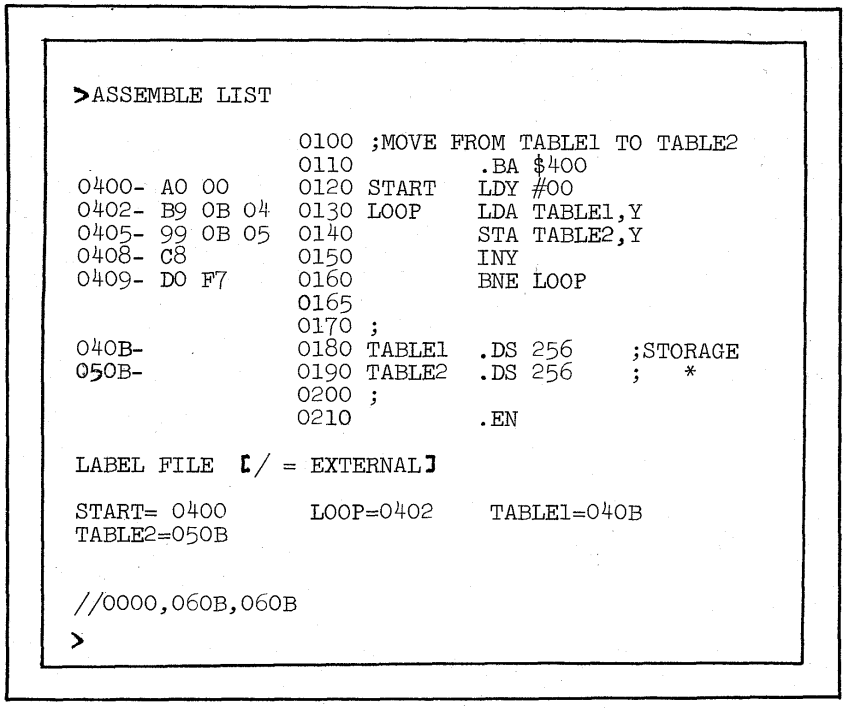
65Ø2

MACRO ASSEMBLER

AND

TEXT EDITOR

FOR PET, APPLE, SYM and OTHERS

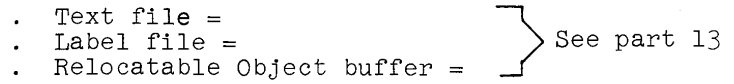


```
>ASSEMBLE LIST

                    0100 ;MOVE FROM TABLE1 TO TABLE2
                    0110          .BA $400
0400- A0 00         0120 START    LDY #00
0402- B9 0B 04      0130 LOOP     LDA TABLE1,Y
0405- 99 0B 05      0140          STA TABLE2,Y
0408- C8            0150          INY
0409- D0 F7         0160          BNE LOOP
                    0165
                    0170 ;
040B-               0180 TABLE1   .DS 256     ;STORAGE
050B-               0190 TABLE2   .DS 256     ;   *
                    0200 ;
                    0210          .EN

LABEL FILE [/ = EXTERNAL]

START= 0400       LOOP=0402     TABLE1=040B
TABLE2=050B


//0000,060B,060B
>
```

This ASSEMBLER and TEXT EDITOR was written in machine language-not BASIC

1. INTRODUCTION

This 6502 relocating Macro assembler (ASSM) and text editor (TED) resides simultaneously in approximately 8K bytes of memory. The ASSM/TED can be loaded into RAM or stored in ROM memory. Sufficient memory must be provided for not only the ASSM/TED but for a text file and label file (symbol table). Approximately 2K is sufficient memory for the text file for small programs or larger programs if assembled from tape. A good rule of thumb is one byte of memory for the label file for each byte of object code. If an executable object code file is to be stored in memory during assembly, sufficient memory must be provided for that also. On cold start entry (2000), the ASSM/TED will set the file boundaries as follows.



The label file and text file that this ASSM/TED generates is position independent and may be located pratically anywhere in RAM memory. The object code file location is dependent on the beginning of assembly (.BA pseudo op) and the .MC pseudo op.

The ASSM/TED was designed such that records in the label file and text file are variable in length and directly dependent on the number of characters to be stored. This results in more efficient utilization of memory.

Some unique features of this ASSM/TED are:

- Macro and conditional assembly support.
- Labels up to 10 characters in length.
- Auto line numbering for ease of text entry.
- Creates both executable code in memory and relocatable object code on tape.
- Manuscript feature for composing letters and other text.
- Loading and storing of text on tape.
- Vectors for linkage to disc operating systems.
- Supports up to two tape decks, CRT and keyboard, and printer.
- String search and replace capability, plus other powerful editing commands.

Throughout this document, output generated by the ASSM/TED is underlined to distinquish from user input.

Initial entry to the ASSM/TED is at address 2000. If the break command (>BR) is executed, one may return to the address following the break. Initial entry provides the following default parameters:

- Format - set
- Manuscript - clear
- Auto line numbering - 0 or clear
- Text file - clear
- Tape decks - off

The ASSM/TED is designed to operate with a record deck and a separate play deck and/or disc system. A single record/play deck may be used but one will not be able to create relocatable object files when assembling from tape.

This software has been extensively tested and is believed to be entirely reliable. It would be foolish to guarantee a program of this size and complexity to be free of errors. Therefore, we assume no responsibility for the failure of this software.

2. TEXT EDITOR (TED) FEATURES

The TED occupies approximately one-half the total memory space of this software. The purpose of the TED is to setup and maintain the source file by interacting with the user via various commands.

When inputting to the TED, the user has the following options:

Control H (hex 08) or
RUBOUT (hex 7F) - Deletes previous character. More than one of these may be entered to delete a number of characters

Control X (hex 18) - Deletes the entire line.

Break - Halts outputting, and waits for input of appropriate control code (part 11).

A. Commands

The TED provides 27 command functions. Each command mnemonic must begin immediately after the prompter (>). When entered, a command is not executed until a carriage return is given. Although a command mnemonic such as >PR may be several non-space characters in length, the ASSM/TED only considers the first two. For example, >PR, >PRI, >PRINT, and >PRETTY will be interpreted as the print command.

Some commands can be entered with various parameters. For example, >PRINT 10 200 will print out the text in the text file with line numbers between 10 and 200. One must separate the mnemonic and the parameters from one another by at least one space. - Do not use commas.

A description of each command follows:

>AUTO x

Automatic line numbering occurs when an x value not equal zero is entered. x specifies the increment to be added to each line number. Auto line numbering starts after one enters the first line. To prevent auto line numbering from reoccurring, enter >Au or >Au 0.

>GET Fx y

Get text file with data associated with file number x from tape or disc. The data will be loaded at line number y, or will be appended to end of the text file if the keyword APPEND is entered for y. Defaults are x=00 and y = line number 0.
Examples: >GE
>GET F13 100
>GET APPEND

>PUT Fw x y

Put text file between lines x and y to tape or disc, and assign the recorded data file number w. If w is not entered, 00 will be assumed. If x and y are not entered, the entire text file is recorded. If the letter X is entered as the parameter such as >PU X and end of file mark is recorded.

>NUMBER x y

Renumber the text file starting at line x in text file and expanding by constant y. For example to renumber the entire text file by 10, enter >NU 0 10.

>DELETE x y

Delete entries in text file between line numbers x and y. If only x is entered, only that line is deleted.

>OUTPUT Fw

Create a relocatable object file on tape deck 0 and assign file number w to the recorded data. If w is not entered 00 will be assumed. This command uses the 256 byte relocatable buffer that can be reallocated via the >SET command.

>HARD w x

Control output to hard copy output device (printer). Turn on outputting (w = SET) or turn off (w = CLEAR). The starting page number is x. This command is designed to leave a small margin at top and bottom, and provide a page number heading at the top of each page. It is designed to work with 66 line pages. An entry of >HA PAGE results in the printer advancing to the top of the next page.

>PRINT x y

Print the text file data between line number x and y on the CRT. If only x is entered, only that line is printed. If no x and y, the entire file is outputted.

>ASSEMBLE w x

Clear the label file and then assemble source in the text file starting at line number x or 0 if **x** is not entered. If w=LIST then a listing will be generated. If w=NOLIST or LIST not entered then an errors only output will be generated.

>RUN label

Run (execute) a previously assembled program. If a symbolic label is entered, the label file is searched for the starting address. The called program should contain an RTS instruction as the last executable instruction.

>LABELS

Print out the label file.

>PASS

Execute the second pass of assembly. Not required if source is all in internal memory and the .CT pseudo op is not encountered.

>FORMAT w

Format the text file (where w = SET) or clear the format feature (where w = CLEAR). Format set tabulates the text file when outputted. This lines up the various source statement fields. This feature, set or clear, does not require extra memory. Assembly output is dependent on the state of the format feature.

>DUPLICATE Fw

Duplicate files from tape 1 to tape 0 until file w. This command starts by reading the next file on tape 1 (or the disc input) and if that file is file w or an end of file (EOF) mark then it stops. If not, the file just read will be written to tape 0 (or the disc output) and then tape 1 is read again. This continues until file w or EOF is encountered.

>COPY x y z

Copy lines y thru z in the text file to just after line number x. The copied lines will all have line numbers equal x. At completion, there will be two copies of this data - one at x and the original at y.

>MOVE x y z

Move lines y thru z in the text file to just after line number x. The moved lines will all have line numbers equal x. The original lines y thru z are deleted.

