-:-+-+-+-+
:0:0:0:1:0:0:0:0:
+-+-+-+-+-+-+-+-+
 : : : : u u u :            u = unused
 : : : :       :
 +-+-+ :       +--- handshake 0-3 line
   :   :
   :   +---- half duplex
   :
   +--- parity disabled, non generated
```

If you want to change the RS232 specifications see your Reference Guide. Works practically on every RS232 printer. We tested it with the Qume SPRINT9, and Decwriter. Not mentioned in the manual is that you have to insert line connectors such as Transmit Data a.s.o.

3. List on Screen

```
      +---- open command line
      :
      :   +----- start list command
      :   :
      :   :+-------- device number
      :   ::
      :   ::+------ mode
      :   :::
      :   :::
      :   :::
      :   :::        :--- close commandline
      :   :::      +--+------+
      :   :::      :         :
<CTRL>-<A>L30<CTRL>-<A><CTRL>-<A>
```

4. List on IEEE-Printer via internal IEEE-Bus. Not for Commodore printers (and Commodore compatible printers). (See M-Command)

```
      +---- open command line
      :
      :   +----- start list command
      :   :
      :   :+-------- device number
      :   ::
      :   ::+------ secondary address (see printer manual)
      :   :::
      :   :::
      :   :::
      :   :::        :--- close commandline
      :   :::      +--+------+
      :   :::      :         :
<CTRL>-<A>L40<CTRL>-<A><CTRL>-<A>
```

5. List on Disk

A unique feature of MACROFIRE is that you can list (SAVE) formatted text to disk and load it back into memory again.

Example:

```
      +---- open command line
      :
```

```
:   +----- start list command
:   :
:   :+-------- device number
:   ::
:   ::+------ secondary address ( always 6 )
:   :::
:   :::+------- drive number (0..4)
:   ::::
:   ::::   +----- file name
:   ::::   :
:   ::::   :  +----- sequential file
:   ::::   :  :
:   ::::   :  : +------- write
:   ::::   :  : :
:   ::::   :  : :         :--- close commandline
:   ::::   :  : :      +--+-------+
:   ::::   :  : :      :          :

<CTRL>-<A>L860:NAME,S,W<CTRL>-<A><CTRL>-<A>
```

This lists to disk similar to the printer, including Line Feeds and CR (formatted). This text can be read by using the READ command (described later). If you do so you can see the <CTRL>-<J> = Line Feeds which you now can delete by the following sequence:

```
+------ open commandline
:
:   +-------- search string command
:   :
:   :      +------ search for <CTRL>-<J>
:   :      :
:   :      :         +------- end of searchstring
:   :      :         :
:   :      :         :  +----- delete character (<CTRL>-<J> in this case)
:   :      :         :  :
:   :      :         :  :+-------- repeat commandline until error
:   :      :         :  ::
:   :      :         :  ::        +--- close commandline
:   :      :         :  ::        :
:   :      :         :  ::    +----+----+
:   :      :         :  ::    :         :

<CTRL>-<A>s<CTRL>-<J><CTRL>-<A>dj<CTRL>-<A><CTRL>-<A>
```

NOTE: Use always 6 as a secondary address.

<M>

Lists text in Commodore format. You use the command exactly like the L-command. It does not however, send an additional line feed after a carriage return. Use this command for CBM 2022 and VIC1525.

Example: List on a 2022 and VIC1525

```
      +---- open command line
      :
      :   +----- start list command for commodore printers or compatibles
      :   :
      :   :+-------- device number
      :   ::
      :   ::+------ secondary address (see printer manual)
      :   :::
      :   :::
      :   :::
      :   :::         :--- close commandline
      :   :::      +--+------+
      :   :::      :         :
<CTRL>-<A>M40<CTRL>-<A><CTRL>-<A>
```

<R>

Read file. The file is loaded, beginning at the current cursor position. After the load is completed, the cursor is positioned at the end of the text just loaded. If there is any text after the cursor, before the file is loaded, then it still will be in the same position after the file is loaded. Text entered prior to the present cursor position, stays in memory. To delete the text in memory you have to kill that text first. The READ command thus works like an INSERT command.

How to read a text file from disk