>SET ts te ls le bs

If no parameters are given, the text file, label file, and relocatable buffer boundaries (addresses indicating text file start, end, label file start, end, and relocatable buffer start) will be output on first line, then on the second line the output consists of the present end of data in the text file followed with the present end of data in the label file. This command is commonly used to determine how much memory is remaining in the text file. If you are inputting hex digits for these addresses, preceed each with a '$' character.

If parameters are entered, the first two are text file start (ts) and end (te) addresses, then the label file start (ls) and end (le) addresses, and finally the relocatable buffer start address (bs).

>USER

User defined command. The ASSM/TED will transfer control to location $0003. The user routine can reenter ASSM/TED via a warm start.

>ENTER filename

Enter a filename in the disc directory. This opens a disc output file. If no filename is entered, the result is a close operation. See parts 6 and 7 for details.

>LOOKUP filename

Look up a filename in the disc directory. This opens a disc input file. If no filename is entered, the result is a close operation. See parts 6 and 7 for details.

>FIND tS1t

Find string S1. See part 10B for details.

>MANUSCRIPT w

If w = SET, line numbers are not outputted when executing the >PR command. If w = CLEAR, line numbers are outputted when the >PR command is executed. Assembly output ignores the >MA command. If manuscript is to be generated with this ASSM/TED, manuscript should be set and format clear (>MA SET,>FO CLEAR). Since the TED considers a blank line a deletion, one must enter a non printable control character to trick the TED into inserting a blank line.

>ON n

Turn on tape deck n (where n is 0 (record), or 1 (play) deck). If an n is not entered, 0 is assumed.

>OFF n

Turn off tape deck n (where n is 0 (record), or 1 (play) deck). If an n is not entered, 0 is assumed.

>CLEAR

Clear text file and turn off tape decks.

>BREAK

Break to monitor (executes BRK instruction). A return to the TED can be performed at the address immediately after the break instruction. (A control C operation does the same thing).

>n

Any entry beginning with one or more decimal digits is considered and entry/deletion of text. Details on this follows.

>EDIT tS1tS2t or EDIT n

See part 10A.

B. Entry/Deletion of Text

Source is entered in the text file by entering a line number (0-9999) followed by the text to be entered. The line number string can be one to n digits in length. If the string is greater than 4 digits in length, only the right-most 4 are considered. Text may be entered in any order but will be inserted in the text file in numerical order. This provides for assembling, printing, and recording in numerical order. Any entry consisting of a line number with no text or just spaces results in a deletion of any entry in the text file with the same number. If text is entered and a corresponding line number already exists in the text file, the text with the corresponding number is deleted and the entered text is inserted.

To delete the entire file, use the >CL command.

To delete a range of lines, use the >DE command. To edit an existing line or lines having similar characteristics, use the >ED command.

To find a string, use the >FI command. To move or copy lines use the >MO or >CO commands. To copy from input tape to output tape until a specific file, use the >DU command.

The CRT input buffer is 80 characters in length. There are 10 tab points preset at 8 character intervals. Thus, the first tab point is at the 8-th column, the second at the 16-th column, etc. Entry of control I (^I) will result in a movement to the next tab point. When inputting, the cursor may not position exactly at the tab point but will position properly when the text file is outputted via the >PR command.

Text may be entered more easily by use of the auto line numbering feature (>AU command). Any >AU x where x does not equal 0 puts the TED in the auto line number mode. To temporarily exit from this mode, type >//. To prevent auto line numbering from reoccurring every time you insert or delete, enter >AU 0.

When entering source for the assembler, one need not space over to line up the various fields. Labels are entered immediately after the line number or > when in auto line numbering. Separate each source field with one or more spaces. If the format feature is set (see >FO command), the TED will automatically line up the fields. Note: If a space is entered before the label, the TED will line up the label in the next field. This should result in an assembler error when assembled. If a control I (tab) is entered, a tab to the 8-th column is performed. These tabs are preset and can not be changed. Commands, mnemonics, and pseudo ops may be entered as upper case or lower case characters. Assembly labels may also be entered in upper or lower case but a label entered as upper case will be unique to the same label entered as lower case.

3. ASSEMBLER (ASSM) FEATURES

The ASSM scans the source program in the text file. This requires at least two passes (or scans). On the first pass, the ASSM generates a label file (or symbol table) and outputs any errors that may occur. On the second pass the ASSM creates a listing and/or object file using the label file and various other internal labels.

A third pass (via >OU) may be performed in order to generate a relocatable object file of the program in the text file. This file is recorded on tape deck 0 (record deck) and may be reloaded into the memory using the relocating loader at practically any location.

A. Source Statement Syntax

Each source statement consists of 5 fields as described below:

>line number label mnemonic operand comment

label

The first character of a label may be formed from the following characters:

@ A thru Z [\ ] ^ _

While the remaining characters which form the label may be constructed from the above characters and the following characters:

. / 0 thru 9 : ; < > ?

The label is entered immediately after the line number or prompter (>) if in the auto line numbering mode.

Mnemonic or
Pseudo Op

Separated from the label by one or more spaces and consists of a standard 6502 mnemonic of table A or pseudo op of table B.

Operand

Separated from mnemonic or pseudo op by one or more spaces and may consist of a label expression from table C and symbols which indicate the desired addressing mode from table D.

Comment

Separated from operand field by one or more spaces and is free format. A comment field begins one or more spaces past the mnemonic or pseudo op if the nature of such does not require an operand field. A free format comment field may be entered if a semicolon (;) immediately follows the line number or > if in auto line numbering mode.

Note: It is permissable to have a line with only a label. This is commonly done to assign two or more labels to the same address.

To insert a blank line, enter control I (^I).

TABLE A - 6502 Mnemonics

(For a description of each mnemonic,
consult the 6502 Software Manual)

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

TABLE B - Pseudo Ops

.BA label expression

Begin assembly at the address calculated from the label expression. This address must be defined on the first pass or an error will result and the assembly will halt.

.CT

Indicates that the source program continues on tape.

.CE

Continue assembly if errors other than !07, !04, and !17 occur. All error messages will be printed.

.LS

Set the list option so that the assembly begins printing out the source listing after the .LS on pass 2.

.LC

Clear the list option so that the assembly terminates printing the source listing after the .LC on pass 2.

.OS

Set the object store option so that object code after the .OS is stored in memory on pass 2.

.OC

Clear the object store option so that object code after the .OC is not stored in memory. This is the default option.

.MC label expression

When storing object code, move code to the address calculated from the label expression but assemble in relation to that specified by the .BA pseudo op. An undefined address results in as immediate assembly halt.

.SE label expression

Store the address calculated from the label expression in the next two memory locations. Consider this address as being an external address. Note: If a label is assigned to the .SE, it will be considered as internal.

.RC

Provide directive to relocating loader to resolve address information in the object code per relocation requirements but store code at the pre-relocated address. This condition remains in effect until a .RS pseudo op is encountered. The purpose of the .RC op is to provide the capability to store an address at a fixed location (via .SI pseudo op) which links the relocatable object code module to a fixed module.

.EJ

Eject to top of next page if >HA SET was previously entered.

.MD

Macro definition. See part 3F.

.ME

Macro end. See part 3F.

.EC

Suppress output of macro generated object code on source listing. See part 3F. This is the default state.

.ES

Output macro generated object code on source listing. See part 3F.

.DS label exp.

Define a block of storage. For example, if label exp. equated to 4, then ASSM will skip over 4 bytes. Note: the initial contents of the block of storage is undefined.

.RS

Provide directive to relocating loader to resolve address information in the object code per relocation, and store the code at the proper relocated address. This is the default condition.

.BY

Store bytes of data. Each hex, decimal, or binary byte must be separated by at least one space. An ascii string may entered by beginning and ending with apostrophes ('). Example: .BY 00 'ABCD' 47 69 'Z' $FC %1101

.SI label expression

Store the address calculated from the label expression in the next two memory locations. Consider this address as being an internal address.

label .DE label exp.

Assign the address calculated from the label expression to the label. Designate as external and put in label file. An error will result if the label is omitted.

label .DI label exp.

Assign the address calculated from the label expression to the label. Designate as internal and put in label file. An error will result if the label is omitted.

.EN

Indicates the end of the source program.

Note: Labels may be entered for any of the pseudo ops.

TABLE C - Label Expressions

A label expression must not consist of embedded spaces and is constructed from the following:

Symbolic Labels:

One to ten characters consisting of the ascii characters as previously defined.

Non-Symbolic Labels:

Decimal, hex, or binary values may be entered. If no special symbol preceeds the numerals then the ASSM assumes decimal (example: 147). If $ preceeds then hex is assumed (example $F3). If % preceeds then binary is assumed (example %11001). Leading zeros do not have to be entered. If the string is greater than 4 digits, only the rightmost 4 are considered.

Program Counter:

To indicate the current location of the program counter use the symbol =.

Arithmetic Operators:

Used to separate the above label representations:
+ addition - subtraction

Examples of some valid label expressions follow:

```
LDA  #%1101     load immediate OD
STA  *TEMP+$01     store at byte following TEMP
LDA  $471E36    load from 1E36
JMP  LOOP+C - $461
BNE  =+8 branch to current PC plus 8 bytes
```

One special label expression is A, as in ASL A. The letter A followed with a space in the operand field indicates accumulator addressing mode. Thus LDA A is an error condition since this addressing mode is not valid for the LDA mnemonic.

ASL A+$00 does not result in accumulator addressing but instead references a memory location.

TABLE D - ADDRESSING MODE FORMATS

Immediate

```
LDA #%11101  binary 0D
LDA #$F3     hex F3
LDA #F3      load value of label F3
LDA #'A      ascii A
LDA #H, label expression    hi part of the address of the label
                            expression
LDA #L, label expression    lo part of the address of the label
                            expression
```

Absolute

```
LDA label expression
```

Zero Page

```
LDA *label expression       the asterisk (*) indicates zero
                            page addressing
```

Absolute Indexed

```
LDA label expression,X
LDA label expression,Y
```

Zero Page Indexed

```
LDA *label expression,X
LDA *label expression,Y
```

Indexed Indirect

```
LDA (label expression,X
```

Indirect Indexed

```
LDA (label expression),Y
```

Indirect

```
JMP (label expression)
```

Accumulator

```
ASL A                       letter A followed with a space
                            indicates accumulator addressing
                            mode.
```

Implied

TAX	Operand field ignored
CLC	

Relative

BEQ label expression

B. Label File (or symbol Table)

A label file is constructed by the assembler and may be outputted at the end of assembly (if a .LC pseudo op was not encountered) or via the >LA command. The output consist of each label encountered in the assembly and its hex address. A label in the label file which begins with a slash (/) indicates that it was defined as an external label. All others are considered as being internal labels. When a relocatable object file is generated (via >OU command), any instruction which referenced an internal label or a label expression which consisted of at least one internal label will be tagged with special information within the relocatable object file. The relocating loader uses this information to determine if an address needs to resolved when the program is moved to another part of memory.

Conversely, instructions which referenced an external label or a label expression consisting of all external references will not be altered by the relocating loader.

At the end of the label file the number of errors which occurred and program break in the assembly will be outputted in the following format: //xxxx,yyyy,zzzz

Where xxxx is the number of errors found in decimal representation, yyyy is last address in relation to .BA, and zzzz is last address in relation to .MC.

C. Assembling not from tape

With the source program in the text file area, simply type >AS x. Assembly will begin starting at line number x. If a .CT pseudo is not encountered, both passes will be accomplished automatically. If a .CT pseudo op was encountered, the >PA command would have to be executed to perform the second pass.

D. Assembling from tape

Source for a large program may be divided into modules, entered into the text file one at a time and recorded (>PU) on tape.

At assembly, the assembler can load and assemble each module until the entire program has been assembled. This would require two passes for a complete assembly. When assembling from tape, the file indentification number assigned to the modules is ignored.

Source statements within a module and the modules themselves will be assembled in the order in which they are encountered.

The ASSM assumes that if an end of file condition is encountered before the .EN pseudo op and a .CT pseudo op had not been encountered, an error is present (!07 AT LINE xxxx)

When assembling from tape, the assembler should encounter a .CT pseudo op before the end of the first module. Two ways to accomplish this are:

1. a) Load the first module via the >GE command.
   b) This module should contain a .CT pseudo op

or

2. a) Clear the text file via the >CL command
   b) enter >9999 .CT
   9999 is entered since one may have requested any assembly beginning with a line number. This insures that the .CT gets executed.

Next ready the play deck and type >AS x. Either way the ASSM will start and stop tape deck 1 in the assembly process until the .EN pseudo op is encountered. At that point tape deck 1 is turned off, and the message READY FOR PASS 2 is outputted.

One is now in the TED mode. Rewind the tape deck (>ON 1 and >OF 1 or T1 accordingly). Perform 1 or 2 as described above and type PASS to perform the second pass. Again tape deck 1 will be turned on and off accordingly under control of the ASSM software.

E. Creating a relocatable object file (>OU)

In order to create a relocatable object file, the programmer should define those labels whose address should not be altered by the relocating loader. This is done via the .DE pseudo op. Non-symbolic labels (example: $0169) are also considered as being external. All other labels (including those defined via the .DI pseudo op) are considered as internal. Addresses associated with internal labels are altered by an offset when the program is loaded via the relocating loader.

Also, the .SE stores a two byte external address and the .SI stores a two byte internal address. Similarily the relocating loader will alter the internal address and not the external address.

An example of an external address would be the calls to your ROM monitor or any location whose address remains the same no matter where the program is located. Locations in zero page are usually defined as external addresses. Expressions consisting of internal and external labels will be combined and considered an internal address. A label expression consisting entirely of external labels will be combined and considered as external.

To record a relocatable object file, insert a blank tape in tape deck 0 and ready. If the entire source program is in memory, simply type >OU.

If the source program is on tape, ready as described in 1 and 2 in part 3D and thentype >OU. The ASSM will turn both tape decks on and off until the end of assembly. The relocatable object file will be recorded on the tape in deck 0.

After the relocatable object file has been recorded, record an end of file mark via the >PU X command.

F. Macros

ASSM/TED provides a macro capability. A macro is essentially a facility in which one line of source code can represent a function consisting of many instruction sequences. For example, the 6502 instruction set does not have an instruction to increment a double byte memory location. A macro could be written to perform this operation and represented as INCD (VALUE.1). This macro would appear in your assembly language listing in the mnemonic field similar to the following:

```
BNE    SKIP
NOP
 {
INCD  (VALUE.1) ; INCREMENT DOUBLE
LDA    TEMP
 }
```

Before a macro can be used, it must be defined in order for ASSM to process it. A macro is defined via the .MD (macro definition) pseudo op. Its form is :

```
!!!label    .MD    (L1 L2  ... Ln)
```

Where label is the name of the macro (!!! must preceed the label), and L1, L2,..., Ln are dummy variables used for replacement with the expansion variables. These variables should be separated using spaces, do not use commas.

To terminate the definition of a macro, use the .ME (macro end pseudo op).

For example, the definition of the INCD (increment double byte) macro could be as follows:

```
!!!INCD     .MD     (LOC)    ; INCREMENT DOUBLE
            INC      LOC
            BNE     SKIP
            INC     LOC+1
SKIP        .ME
```

This is a possible definition for INCD. The assembler will not produce object code until there is a call for expansion. Note: A call for expansion occurs when you enter the macro name along with its parameters in the mnemonic field as INCD (TEMP) or INCD (COUNT) or INCD (COUNT+2) or any other labels or expressions you may choose.

Note:In the expansion of INCD, code is not being generated which increments the variable LOC but instead code for the associated variable in the call for expansion.

If you tried to expand INCD as described above more than once, you will get a !06 error message. This is a duplicate label error and it would result because of the label SKIP occurring in the first expansion and again in the second expansion.

There is a way to get around this and it has to do with making the label SKIP appear unique with each expansion. This is accomplished by rewriting the INCD macro as follows:

```
!!!INCD     .MD     (LOC)     ;INCREMENT DOUBLE
            INC      LOC
            BNE      ...SKIP
            INC      LOC+1
...SKIP     .ME
```

The only difference is ...SKIP is substituted for .SKIP. What the ASSM does is to assign each macro expansion a unique macro sequence number (2**16 maximum macros in each file). If the label begins with ... the ASSM will assign the macro sequence number to the label. Thus, since each expansion of this macro gets a unique sequence number, the labels will be unique and the !06 error will not occur.

If the label ...SKIP also occurred in another macro definition, no !06 error will occur in its expansion if they are not nested. If you nest macros (i.e. one macro expands another), you may get a !06 error if each definition uses the ...SKIP label.

The reason this may occur is that as one macro expands another in a nest, they each get sequentially assigned macro sequence numbers. As the macros work out of the nest, the macro sequence numbers are decremented until the top of the nest. Then as futher macros are expanded, the sequence numbers are again incremented. The end result is that it is possible for a nested macro to have the same sequence number as one not nested or one at a different level in another nest. Therefore if you nest macros, it is suggested that you use different labels in each macro definition.

Some futher notes on macros are:

1) The macro definition must occur before the expansion.

2) The macro definition must occur in each file that references it. Each file is assigned a unique file sequence number (2**16 maximum files in each assembly) which is assigned to each macro name. Thus the same macro can appear in more than one file without causing a !06 error. If a macro with the same name is defined twice in the same file, then the !06 error will occur.

3) Macros may be nested up to 32 levels. This is a limitation because there is only so much memory left for use in the stack.

4) If a macro has more than one parameter, the parameters should be separated using spaces - do not use commas.

5) The number of dummy parameters in the macro definition must match exactly the number of parameters in the call for expansion.

6) The dummy parameters in the macro definition must be symbolic labels. The parameters in the expansion may be symbolic or nonsymbolic label expressions.

7) If the .ES pseudo op is entered, object code generated by the macro expansion will be output in the source listing. Also, comment lines within the macro definition will be output as blank lines during expansion. Conversely, if .EC was entered, only the line which contained the macro call will be output in the source listing.

8) A macro name may not be the same as a 6502 mnemonic, pseudo op, or conditional assembly operator.

G. Conditional Assembly

ASSM also provides a conditional assembly facility to conditionally direct the assembler to assemble certain portions of your program and not other portions. For example, assume you have written a CRT controller program which can provide either 40,64 or 80 characters per line. Instead of having to keep 3 different copies of the program you could use the ASSM conditional assembly feature to assemble code concerned with one of the character densities.

Before we continue with this example, lets describe the Conditional assembly operators:

IFE label exp.

If the label expression equates to a zero quantity, then assemble to end of control block.

IFN label exp.

If the label expression equates to quantity not equal to zero, then assemble to end of control block.

IFP label exp.

If the label expression equates to a positive quantity (or 0000), then assemble to end of control block.

IFM label exp.

If the label expression equates to a negative (minus) quantity, then assemble to end of control block.

***

Three asterisks in the mnemonic field indicates the end of the control block.

SET label=label exp.

Set the previously defined label to the quantity calculated from the label expression.

Note: All label expressions are equated using 16 - bit precision arithmetic.

Going back to the CRT controller software example, a possible arrangement of the program is as follows:

```
CHAR.LINE .DE 40
           {
          IFE CHAR.LINE-40
;CODE FOR 40 CHAR./LINE
           {
          ***
          IFE CHAR.LINE-64
;CODE FOR  64 CHAR./LINE
           {
          ***
          IFE CHAR.LINE-80
;CODE FOR  80 CHAR./LINE
           {
          ***
;COMMON CODE
           {
           {
```

Shown is the arrangement which would assemble code associated with 40 characters per line since CHAR.LINE is defined as equal 40. If you wanted to assemble for 80 characters, simply define CHAR.LINE as equal 80.

Conditional assembly can also be incorporated within macro definitions. A very powerful use is with a macro you don't want completely expanded each time it is referenced. For example, assume you wrote a macro to do a sort on some data. It could be defined as follows:

```
EXPAND    .DE  0
!!!SORT   .MD
          IFN  EXPAND
          JSR  SORT.CALL  ;CALL SORT
          ***
          IFE  EXPAND
          JSR  SORT.CALL
          JMP  ...ABC
;SORT CODE FOLLOWS
SORT.CALL
           {
           {
          RTS
...ABC    SET EXPAND=1
          ***
          !ME
```

In this example, EXPAND is initially set to 0. When the macro is expanded for the first time, EXPAND equals 0 and the code at SORT.CALL will be assembled along with a JSR to and a JMP around the sort subroutine. Also the first expansion sets EXPAND to 1. On each suceeding expansion, only a JSR instruction will be assembled since EXPAND equals 1. Using conditional assembly in this example resulted in more efficient memory utilization over an equivalent macro expansion without conditional assembly.

H. Default Parameters on entry to ASSM

- Assumes not assembling from tape (otherwise use .CT)
- Does not store object code in memory (otherwise use .OS)
- Begins assembly at $0200 (otherwise use .BA)
- Output listing set (otherwise use .LC)
- Stops assembly on errors (otherwise use .CE)
- Stores object code beginning at $0200 unless a .BA or .MC is encountered and if .OS is present.
- Object code generated by macros does not appear on the assembly listing (i.e. default is .EC).

4. EXAMPLES

A. Listing illustrating text entry

An example of the printout which occurs when inputting text in the text file follows:

```
>FORMAT SET
>AUTO 10
>100;THIS PROGRAM ADDS 06 TO REGISTER X
0110>START TXA
0120>  CLC
0130>  CLD
0140>  ADC #6
0150>END RTS
0160> .EN
0170>//
>141  TAX
0151>//
```

Note the use of // to terminate the auto line numbering. Auto line numbering can be restarted by simply entering the line number where insertion is to begin. To prevent auto line numbering, simply type >AU or >AU 0.

B. Output listing from ASSM

Listing 1 is a source listing output of a program which provides a formatted hex dump of a block of memory. It is presently configured for TIM based systems but can be easily modified for other systems.

5. USING THE RELOCATING LOADER

A source listing of the relocating loader (listing 2) is provided. The relocating loader is not part of the ASSM/TED program body, and the user will have to enter it via the listing.

If you prefer to have the loader to reside in some other part of memory, you should enter the source into the text file, assemble, and then create a relocatable object file on tape.

To record a program in relocatable format, first assemble (without a .OS pseudo op) the program at location 0000 (.BA $0). Next create a relocatable object file via the >OU command. Terminate the relocatable object file with an end of file mark via the >PU X command. To reload a program in relocatable format, first enter the address where you want the program to reside in memory locations 00E0 (lo) and 00E1 (hi), the modules file number in 0110, and then execute.

When executing the relocating loader, if an error or an end of file mark is detected, a break (BRK) instruction will be executed so as to return to your monitor. The contents of register A indicates the following:

00 good load
EE error in loading

All programs to be created in relocatable format should be assembled at $0000. This is because the offset put in 00E0 and 00E1 before execution is added to each internal address by the loader in order to resolve addresses while relocating the program. If the program was originated at say 1000, then one would have to enter F200 as the offset in order to relocate to 0200 (i.e. F200+1000= 0200). This is somewhat more confusing than an assembly beginning 0000.

In addition to the program memory space, the relocating loader uses the following memory locations.

00C8-00C9, 00DC-00E1
0110, 011E-0121, 017A-0184

Plus other stack area for subroutine control.

6. CONFIGURE ASSM/TED FOR DISC OPERATION

ASSM/TED provides the user with four 2-byte address vectors for linkage to your disc operating system (DOS). They are:

DISC1 #FD,#F1

Address vector to your DOS (or patch to DOS) which accepts the output data filename beginning at $0135,Y. The user provided patch should accept filename characters by incrementing R(Y) until a space is encountered. If R(Y)=50 hex then your DOS should instead treat this as a CLOSE output file operation.

DISC2 #F2,#F3

Address vector to your DOS (or patch to DOS) which accepts the input data file name beginning at $0135,Y. The user provided patch should accept filename characters by incrementing R(Y) until a space is encountered. If R(Y)=50 hex then your DOS should instead treat this as a CLOSE input file operation.

DISCI.VEC #F6,#F7

Vector to your DOS (or patch) indicating that data is to be conditionally loaded into memory defined as follows:

LOAD/NO -if=1 then enter in memory.
($123) if=0 then get from disc but don't move to memory.
        This is required to skip over files not selected.

START.ADD - start address of memory.
 ($124-125)

END.ADD - end address of memory.
($126-127)

DISCO.VEC $F4, $F5

Vector to your DOS (or patch)indicating that data in memory range START.ADD thru END.ADD is to be stored on disc. LOAD/NO should be ignored.

7. USING ASSM/TED WITH DISC

Before operating with the disc, the user should set up the address vectors as described in part 6. This could be done by executing user provided code using the >RUN command, or simply manually entering address vectors using your system monitor.

There are two commands which determine if data is to input or output from tape or disc. They are:

>ENTER

Enter in disc directory. A vector thru DISC1 is performed. If entered with a filename then an open of the output file is performed. At this point all output normally going to tape will go through vector DISCO.VEC. If no parameters are entered, when your DOS should assume a close operation. At this point any output will be to tape.

>LOOKUP

Lookup in disc directory. A vector thru DISC2 is performed. If entered with a filename then an open of the input file is performed. At this point all input normally read from tape will go through vector DISCI.VEC. If no parameters are entered, then your DOS should assume a close operation. At this point any input will be from tape.

8. ERROR CODES

An error message of the form !xx AT LINE yyyy/zz where xx is the error code, yyyy is the line number, and zz is the file number will be outputted if an error occurs. Sometimes an error message will output an invalid line number. This occurs when the error is on a non-existant line such as an illegal command input.

The following is a list of error codes not specifically related to macros:

ERROR CODE	
17	Checksum error on tape load.
16	Illegal tape deck number.
15	Syntax error in >ED command.
12	Command syntax error or out of range error.
11	Missing parameter in >NU command.
10	Overflow in line # renumbering. CAUTION--You should properly renumber the text file for proper command operations.
0F	Overflow in text file - line not inserted.
0E	Overflow in label file - label not inserted.
0D	Expected hex characters, found none.
0C	Illegal character in label.
0B	Unimplemented addressing mode.
0A	Error in or no operand.
09	Found illegal character in decimal string.
08	Underfined label (may be illegal label).
07	.EN pseudo op missing.
06	Duplicate label
05	Label missing in .DE or .DI pseudo op.
04	.BA or .MC Operand Undefined.
03	Illegal pseudo op.
02	Illegal mnemonic.
01	Branch out of range.
00	Not a zero page address.
ED	Error in command input.

The following is a list of error codes that are specifically related to macros and condition assembly:

ERROR CODE

2F	Overflow in file sequence count (2**16 max.)
2E	Overflow in number of macros (2**16 max.)
2B	.ME without associated .MD
2A	Non symbolic label in SET
29	Illegal nested definition
27	Macro definition overlaps file boundary
26	Duplicate macro definition
25	Quantity parms mismatch or illegal characters
24	Too many nested macros (32 max.)
23	Macro definition not complete at .EN
22	Conditional suppress set at .EN
21	Macro in expand state at .EN
20	Attempt expansion before definition

9. FILE NUMBERS

Information to be recorded on tape via the >PU and >OU commands may be assigned a file indentification number to distinquish between other files of information. A file number is a decimal number between 0 and 99. To enter a file number as a parameter in the >PU,>OU, or >GE commands, begin with the letter 'F' followed by the file number. Examples are F0, F17, F6, etc. If no file number is entered with the >PU and >OU commands, file number 0 will be assigned by default.

When loading, all files encountered will result in the outputting of their associated file numbers and file length in bytes. The loaded file has, in addition, the memory range of the location of the loaded data. Example:

```
>GET F17
F00 01A3
F67 0847
F17 0F93  0200-1193
>
```

An end of file mark may be recorded via the >PU X command to indicate the end of a group of files. If an end of file mark is encountered when loading, FEE will be outputted and a return to the command mode will be performed.

10. STRING SEARCH AND REPLACE COMMANDS

A. Edit command

A powerful string search and replace, and line edit capability is provided via the >EDIT command to easily make changes in the text file. Use form 1 to string search and replace, and form 2 to edit a particular line.

Form 1

```
                          Δ
                          #
>EDIT tS1tS2t  %d  *  x  y
```

Where:
- t — is a non-numeric, non-space terminator
- S1 — is string to search for.
- S2 — is string to replace S1.
- d — is don't care character. Preceed with % character to change the don't care, else don't care character will be % by default.
- * — indicates to interact with user via subcommands before replacing S1.
- # — indicates to alter but provide no printout.
- Δ — (space) indicates to alter and provide printout.
- x — line number start in text file.
- y — line number end in text file.

asterisk (*) prompted subcommands:
- A — alter field accordingly.
- D — delete entire line.
- M — move to next field - don't alter.
- S — skip line - don't alter.
- X — exit >ED command
- ^F — (control F) - enter form 2

defaults
- d = %
- x = 0
- y = 9999
- Δ = (space) print all lines altered

For example, to replace all occurances of the label LOOP with the label START between lines 100 and 600, enter:

```
>EDIT   .LOOP.START. 100  600
```

To simple delete all occurrances of LOOP, enter:

```
>EDIT   .LOOP..   100  600
```

Use the *,#,and Δ as described.

The period was used in the above examples as the terminator but any non-numeric character may be used.

AT the end of the >EDIT operation, the number of occurrances of the string will be output as //xxxx where xxxx is a decimal quantity.

Form 2

>EDIT n

Where: n is line number (0-9999) of line to be edited.

subcommands:
- ^F (control F) - Find user specified character.
- cr (carriage return) - retain any remaining part of a line.
- ^D (control D) - delete any remaining part of line.
- ^H delete a character.

For example, to change LDA to LDY in the following line

```
    LOOP1   LDA  #L,CRTBUFFER  LOAD FROM BUFFER
```

Type ^F followed with A, then ^H, then Y, and then terminate line with a carriage return.

The corrected line will then be outputted and entered in the text file.

B. Find Command

If you want to just find certain occurrances of a particular string, use the >FIND command. Its form is:

$$\text{>FIND } t\,Sl\,t\,\overset{\Delta}{\#}\;x\;y$$

Where: t, Sl, #, Δ , x, y are as defined in part 10.A.

For example, >FIND /LDA/ will output all occurrances of the string LDA in the text file.

AT the end of the >FIND operation, the number of occurrances of the string will be cutput as //xxxx where xxxx is a decimal quantity.

A unique use of this command is to count the number of characters in the text file (excluding line numbers). The form for this is:

>FIND /%/#

---

11. CONTROL CODES

Ascii characters whose hex value is between hex 00 and 20 are normally non-printing characters. With a few exceptions, these characters will be output in the following manner: ^c where c is the associated printable character if hex 40 was added to its value. For example, ascii 03 will be output as ^C, 18 as ^X, etc.

In addition, some of these control codes have special functions in ASSM/TED.

Control codes which have special functions are:

^@	*	null (hex00)
^B		go to Basic
^C		go to Monitor (executes BRK instruction)
^D		delete - used by >EDIT
^F		find - used by >EDIT
^G	*	bell
^H	*	backspace (delete previous character)
^I	*	horizontal tab
^J	*	linefeed
^M	*	carriage return
^O		continue processing but suppress output to CRT
^Q	*	continue after break operation
^T		(as ^Tn) toggle Motor Control on deck n
^X		delete entire line entered
^Y		jump to location $0000. Return via warm start
^Z		terminate processing and go to ">" mode.
^[	*	escape character

* = Non-printing control character

12. SPECIAL NOTES

. In addition to the program memory space the ASSM/TED uses the following memory locations

0100 - up depending on type of function
00B9 - 00F8

plus other stack area for subroutine control. The CRT buffer is in locations 0135 - 0185

. Keep the cover closed on the tape deck as this keeps the cassette cartridge stable.

. When entering source modules (without .EN) you can perform a short text on the module by assembling the module while in the text file and looking for the !07 error. If other error messages occur, you have errors in the module. This short test is not a complete test but does check to make sure you have lined up the fields properly, not entered duplicate labels within the module, or entered illegal mnemonics or addressing modes.

- A 64 character/line (or greater) output device should be used with this program when printing an assembly listing in order to provide a neat printout without foldover to next line.

- Any keyboard input greater than 80 characters in length will be automatically inserted in the text file without the user having to enter a carriage return.

- Locations $00D5 (lo) and $00D6 (hi) contain the address of the present end of the label file. This address +2 should contain a zero (a forward pointer).

- Locations $00D3 (lo) and $00D4 (hi) contain the address of the present end of the text file. This address +2 should contain a zero (a forward pointer).

- The ASSM/TED and the Relocating Loader were designed so that they will execute in RAM or ROM.

- To find the address of an entry in the text file, output the line via the PR command, issue the BR command, and then get the contents of memory location 00DD, 00DE. This is an address which points to the end of the outputted line.

LISTINGS

1. Hex dump program
2. Source listing of relocating loader

TABLES

A) 6502 Mnemonics
B) Pseudo ops
C) Label expression
D) Addressing Modes

Listing 1

```
>ASSEMBLE LIST

                    0100 ;THIS PROGRAM IS PROVIDED AS AN EXAMPLE OF A PROGRAM
                    0200 ;WHICH USES VARIOUS FEATURES DESCRIBED IN THIS MANUAL.
                    0300 ;THIS PROGRAM OUTUTS A HEX LISTING
                    0400 ;
                    0500              .BA $0
                    0600              .OC
                    0700 CRLF         .DE $728A
                    0800 TBYT         .DE $72B1
                    0900 SPACE        .DE $7377
                    1000 SPACE2       .DE $7374
                    1100 COUNT        .DI END+OF+PGM
                    1200 ADDRS        .DE $0
                    1300 END          .DE $010A
                    1400 ;
                    1500 ;AT START, SET PRINTER TO BEGIN PRINTING ON 3-RD LINE
                    1600 ;ON 3-RD LINE
                    1700 ;START ADDRESS IN ADDRS
                    1800 ;END ADDRESS IN END
                    1900 ;
                    2000 ;
                    2100 ;MACRO DEFINITION -- INCREMENT DOUBLE BYTE
                    2200              .ES
                    2300 ;
                    2400 !!!INCD      .MD (X)
                    2500              INC *X
                    2600              BNE ...SKIP
                    2700              INC *X+1
                    2800 ...SKIP      .ME
                    2900 ;
                    3000 ;
0000- A9 00         3100 BEGIN        LDA #$00
0002- AA            3200              TAX
0003- 8D 5B 00      3300              STA COUNT
0006- 20 8A 72      3400 NEXT+LN      JSR CRLF
0009- AD 5B 00      3500              LDA COUNT
                    3600 ;DEC. 60 LINES PER PAGE
000C- C9 3C         3700              CMP #$3C        ;DECIMAL 60
000E- 90 0D         3800              BCC SKIP
0010- A9 00         3900              LDA #$00
0012- 8D 5B 00      4000              STA COUNT
                    4100 ;ISSUE 6 CRLF'S AT END OF 60-TH LINE TO GO
                    4200 ;TO NEXT PAGE
0015- A0 06         4300              LDY #$06
0017- 20 8A 72      4400 LOOP3        JSR CRLF
001A- 88            4500              DEY
001B- D0 FA         4600              BNE LOOP3
001D- A0 10         4700 SKIP         LDY #$10
001F- A5 01         4800              LDA *ADDRS+$1
0021- 20 B1 72      4900              JSR TBYT
0024- A5 00         5000              LDA *ADDRS+$0
0026- 20 B1 72      5100              JSR TBYT
0029- 20 74 73      5200              JSR SPACE2
                    5300 ;NOW ADDRESS IS OUTPUTTED
002C- A1 00         5400 LOOP2        LDA (ADDRS,X)
002E- 20 B1 72      5500              JSR TBYT
```

```
0031- A5 01     5600             LDA ↑ADDRS+$01
0033- CD 0B 01  5700             CMP END+$1
0036- 90 11     5800             BCC NOT↑END
0038- F0 08     5900             BEQ CKLO
003A- 20 8A 72  6000 END↑PGM     JSR CRLF
003D- 00        6100             BRK
003E- EA        6200             NOP
003F- 4C 00 00  6300             JMP BEGIN
0042- A5 00     6400 CKLO        LDA ↑ADDRS+$0
0044- CD 0A 01  6500             CMP END+$0
0047- B0 F1     6600             BCS END↑PGM
                6700 NOT↑END     INCD (ADDRS)          ;INCREM. ADDRS

0049- E6 00
004B- D0 02
004D- E6 01

004F- 20 77 73  6800             JSR SPACE
0052- 88        6900             DEY  ;R(Y)=BYTE COUNTER
0053- D0 D7     7000             BNE LOOP2
0055- EE 5B 00  7100             INC COUNT
0058- 4C 06 00  7200             JMP NEXT↑LN
                7300 END↑OF↑PGM  .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/CRLF=728A              /TBYT=72B1              /SPACE=7377
/SPACE2=7374            COUNT=005B              /ADDRS=0000
/END=010A               BEGIN=0000              NEXT↑LN=0006
LOOP3=0017              SKIP=001D               LOOP2=002C
END↑PGM=003A            CKLO=0042               NOT↑END=0049
X=0000                  END↑OF↑PGM=005B
//0000,005B,005B
>
```

```
                    0060              .EJ
                    0070 ;*************************************
                    0080 ;*                                   *
                    0090 ;* COPYRIGHT 1980 BY U.O.SCHRODER    *
                    0100 ;*                                   *
                    0110 ;*************************************
                    0120 ;
                    0130 ;PURPOSE OF THIS PART:
                    0140 ;1.ASSIST YOU TO IMPLEMENT THE ASSM/TED ON YOUR SYSTEM
                    0150 ;2.WARN YOU ON SOME DIFFICULTIES THAT MIGHT OCCUR
                    0160 ;   USING THE ASSM/TED
                    0170 ;
                    0180 ;.................................
                    0190 ;
                    0200              .BA $3C27
3C27- 4C E8 3F      0210              JMP CRT/OUTPUT
                    0220 ;
                    0230              .BA $3FD6
                    0240 ;CRT/OUTPUT
                    0250 ;PURPOSE: OUTPUT A CHARAKTER TO THE OUTPUTDEVICE
                    0260 ;         THE ASCII-CHARAKTER IS IN A.
                    0270 ;         AT RETURN THE CONTENT OF ALL REGISTERS
                    0280 ;         MAY BE CHANGED.
                    0290 ;
3FD6- A9 1F         0300 CRT/CR.FND LDA #$1F      ;DELAYTIME AFTER CR
3FD8- 20 87 3F      0310            JSR DELAY
3FDB- A9 0A         0320            LDA #$0A      ;OUTPUT LNFD AFTER CR
3FDD- 20 E8 3F      0330            JSR CRT/OUTPUT
3FE0- A9 1E         0340            LDA #$1E      ;DELAYTIME AFTER LNFD
3FE2- 20 87 3F      0350            JSR DELAY
3FE5- EA            0360            NOP
3FE6- EA            0370            NOP
3FE7- 60            0380            RTS
3FE8- 48            0390 CRT/OUTPUT PHA
3FE9- EA            0400            NOP  ;INSERT LDY $011F
3FEA- EA            0410            NOP  ;       BNE CRT/OUT2
3FEB- EA            0420            NOP  ;IF YOU USE ONLY ONE
3FEC- EA            0430            NOP  ;HARDCOPY DEVICE FOR INPUT/OUTPUT
3FED- EA            0440            NOP
3FEE- 20 A0 1E      0450            JSR KIM/OUTCH
3FF1- 68            0460 CRT/OUT2   PLA
3FF2- C9 0D         0470            CMP #$0D      ;CARRIAGE RETURN NEEDS
3FF4- F0 E0         0480            BEQ CRT/CR.FND
3FF6- 60            0490            RTS
                    0500 ;.................................
                    0510 ;
                    0520 ;KEYBOARD INPUT
                    0530 ;THE KIMVERSION ASSUMES THAT INPUT WILL
                    0540 ;BE ECHOED BY HARDWARE. IF YOU CAN PREVENT
                    0550 ;THE ECHO, PLEASE INSERT JSR $3BF1 AT 3C9F
                    0560              .BA $3C89
3C89- 20 5A 1E      0570              JSR KIM/GETCH        ;RETUR CHAR IN (A)
                    0580 ;
                    0590              .BA $3C9F
                    0600 ;
3C9F- EA            0610              NOP  ;CHANGE THIS INTO JSR $3BF1
3CA0- EA            0620              NOP  ;IF IT IS POSSIBLE FOR YOU
3CA1- EA            0630              NOP  ;TO PREVENT ECHOING THE INPUTCHAR.
```

```
                    0640 ;THE BREAK-TEST ROUTINE IS ENTERED ONLY IMMEDIATLY
                    0650 ;AFTER PRINTING A CARRIAGE RETURN.
                    0660 ;
                    0670 ;    <<<<<<   WARNING   >>>>>>
                    0680 ;
                    0690 ;IF YOU DON'T WAIT ON THE PROMPTING CHATAKTER
                    0700 ;AND ENTER ANY KEY WHILE THE CARRIAGE RETURN
                    0710 ;IS PRINTED, THAT KEY WILL BE INTERPRETED
                    0720 ;AS A BREAK!!!  EVERY KEY OR COMMAND THEREAFTER
                    0730 ;WILL BE IGNORED - EXCEPT SOME CONTROL KEYS!!!
                    0740 ;TO RECOVER PRESS CONTROL-Q OR CONTROL-Z
                    0750 ;UNTIL YOU GET THE PROMPTING CHARAKTER AGAIN.
                    0760 ;
                    0770 ;................................
                    0780 ;
                    0790            .BA $3BD2     ;ONLY ONE CALL!!!
3BD2- 20 77 3F      0800            JSR BREAKTEST         ;CARRY=1=BREAK
                    0810 ;
                    0820            .BA $3F77
                    0830 ;BREAKTEST
                    0840 ;PURPOSE:      DETECT IF ANY KEY IS DOWN WHILE
                    0850 ;              PRINTING CARRIAGE-RETURN
                    0860 ;EXPLANATION: INPUT ON KIM IS ON THE
                    0870 ;              MOST SIGNIFICANT BIT OF $1740
                    0880 ;
3F77- 2C 40 17      0890 BREAKTEST  BIT $1740
3F7A- 18            0900            CLC  ;CARRY=0=NO BREAK
3F7B- 30 09         0910            BMI NO/BREAK          ;MSB=1=NO INPUT
3F7D- 2C 40 17      0920 BREAK/WAIT BIT $1740     ;WAIT UNTIL END OF BREAK
3F80- 10 FB         0930            BPL BREAK/WAIT
3F82- 20 5A 1E      0940            JSR KIM/GETCH         ;IGNORE KEY AFTER BREAK
3F85- 38            0950            SEC  ;CARRY=1=BREAK
3F86- 60            0960 NO/BREAK   RTS
                    0970 ;
                    0980 ;................................
                    0990 ;
                    1000 ;DELAY
                    1010 ;PURPOSE: DELAY TIME ACCORDING TO CONTENT OF A.
                    1020 ;         DURATION IS (A)*(A)*(A),  WITH   2 < A < $FF
                    1030 ;         USED ONLY BY CRT/CR.FND
3F87- 48            1040 DELAY      PHA
3F88- 48            1050 DELAY.2    PHA
3F89- E9 01         1060 DELAY.4    SBC #1
3F8B- D0 FC         1070            BNE DELAY.4
3F8D- 68            1080            PLA
3F8E- E9 01         1090            SBC #1
3F90- D0 F6         1100            BNE DELAY.2
3F92- 68            1110            PLA
3F93- E9 01         1120            SBC #1
3F95- D0 F0         1130            BNE DELAY
3F97- 60            1140            RTS
                    1150 ;
                    1160 ;................................
                    1170 ;
                    1180 ;
                    1190 KIM/GETCH  .DE $1E5A     ;USED TWICE
                    1200 KIM/OUTCH  .DE $1EA0     ;USED ONE TIME
```

```
                    1210            .EJ
                    1220 ;THE BREAK IS USED ONCE AT 20A9 TO EXIT TED/ASSM
                    1230 ;IF YOU DON'T LIKE THIS WAY OF EXIT
                    1240 ;CHANGE THIS BREAK INTO A JUMP TO ???
                    1250 ;
                    1270 ;ADDRESS WHERE A SUBROUTINE IS CALLED
                    1280 ;TO INIT THE BREAK-VEKTOR ON THE KIM
                    1290            .BA $202B
202B- 20 A6 3F      1300            JSR BRK.VCT.IN IT
                    1310 ;
                    1320            .BA $3FA6
3FA6- A9 00         1330 BRK.VCT.IN LDA #$00      ;$1C00 IS BREAK-ENTRY ON KIM
3FA8- 8D FE 17      1340            STA $17FE     ;KIM DOES JMP ($17FE) AT BREAK
3FAB- A9 1C         1350            LDA #$1C
3FAD- 8D FF 17      1360            STA $17FF
                    1370 ;
                    1380 ;INIT OF I/O PORT FOR CASSETTE
                    1390 ;
3FB0- AD 03 17      1400            LDA $1703
3FB3- 09 0B         1410            ORA #%00001011
3FB5- 8D 03 17      1420            STA $1703
3FB8- 60            1430            RTS
                    1440 ;
                    1450 ;
                    1460 ;.................................
                    1470 ;
                    1480 ;TABLE OF MEMORY-USE AT 362C
                    1490 ;
                    1500            .BA $362C
                    1510 ;
                    1520 ;          .SI TEXTBUF/START
                    1530 ;          .SI TEXTBUF/END ;ASSM/TED USES 3 BYTES MORE!!
                    1540 ;
                    1550 ;          .SI SYMTAB/START
                    1560 ;          .SI SYMTAB/END  ;ASSM/TED USES 3 BYTES MORE
                    1570 ;
                    1580 ;          .SI RELOC/BUF   ;256 BYTE BUFFER
```

```
                    1590              .EJ
                    1600 ;EXAMPLES OF HARDCOPY DRIVER FOR PRINTER.
                    1610 ;NOTE: FOR EACH NEW LINE THE SUBROUTINE HAS TO RETURN
                    1620 ;       ONE BYTE $0A FOR THE LINECOUNTER ROUTINE.
                    1630 ;        REGISTERS X,Y MAY BE CHANGED.
                    1640 ;
                    1650 HARDCOPY     .DE 0000      ;FOR REFERENCE ONLY
                    1660 ;
                    1670 ;
                    1680 ;.................................
                    1690 ;
                    1700 ;PRINTER WITH AUTO-LINEFEED
                    1710 ;
                    1720              .BA $38D4
38D4- 20 68 41      1730              JSR HARDCOPY/A
                    1740 ;
                    1750              .BA $4168     ;OR ELSEWHERE
4168- 48            1760 HARDCOPY/A   PHA
4169- 20 00 00      1770              JSR HARDCOPY
416C- 68            1780              PLA
416D- C9 0D         1790              CMP #$0D
416F- D0 02         1800              BNE HC/RTS    ;ON CARRIAGE RETURN
4171- A9 0A         1810              LDA #$0A      ;CHANGE $0D INTO $0A
4173- 60            1820 HC/RTS       RTS
                    1830 END/A
                    1840 ;
                    1850 ;CHANGE START OF TEXTBUFFER
                    1860              .BA $362C
362C- 74 41         1870              .SI END/A
                    1880 ;
                    1890 ;
                    1900 ;
                    1910 ;.................................
                    1920 ;
                    1930 ;PRINTER WITHOUT AUTO-LINEFEED
                    1940 ;
                    1950              .BA $38D4
38D4- 20 6A 41      1960              JSR HARDCOPY/B
                    1970 ;
                    1980              .BA $4168     ;OR ELSEWHERE
4168- A9 0A         1990 DO/LINFEED   LDA #$0A
416A- 48            2000 HARDCOPY/B   PHA
416B- 20 00 00      2010              JSR HARDCOPY
416E- 68            2020              PLA
416F- C9 0D         2030              CMP #$0D      ;ON CAR.RET.: ADD LNFD
4171- F0 F5         2040              BEQ DO/LINFEED
4173- 60            2050              RTS
                    2060 END/B
                    2070 ;
                    2080 ;CHANGE START OF TEXTBUFFER
                    2090              .BA $362C
362C- 74 41         2100              .SI END/B
```

```
                    2110            .EJ
                    2120 ;MODIFICATIONS IN THE CASSETTE ROUTINES SUPPLIED
                    2130 ;
                    2140 ;................................
                    2150 ;
                    2160 C/PORT     .DE $1702
                    2170 ;
                    2180 ;CHANGED IN/PORT ROUTINE OF CASSETTE PROGRAM
                    2190 ;
                    2200            .BA $415B
  415B- AD 02 17    2210 IN/PORT    LDA C/PORT
  415E- 29 FF       2220            AND #$FF
  4160- 29 04       2230            AND #%00000100      ;MASK ALL BUT BIT 2
  4162- 60          2240            RTS
                    2250 ;
                    2260 ;................................
                    2270 ;
                    2280 ;VERSION IF YOUR CASSETTERECORDER INVERTS DATA
                    2290 ;
                    2300            .BA $415B
  415B- AD 02 17    2310 IN/PORT    LDA C/PORT
  415E- 49 FF       2320            EOR #$FF      ;SOFTWARE CAN INVERT TO!!
  4160- 29 04       2330            AND #%00000100      ;MASK ALL BUT BIT 2
  4162- 60          2340            RTS
                    2350 ;
                    2360 ;................................
                    2370 ;
                    2380 ;   <<<<<<   HARDWARE NOTES   >>>>>>
                    2390 ;
                    2400 ;1. DO NOT CONNECT THE REMOTE-CONTROL OR EARPHONE
                    2410 ;   DIRECTLY TO THE APPLICATION-CONNECTOR OR PIA.
                    2420 ;   IT MIGHT DESTROY THE PIA!!!
                    2430 ;2. USING BITS OF THE B-PORT AS OUTPUT - NOTE THIS:
                    2440 ;     READING A OUTPUTBIT WILL NOT ALWAYS REFLEKT THE
                    2450 ;     ACTUAL VALUE THAT IS OUTPUTTED. READING THE
                    2460 ;     B-PORT RETURNS THE VALUE THAT IS READ ON
                    2470 ;     THE OUTPUT PINS - SO EXCESSIVE LOADING A PORT
                    2480 ;     TO GROUND WILL ALWAYS RETURN ZERO.
                    2490 ;  CONCLUSION: DO NOT EXCESSIVE LOAD ANY BIT.
                    2500 ;
                    2510 ;U.O.SCHRODER
                    2520 ;
                    2530            .EN
```

LABEL FILE: ↑ / = EXTERNAL ←

```
CRT/CR.FND=3FD6          CRT/OUTPUT=3FE8          CRT/OUT2=3FF1
BREAKTEST=3F77           BREAK/WAIT=3F7D          NO/BREAK=3F86
DELAY=3F87               DELAY.2=3F88             DELAY.4=3F89
/KIM/GETCH=1E5A          /KIM/OUTCH=1EA0          BRK.VCT.IN=3FA6
/HARDCOPY=0000           HARDCOPY/A=4168          HC/RTS=4173
END/A=4174               DO/LINFEED=4168          HARDCOPY/B=416A
 ND/B=4174               /C/PORT=1702             IN/PORT=415B

//0000,4163,4163
```

MACRO ASSM/TED - Unconfigured Version and Versions for KIM

The information on these sheets describe how to load the data on the supplied cassette and configure this software for your system.

The supplied cassette is in a specially recorded format which can be read by any 6502 based system with a system clock of 1 mhz. The procedure for loading the data on this cassette is as follows:

Procedure for loading the data on the cassette tape

1) Read the description of the Fast Cassette Interface. This is the software which reads the cassette data.

2) Manually enter the object code contained on the Fast Cassette Interface listing, and construct the connection to your tape deck.

3) Configure this listing per your system. The required changes pertain to cassette input/output ports and are underlined in the listing.

4) Enter the following data:

Address	Data
0123	01
0124	00
0125	20
0126	FD
0127	3F

5) Insert cassette tape and position a few seconds before start of data, execute the Fast Cassette Interface software at 4141, and then press the Play switch on the tape deck.

6) After approximately one minute, the data should have been loaded. If the contents of the accumulator = 00 then you have a good load. If = EE then an error was detected, and you should try again. If it appears the program "hung up", then recheck connections and the modifications to the Fast Cassette Interface software. Also try via the invertor in the circuit.

The following is not required for the following versions: KIM

Configure ASSM/TED for your system requirements

1) Configure via table A by entering the address of appropriate routines or patches.

2) If you prefer, link ASSM/TED to the Fast Cassette Interface software as follows:

Address	Data	
3FA3	4C A5 40	links in load
3FD3	4C 00 40	links in record

C. SYM

The default file boundaries for SYM are: text file = 0200-0BFC, label file =0C00-0EFC, and relocatable buffer = 0F00. When entering the file boundary via the >SET command, enter the end address minus 3 (example: If the end = 0BFF, then enter 0BFC).

ASSM/TED provides software for controlling two tape motors. ASSM/TED assumes the record deck (deck 0) is connected to the on board motor control. If the user implements motor control hardware for the play deck (deck 1), ASSM/TED can control it via pin A-15 ("1" = on, "0" = off).

ASSM/TED for the SYM uses BB-F8 of zero page and most of the bottom of the stack (0100 up).

Listing 2 (SYM-1)

```
>ASSEMBLE LIST

0010 ;***RELOCATING LOADER FOR THE SYM-1 ASSM/TED***
0020 ;
0030 ;
0040 ;
0050            .OS
0060 ;
0070 ;*****COPYRIGHT 1979 BY CARL MOSER.*****
0080 ;*****    ALL RIGHTS RESERVED.     *****
0090 ;
0100 ;
0110 ;
0120 ;
0130 ;+++++++  USER INPUTTED VARIABLES BEFORE EXECUTION +++++++
0140 FILE/NO    .DE $0110     ;FILE NUMBER (0-99)
0150 OFFSET     .DE $E0       ;RELOCATOR OFFSET (2 BYTES)
0160 BUFFER     .DE $C8       ;ADDRS. OF R.L. BUFFER
0170 ;
0180 ;
0190 ;
0200 ;             RELOCATOR DIRECTIVES
0210 ;
0220 ;  DIRECTIVE                 DESCRIPTION
0230 ;  ---------                 -----------
0240 ;    0F            EXTERNAL 2 BYTE ADDRS. PRECEEDS,
0250 ;                  DON'T RELOCATE.  OTHERWISE RELOCATE.
0260 ;
0270 ;    1F            #L,  DATA PRECEEDS.
0280 ;
0290 ;    2F            #H,  DATA PRECEEDS, LO PART FOLLOWS.
0300 ;
0310 ;    3F            .AS OR .HS BYTE FOLLOWS.
0320 ;
0330 ;    4F            .SE OR .SI 2 BYTE ADDRS. FOLLOWS.
0340 ;
0350 ;    5F            TURN RELOCATOR ON (VIA .RS).
0360 ;                  (RESOLVE ADDRESSES AND RELOCATE
0370 ;                   CODE.)
0380 ;
0390 ;    6F            TURN RELOCATOR OFF (VIA .RC).
0400 ;                  (RESOLVE ADDRESSES BUT DO NOT
0410 ;                   RELOCATE CODE.)
0420 ;
0430 ;    7F            .DS - 2 BYTE BLOCK VALUE FOLLOWS.
0440 ;
0450 ;
0460            .BA $0200
0470 ;
0480 ;TAPE INPUT PARMS
0490 LOAD/NO    .DE $0180 0: NO STORE; 1: STORE
0500 TSTART     .DE $A64C LOAD BEGINNING AT TSTART
0510 TEND       .DE $A64A STOP LOADING AT TEND
0520 ;
0530 ;
0540 ;HEADER INPUT DATA
0550 HFILE/NO   .DE $017A HEADER FILE NUMBER
```

```
                        0560 HSTART      .DE $017B HEADER START
                        0570 HEND        .DE $017D HEADER END
                        0580 ;
                        0590 ;
                        0600 ;VARIABLES
                        0610 SCRAT       .DE $11E SCRATCH AREA
                        0620 TEMP1       .DE $11F SCRATCH AREA
                        0630 TEMP2       .DE $120 SCRATCH AREA
                        0640 SAVE        .DE $121 SCRATCH AREA
                        0650 ADDRS       .DE $DC 4 BYTES OF ADDRESS INFO.
                        0660 BUFF.END    .DE $0123 END OF 256 BYTE BUFFER
                        0670 BUFF.INDEX  .DE $0124 PRESENT ACCESSED DATA FROM BUFFER
                        0680 ;
                        0690 ;
                        0700 ;R(X)=00:  RELOCA+COR ON
                        0710 ;R(X)=02:  RELOCATOR OFF
                        0720 ;
                        0730 ;BEGIN EXECUTION AT LABEL START
                        0740 ;
0800- A2 FF             0750 START       LDX #$FF
0802- 9A                0760             TXS INITIALIZE STACK
0803- E8                0770             INX R(X)=00: SET RELOCATOR INITIALLY TO ON
0804- D8                0780             CLD
0805- 8E 21 01          0790             STX SAVE R(X)=00
0808- 20 E3 08          0800             JSR LOAD+BUFF
080B- 4C 11 08          0810             JMP ENTY
080E- 20 71 09          0820 LOOP1       JSR GET+DATA
                        0830 ;
0811- C9 7F             0840 ENTY        CMP #$7F      ;CK6. FOR .DS
0813- D0 03             0850             BNE PRO.3F
0815- 4C A7 09          0860             JMP PRO.7F    ;JUMP TO PROCESS DIR. 7F
0818- C9 3F             0870 PRO.3F      CMP #$3F CK6. FOR RELOCATOR DIRECTIVE
081A- D0 0B             0880             BNE OP+CK6
081C- 20 71 09          0890             JSR GET+DATA
081F- 81 DC             0900             STA (ADDRS,X)
0821- 20 85 09          0910             JSR INC+ADDRS
0824- 4C 0E 08          0920             JMP LOOP1
0827- C9 4F             0930 OP+CK6      CMP #$4F CK6. FOR .SE, .SI
0829- D0 03             0940             BNE W:
082B- 4C AA 08          0950             JMP TWO+BYT+AD
082E- C9 5F             0960 W:          CMP #$5F CK6. FOR RELOCATOR ON
0830- D0 04             0970             BNE CKNX
0832- A2 00             0980             LDX #$00
0834- F0 D8             0990             BEQ LOOP1
                        1000 ;
0836- C9 6F             1010 CKNX        CMP #$6F CK6. FOR RELOCATOR OFF
0838- D0 04             1020             BNE NO+REL
083A- A2 02             1030             LDX #$02
083C- D0 D0             1040             BNE LOOP1
083E- 81 DC             1050 NO+REL      STA (ADDRS,X) STORE OP CODE
0840- 20 85 09          1060             JSR INC+ADDRS
0843- C9 00             1070             CMP #$00 CK6. FOR BRK INSTR.
0845- F0 C7             1080             BEQ LOOP1
0847- C9 20             1090             CMP #$20 CK6. FOR JSR INSTR.
0849- F0 5F             1100             BEQ TWO+BYT+AD
084B- 8D 21 01          1110             STA SAVE SAVE R(A), IT CONTAINS OP CODE
084E- 29 9F             1120             AND #$9F
0850- F0 BC             1130             BEQ LOOP1
```

```
0253- F0 BC     1140            BEQ LOOP1
0255- AD 21 01  1150            LDA SAVE RESTORE OP CODE
0258- 29 1D     1160            AND #$1D
025A- C9 08     1170            CMP #$08 ↑↑KG. FOR ONE BYTE INSTR.
025C- F0 B3     1180            BEQ LOOP1
025E- C9 18     1190            CMP #$18 CKG. FOR ONE BYTE INSTR.
0260- F0 AF     1200            BEQ LOOP1
                1210 ;
                1220 ;NOW, TEST FOR INSTR. CONTAINING 2 BYTES
                1230 ;OF ADDRESS INFORMATION
                1240
0262- AD 21 01  1250            LDA SAVE RESTORE OP CODE
0265- 29 1C     1260            AND #$1C
0267- C9 1C     1270            CMP #$1C
0269- F0 42     1280            BEQ TWO←BYT←AD
026B- C9 18     1290            CMP #$18
026D- F0 3E     1300            BEQ TWO←BYT←AD
026F- C9 0C     1310            CMP #$0C
0271- F0 3A     1320            BEQ TWO←BYT←AD
                1330 ;
                1340 ;THE REMAINING CONTAIN ONE BYTE OF
                1350 ;ADDRESS INFORMATION
                1360 ;
                1370 ;PROCSSING OF ON BYTE ADDRESSES AND IMMEDIATE DATA
0273- 20 74 03  1380 ONE←BYT←AD JSR GET←DATA
0276- 81 DC     1390            STA (ADDRS,X)
0278- 20 88 03  1400            JSR INC←ADDRS
027B- 20 74 03  1410            JSR GET←DATA
027E- C9 2F     1420            CMP #$2F CKG. FOR RELOCATOR DIRECTIVE
0280- F0 14     1430            BEQ IMM←HI CKG. FOR #H,
0282- C9 1F     1440            CMP #$1F CKG. FOR RELOCATOR DIRECTIVE
0284- D0 8E     1450            BNE ENTY
                1460 ;
                1470 ;PROCESS #L,  DATA FOR RELOCATION
0286- 20 95 03  1480 IMM←LO     JSR DEC←ADDRS
0289- 18        1490            CLC
028A- A1 DC     1500            LDA (ADDRS,X)
028C- 65 E0     1510            ADC ♦OFFSET+$00 ADD OFFSET LOW PART FOR #L,
028E- 81 DC     1520            STA (ADDRS,X)
0290- 20 88 03  1530            JSR INC←ADDRS
0293- 4C 11 02  1540 BACK←TO←L1 JMP LOOP1
                1550 ;PROCESS #H,  DATA FOR RELOCATION
0296- 20 74 03  1560 IMM←HI     JSR GET←DATA LOW BYTE FOLLOWS REL. DIR.
0299- 18        1570            CLC
029A- 65 E0     1580            ADC ♦OFFSET FORM THE LO ADDRS. PART
029C- 08        1590            PHP
029D- 20 95 03  1600            JSR DEC←ADDRS
02A0- 28        1610            PLP
02A1- A1 DC     1620            LDA (ADDRS,X)
02A3- 65 E1     1630            ADC ♦OFFSET+$1 NOW FORM THE EFFECTIVE #H,
02A5- 81 DC     1640            STA (ADDRS,X)
02A7- 20 88 03  1650            JSR INC←ADDRS
02AA- 4C 11 02  1660            JMP LOOP1
                1670 ;
                1680 ;PROCESSING OF TWO BYTE ADDRESSES
02AD- A0 02     1690 TWO←BYT←AD LDY #$02
02AF- 98        1700 XX         TYA
02B0- 48        1710            PHA SAVE R(Y)
```

```
02B1- 20 74 03  1720            JSR GET+DATA
02B4- 81 DC     1730            STA (ADDRS,X)
02B6- 20 88 03  1740            JSR INC+ADDRS
02B9- 68        1750            PLA
02BA- A8        1760            TAY RESTORE R(Y)
02BB- 88        1770            DEY
02BC- D0 F1     1780            BNE XX
02BE- 20 74 03  1790            JSR GET+DATA
02C1- C9 0F     1800            CMP #$0F CKG. FOR RELOCATOR DIRECTIVE
02C3- D0 03     1810            BNE XY
02C5- 4C 11 02  1820            JMP LOOP1
02C8- 48        1830 XY         PHA
02C9- 20 95 03  1840            JSR DEC+ADDRS
02CC- 20 95 03  1850            JSR DEC+ADDRS
                1860 ;DECREMENT BACK TO ADDRESS START
                1870 ;
02CF- A1 DC     1880            LDA (ADDRS,X)
02D1- 18        1890            CLC
02D2- 65 E0     1900            ADC +OFFSET ADD OFFSET LO
02D4- 81 DC     1910            STA (ADDRS,X)
02D6- 20 88 03  1920            JSR INC+ADDRS
02D9- A1 DC     1930            LDA (ADDRS,X)
02DB- 65 E1     1940            ADC +OFFSET+$1 ADD OFFSET HI
02DD- 81 DC     1950            STA (ADDRS,X)
02DF- 20 88 03  1960            JSR INC+ADDRS
02E2- 68        1970            PLA
02E3- 4C 14 02  