```
      +---- open command line
      :
      :   +----- start formatter command
      :   :
      :   :+-------- device number
      :   ::
      :   ::+------ drive number (0..4)
      :   :::
      :   :::  +------ file name
      :   :::  :
      :   :::  :          :--- close commandline
      :   :::  :       +--+------+
      :   :::  :       :         :
<CTRL>-<A>R80:NAME<CTRL>-<A><CTRL>-<A>
```

How to read a textfile from cassette

```
      +---- open command line
      :
      :   +----- start formatter command
      :   :
      :   :+-------- device number
      :   ::
      :   :: +------ file name
      :   :: :
      :   :: :
      :   :: :
      :   :: :         :--- close commandline
      :   :: :        +--+------+
      :   :: :        :         :

<CTRL>-<A>R1NAME<CTRL>-<A><CTRL>-<A>
```

<S>   search for string
Format: S<string><CTRL>-<A>
Example: $Sbaseball$#

This command will search for the next occurrence of the specified string and move the cursor to the point just past the occurrence. If you include a number (from 1 thru 255) before the "<S>," the search will stop at that occurrence of the target string. For example, $5Sand$# will stop at the 5th occurrence of the word "and", beginning at the current cursor position. You can do manual multiple searches as well by not including a number first, and using the <CTRL>-<G> to re-execute the search. The search will continue until the end of the text is reached. If the target string cannot be found, the error message "S?" appears at the upper right side of the screen.

<U>

Jump to userprogram located at $B000 (hexadecimal) You can safely place your programs in the memory range from $B000 to $CFFF, but watch out. Any printer, cassette, or disk I/O in the editor will use this area!

<W>

To write a textfile to disk, you enter the commandline with <CTRL>-<A>. The cursor must be positioned at the beginning of the text or of the part of text you want to save.

```
       +------ enter commandline
       :
       :   +----- write command
       :   :
       :   :+------ device number
       :   ::
       :   ::+------- drive number (0-4)
       :   :::
       :   :::     +----- name of your file
       :   :::     :
       :   :::     :  +----- sequential
       :   :::     :  :
       :   :::     :  : +------ write
       :   :::     :  : :
       :   :::     :  : :           +---- finish and leave commandline
```

```
      :   :::   :  :  :           :
      :   :::   :  :  :      +----+----+
      :   :::   :  :  :      :         :

<CTRL>-<A>W80:NAME,S,W<CTRL>-<A><CTRL>-<A>
```

If the file already exists you can overwrite inserting a @ after the number 8.

```
<CTRL>-<A>W8@0:NAME,S,W<CTRL>-<A><CTRL>-<A>
```

To write a textfile to cassette you also enter the commandline with <CTRL>-<A>

```
      +------ enter commandline
      :
      :   +----- write command
      :   :
      :   :+------ device number
      :   ::
      :   :: +------- name of your file
      :   :: :
      :   :: :            +---- finish and leave commandline
      :   :: :            :
      :   :: :       +----+----+
      :   :: :       :         :

<CTRL>-<A>W1NAME<CTRL>-<A><CTRL>-<A>
```

During the WRITE Command the screen will be cleared. After the file is written or an error occurs, MACROFIRE jumps back to the editor.

ERROR MESSAGES OF THE EDITOR

The error messages of MACROFIRE are two characters long and are displayed in the upper right-hand corner of the screen. If an error occurs, the message is displayed and control returns to the editor. These are the error messages and circumstances:

CO	Command line overflow. No more than 40 characters are allowed on the command line.
E?	Invalid character in command line.
#?	The number in front of a command is larger than 255
I?	No more space in text buffer.
C?	Copy register is full
H?	Invalid hex value indicated (valid range = $00 to $FF)
T?	Tab value larger than 9 or smaller than 1
S?	String not found, or no target string indicated.
L?	Something went wrong during execution of the L(ist) command. (Wrong file, printer not attached, etc.)
RW	Read/Write error. (File does not exist or error occurred during the read/write operation.)
V?	No directory. An error occurred during an attempt to read the directory.

How to connect your Commodore 64 to a serial printer

The Commodore-64 comes with a built in RS232 interface for connection with a printer, modem, or another computer with a RS232 interface (the TRS-80 Model 100 for instance).

The RS232 interface, which is implemented through the user port, is not a real RS232 with the +-12V levels. There is only a TTL-level as a transmitted data line available. We used those TTL-level RS232 interfaces with the following printers; the DECWRITER, the QUME Sprint9, the BROTHER HR15, and the NEC Spinwriter. On all these printers and even with the smartmodem from Hayes the TTL-level RS232 worked fine.

The RS232 interface software is built in and the transmission specifications can be set up via certain register settings.

To hook up your RS232 printer all you need is the following:

1. A user port connector 24 pin from
   TRW CINCH 251-12-50-170/50-24sn-98124
   available from your local computer store or from a distributor specialized in connectors.

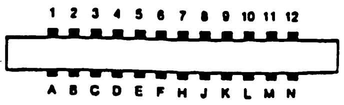
2. An RS232 connector, 25 pin

3. Two or three lines

It turned out that a 3 line interface, without any handshaking, was easiest to connect to our C-64 with a Qume Sprint9 letter quality printer. To wire a 3 line interface you only need to connect:

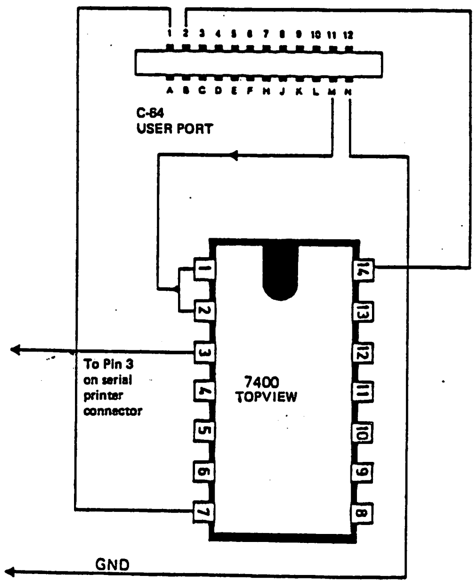
Pin	Description	In/Out	6526 Pin
C	RECEIVE DATA	IN	PBO
B	RECEIVE DATA	IN	FLAG2
M	TRANSMIT DATA	OUT	PA2
N	GROUND	-	GND

If you want to connect an RS232 printer you only need to wire the two lines GND and transmitted data.



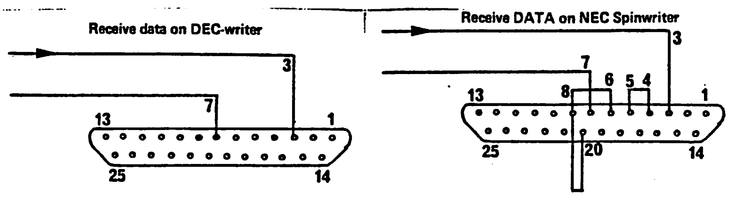
User port connector on your C-64

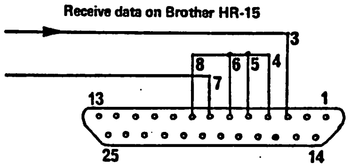
We found out that the transmit data line, coming out of pin M, must be converted before feeding into the printer. Therefore, you can connect a 7400 NAND gate directly to the user port connector using Pin 1 and Pin 2 as power supply lines.



If you need more than 4 lines to be inverted, use a 7404 inverter chip containing 6 inverters.

Many printers with RS232 interface need some handshaking. You have to wire the printer input connector according to the following schematics. We have tried it out with the following printers:





THE ASSEMBLER

The assembler can be started directly from the editor by pressing <CTRL>-<Y>. The assembler translates (in three passes) the source-code text stored in the editor textbuffer starting from textbegin till a <CTRL>-<Z> (assembler stop sign) is encountered. If no <CTRL>-<Z> is set in the text, assembling is executed till the end of the text. When the assembler enters the second pass an action indication is displayed at the last position of the line on top. As an output option, you can obtain from the third pass on a list of the assembled text , a list of the labels, or both, together edited on screen. These lists can also be ouput to a printer.

If during assembling no errors have been encountered, the assembler stops after completion. By pressing any key you return to the editor via the copyright message (press any key again).

If an error is found during assembling, the assembler stops immediately and a self-explaining error message is displayed on screen. By pressing any key you return to the editor, via the copyright statement, the cursor will be positioned behind the erroneous textpart. These features aid considerably to a quicker correction.

Formats and syntax of the opcodes

The assembler interprets all existing opcodes of the 6500 assembler language set, as well as a few additional pseudo opcodes which are in fact control commands for the assembler.

6500 OPcodes:

```
ADC AND ASL BCC BCS BEQ BIT BMI BNE BPL BRK
BVC CLC CLD CLI CLV CMP CPX CPY DEC DEX DEY
EOR INC INX INY JMP JSR LDA LDX LDY LSR NOP
ORA PHA PHP PLA PLP ROL ROR RTI RTS SBC SEC
SED SEI STA STX STY TAX TAY TSX TXA TXS TYA
```

Pseudo OPcodes:

```
EQU       : EQUal used for label definitions. Label becomes value of
            expression after the EQU.
EPZ       : Equal Page Zero. Same as EQU.
ORG       : ORiGine. To fix start adress for assembled object code.
            Where the program has to be put!
DFB       : DeFine Byte. Inserts following expression(s) as byte(s).
DFW       : DeFine Word. Same as DFB but now for words. That means
            two bytes per Expression (lower than higher byte).
ASC       : ASCII-String,insertion of an ASCII-string
OUT       : OUTput (see later)
INCLUDE   : INCLUDE sourcefile during assembly. After assembly there
            is no text really in the buffer included. So this Opcode
            allows you to assemble bigger source files then the length
            of the textbuffer.(assembling from disk)
MACRO     : Begin of macro with formal parameterlist (see later)
MEND      : End of macro
```

Format of adressing modes:

```
Impl.Acc  : [label]_OPC[_;<com>]<RETURN>
Immediate: [label]_OPC_#<expr>[_<com>]<RETURN>
Abs,Zp,Re: [label]_OPC_<expr>[_<com>]<RETURN>
AbsX;ZpX  : [label]_OPC_<expr>,X[_<com>]<RETURN>
AbsY;ZpY  : [label]_OPC_<expr>,Y[_<com>]<RETURN>
IndX.     : [label]_OPC_(<expr>,X)[_<com>]<RETURN>
IndY.     : [label]_OPC_(<expr>),Y[_<com>]<RETURN>
Ind       : [label]_OPC_(<expr>)[_<com>]<RETURN>
```

Format of Pseudo OPcodes:

```
EQU       : <label>_EQU_<expr>[_<com>]<RETURN>
EPZ       : <label>_EPZ_<expr>[_<com>]<RETURN>
ORG       : [label]_ORG_<PL>[,PP][_<com>]<RETURN>
DFB       : [label]_DFB_<expr>[,<expr>]"[_<com>]<RETURN>
DFW       : [label]_DFW_<expr>[,<expr>]"[_<com>]<RETURN>
ASC       : [label]_ASC_<delim>[string]<delim>[_<com>]<RETURN>
OUT       : [label]_OUT_[L][N][M][C][,<dev>[,<sec>[,"<name>"]]]<RETURN>
INCLUDE   : [label]_INCLUDE_<dev>,<sec>,"<name>"<RETURN>
MACRO     : <label>_MACRO_[<formp>[,<formp>]"]<RETURN>
MEND      : [label]_MEND_[_<com>]<RETURN>
```

Syntax legend:

Expressions within <> are mandatory. Expressions within [] are optional. A quote (") after] indicates that the whole expression within [] can be repeated as many times as desired. A _ indicates that at least one space or one <CTRL>-<I> must be inserted. OPC indicates a legal OPcode.

Textlines beginning with * or <CTRL>-<L> are skipped by the assembler as comment. An empty line (a line containing only a <RETURN>) is also skipped.

label:	Label consisting of a letter followed by letters or digits. All characters are significant.
string:	String of ASCII-characters. Their values will be inserted here according to their position within the string. A string may contain a maximum of 250 characters and may be opened and closed by the same delimiter <delim>.
delim:	All non-alphanumeric characters. The use of \ as delimiter increases automatically the value of the last string character by 128 (signbit on).
com:	Comment may placed here
<RETURN>:	The return key on the right side of your keyboard.
expr:	An expression can be a decimal number, a hexnumber(preceded by an $ like $40 or $FFFB), a binary number(preceded by an % like %1011 or %10101111), a label, an ASCII character(preceded by ′ like ′A or ′#) or in combination with arithmetic operators + - * / (an expression containing sums, products or quotients of other expressions). The Arithmetic value range is -65536 to +65535. An asterisk * as a term in an expression will be interpreted as the value of the actual program pointer. This makes relative indexing very easy. Expressions will be evaluated from left to right and they may be nested by using brackets ().
PL,PP:	PL = logical start address of the assembled program. This means that the assembled program will only work starting on this address. PP = physical start (dump) address of the program. That means the program will be dumped starting from this address on, but not necessarily work. (except PL=PP) So you can assemble a program for a memory range that, for instance, is occupied by another program or by the labeltables or textbuffer of the assembler. If you don′t define PP explicite PP will be equal PL.
dev:	Stands for devicenumber. That is the regular number for the devices specified in the different manuals. The device number may be an expression like any other one.
sec:	Stands for secondary adress and is mostly device dependent. See manual. By diskusage and combined INCLUDE from and OUTput to disk the secondary addresses defined by both speudo Opcodes (INCLUDE and OUT) must be different. Example:

```
        INCLUDE 8,5,"0:FILE"
```

Works properly only if the OUT command looks like:

```
OUT LN,8,6,"0:OUTFILE,S,W"
```

You see, once the secondary address is equal to 5 and the other time it is equal to 6, so they cannot interfere.

name: The filename of the file written or read to/from a device By disk usage for output the name must contain ",S,W" at the end as you can see in the example above.

formp: The formal parameter used in the macro body (between MACRO and MEND). The formal parameter will be replaced by an actual parameter in the inserted macro body during calling of the macro, followed by the actual parameter:
Macrobody somewhere declared:

```
SAMPLE MACRO A,B  (A and B are the formal parameters)
       LDA #A
       STA B
       MEND
```

example of a MACRO call:

```
SAMPLE 145,LOC (145 and LOC actual parameters)
```

this call will result in:

```
LDA #145
STA LOC
```

So as you can see, the formal parameters are replaced by the actuals. More about macros later.

The OUT direction

To spool your listing to a printer, you have to use the OUT direction of the assembler. The text directly after the OUT direction will be printed, but if the OUT has no options (L,M,N) the OUT direction will reset the printfunction. This means that no text from that statement on is produced. As you can see this enables you to print only parts of the listing.

The OUT direction has three to four options and needs a device number and sometimes a secondary address and filename. The 'L' option stands for Listing. This means the printing of opcodes and the generated objectcode. The 'N' option stands for Namelisting, which means a printout of all declared labels and macros. If a label or a macro is not used, the assembler will print 'UNUSED' in the Namelisting. The 'M' option stands for 'Macro not expanded' indicating that the macro body will not be printed at every macrocall in the listing, but only the single call with the generated code.

There is another option which you can place directly after the other options. This option is a single 'C' which stands for Commodore and is necessary if you want to print on a VIC1525 printer or compatible!

How to print the listing on screen

To print on screen, the following line has to be entered in front of the part of text that you wish to be printed:

```
      +------ TAB to have a nice looking source
      :
      :    +---- OUT direction (speudo opcode)
      :    :
      :    :  +--- assemblerlisting (optional)
      :    :  :
      :    :  :+----- labellisting (optional)
      :    :  ::
      :    :  ::+----- macros not expanded (optional)
      :    :  :::
      :    :  :::+--- comma
      :    :  ::::
      :    :  ::::+----- device number ( screen=3 )
      :    :  :::::
      :    :  :::::   +---- return to terminate the line
      :    :  :::::   :
<CTRL>-<I>OUT LNM,3<RETURN>
```

How to print to a printer connect via the internal (IEEE) bus

To print on a printer (mostly device #4) you have to enter the following line in front of the text.

```
      +------ TAB to have a nice looking source
      :
      :    +---- OUT direction (speudo opcode)
      :    :
      :    :  +--- assemblerlisting (optional)
      :    :  :
      :    :  :+----- labellisting (optional)
      :    :  ::
      :    :  ::+----- macros not expanded (optional)
      :    :  :::
      :    :  :::+--- comma
      :    :  ::::
      :    :  ::::+----- device number ( printer=4 )
      :    :  :::::
      :    :  :::::+------ comma
      :    :  ::::::
      :    :  ::::::+---- secondary address (see printer manual)
      :    :  :::::::
      :    :  :::::::   +---- return to terminate the line
      :    :  :::::::   :

<CTRL>-<I>OUT LNM,4,0<RETURN>
```

Add a 'C' after the options (L,N,M) if you want to print on the commodore VIC1525 or compatible.

How to print via the build-in RS232 interface

Enter the following line in source:

```
       +------ TAB to have a nice looking source
       :
       :    +---- OUT direction (speudo opcode)
       :    :
       :    :  +--- assemblerlisting (optional)
       :    :  :
       :    :  :+---- labellisting (optional)
       :    :  ::
       :    :  ::+---- macros not expanded (optional)
       :    :  :::
       :    :  :::+--- comma
       :    :  ::::
       :    :  ::::+--- device number ( rs232 = 2)
       :    :  :::::
       :    :  :::::+--- comma
       :    :  ::::::
       :    :  ::::::+--- secondary address (has to be zero)
       :    :  :::::::
       :    :  :::::::+--- comma
       :    :  ::::::::
       :    :  ::::::::+---- quote
       :    :  :::::::::
       :    :  :::::::::   +------ control register byte as character (*)
       :    :  :::::::::   :
       :    :  :::::::::   :       +---- command register byte as character (*)
       :    :  :::::::::   :       :
       :    :  :::::::::   :       :    +-- quote
       :    :  :::::::::   :       :    :
       :    :  :::::::::   :       :    :   +--- return
       :    :  :::::::::   :       :    :   :
<CTRL>-<I>OUT LNM,2,0,"<contrb><comndb>"<RETURN>
```

(*) to enter these bytes as characters, you have to enter following sequence

```
      +--- open command line
      :
      :   +--- insert hex byte command
      :   :
      :   :+--- hex value of control register byte (*)
      :   ::
      :   :: +--- insert hexbyte command
      :   :: :
      :   :: :+------ hex value of command register byte (*)
      :   :: ::
      :   :: ::           +---- close and execute commandline
      :   :: ::           :
      :   :: ::      +----+----+
      :   ::::::     :         :
```

<CTRL>-<A>h86h10<CTRL>-<A><CTRL>-<A>

(*) The example above causes the RS232C interface to work with the following settings.

a) Control register byte

```
        8   :   6
     +-+-+-+-+:-+-+-+-+
     :1:0:0:0:0:1:1:0:
     +-+-+-+-+-+-+-+-+
      : : : u     :              u = unused
      : +++       :
      :  :        :
      :  +---+    +--- 300 baud
      :      :
    2 stop  8 bits
     bits
```

See your Commodore Programmers Reference Guide on page 350.

b) Command Register byte

```
        1   :   0
     +-+-+-+-+:-+-+-+-+
     :0:0:0:1:0:0:0:0:
     +-+-+-+-+-+-+-+-+
      : : : : u u u :          u = unused
      : : : :       :
      +-+-+ :       +--- handshake 0-3 line
        :   :
        :   +---- half duplex
        :
        +--- parity disabled non generated
```

If you want to change the RS232 specifications, see your reference guide. This works practically on every RS232 printer. We tested it with the Qume SPRINT9 and the Decwriter. For the connection of the printer to the RS232 interface check previous chapter.

It is also possible to spool the printout to the disk or cassette. You only have to give the devicenumber, secondary address, and filename.

THE INCLUDE DIRECTION

A unique feature of MACROFIRE is the INCLUDE direction. This feature makes it possible to chain previously saved sourcefiles on disk or cassette during assembling into the actual sourcetext in the textbuffer of the editor. After completion or abortion of the assembly there will be nothing changed in the textbuffer. As you can see it is now possible to assemble larger sourcetexts as the capacity of the textbuffer allows.

How to include a file from cassette

First you have to save the text that you want to include, three times on tape. The assembler has 3 passes and every pass it opens the file again.

How to embed the INCLUDE direction into the text

You have to enter following line:

```
+--- TAB for nice looking source text
:
:      +--- INCLUDE direction
:      :
:      :    +--- devicenumber (cassette = 1)
:      :    :
:      :    :+--- comma
:      :    ::
:      :    ::+--- secondary address has to be zero
:      :    :::
:      :    :::+--- comma
:      :    ::::
:      :    ::::+--- quote
:      :    :::::
:      :    :::::   +--- name of file you have saved via editor 3 times on
:      :    :::::   :                                               tape
:      :    :::::   :    +--- quote
:      :    :::::   :    :
:      :    :::::   :    :   +--- return
:      :    :::::   :    :   :
<CTRL>-<I>INCLUDE 1,0,"FILENAME"<RETURN>
```

If you want to include from a disk file, you don't have to save the file 3 times! The file will be opened 3 times according to the 3 passes. Next figure shows you how to insert the INCLUDE direction for a disk file.

```
+--- TAB for nice looking source text
:
:      +--- INCLUDE direction
:      :
:      :    +--- devicenumber (disk = 8)
:      :    :
:      :    :+--- comma
:      :    ::
:      :    ::+--- secondary address has to be 5
:      :    :::
:      :    :::+--- comma
:      :    ::::
:      :    ::::+--- quote
:      :    :::::
:      :    :::::+--- drive number (0-4)
:      :    ::::::
:      :    ::::::    +--- name of file you have saved via editor
:      :    ::::::    :
    :     :    ::::::    :    +--- quote
    :     :    ::::::    :    :
    :     :    ::::::    :    :   +--- return
    :     :    ::::::    :    :   :
```

```
<CTRL>-<I>INCLUDE 8,5,"0:FILENAME"<RETURN>
```

It is possible to include and output at the same time. During including the assembler will of course slow down, because the disk I/O is very slow. It is not allowed to nest several INCLUDE's, which means the included file is not allowed to contain another INCLUDE.

THE USE OF TABS

To have a neat looking output during assembly and editing, it is of great use to insert <CTRL>-<I> (Tab's) in front of the opcode and between a label and the opcode. All opcodes will start at the same column. To keep them on the same column it is also necessary to use labels with a maximum of eight characters.

Example:

type in the following sequence:

```
<CTRL>-<I>OUT LN,3<RETURN>
BSOUT<CTRL><I>EQU $FFD2<RETURN>
<RETURN>
<CTRL>-<I>ORG $B000<RETURN>
<RETURN>
START<CTRL>-<I>LDX #32<RETURN>
LOOP<CTRL>-<I>TXA<RETURN>
<CTRL>-<I>JSR BSOUT<RETURN>
<CTRL>-<I>INX<RETURN>
<CTRL>-<I>BPL LOOP<RETURN>
<CTRL>-<I>BRK<RETURN>
```

Now, by starting the assembler with <CTRL>-<Y>, the listing and labellisting will appear on screen. Press any key. The copyright message will appear. Press any key and you will return to the top of the text.

Now type <CTRL>-<P> and you will jump into the monitor. The program dump starts at location $B000 (hexadecimal). Type G B000 and the program will execute. A A sequence of characters will appear on screen. By typing X, you will return via the copyright message into the editor.

WHERE TO DUMP PROGRAMS DURING ASSEMBLY

For the user there is a user area reserved. It starts from location $B000 and ends at location $CFFF (hexadecimal). Watch out! This area will be used by any kind of I/O (Disk, Cassette, RS232) in the editor so save the dumped program via the monitor if necessary. The memory range $0800 to $AFFF is occupied by the Assembler and Editor. The following zeropage locations, $50 to $8F, may not be destroyed if you wish to return to the editor again.

THE MACRO's AND THEIR USAGE

The purpose of macros as well as of subroutines is to replace often needed routines by a simple call.

Although macros and subroutines are similar there are differences which you have to know in order to be able to use them the best way.

Subroutines as well as macros are written only once by the programmer but the assembler writes subroutines only once while inserting the whole macro every time they are called. By that technique the stackoperations necessary with the use of subroutines are sowed.

In general we can say:
Macros need more memory but they are faster. Subroutines are easier to use.

At the beginning of the main program where the macro is defined, we use formal parameters. These formal parameters are used instead of the actual parameters handled over to the macro every time we call it. Parameters used only within the macro are named local parameters.

Example:

```
macro-definition: INC2      MACRO PT
                            INC PT
                            BNE G@
                            INC PT+1
                  G@        EQU *
                            MEND
```

Name of the macro ist INC2
Name of the formal parameter is PT
A local label is G@.
Putting the @ after the label in the macrobody will generate
a unique label with every call.

```
Macro-call                  INC2 AUX
```

Name of the actual parameter is AUX

Another example shows a blocktransfer-routine. This routine moves a block of bytes between STARTP and ENDP to a location beginning at INTOP.

```
MOV     LDA STARTP
        CMP END
        BNE CONT
        LDA STARTP+1
        CMP ENDP+1
```

```
         BEQ ENDMOV
CONT     LDX #0
         LDA (STARTP,X)
         STA (INTOP,X)

         INC STARTP    -+
         BNE INCINTOP   +-- INC2 STARTP
         INC STARTP+1  -+

INCINTOP INC INTOP     -+
         BNE JUMP       +-- INC2 INTOP
         INC INTOP+1   -+

JUMP     JMP MOV
ENDMOV   BRK
```

If you have defined our sample macro at the beginning you can replace the parts of the program marked by brackets by calling macro INC2 STARTP and INC2 INTOP.

ERROR MESSAGES

If an error occurs during assembling, an error message will be displayed and the assembler stops. Pressing any key will return to the copyright message. You have to touch any key again and the cursor will be positioned directly after the erroneous textpart so you can immediately start correcting. The exception happens when the error occurrs in an included file. The cursor then will be positioned after the include statement where the erroneous textpart was included. The majority of error messages are self explaining. The few exceptions we will now discuss.

LINE TOO LONG

The line contains more than 127 characters.

OPCODE DIFFERENT

Opcode is recognized different between pass 2 and pass 3. For instance, a label was recognized as an absolute label on the second pass and then in pass 3 it turned out as zero page instruction. Define label more carefully. Another example 2 ORG commands are in conflict.

WARNINGS DURING ASSEMBLY

During the list of the assembled code, a warning can appear:

"WARNING OPERAND OVERFLOW"

This happens if you want to put a two byte expression into a one byte location. This must not necessarily be an error, because it can happen on purpose. If no warnings occur, "NO WARNINGS" is printed by the assembler.

HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER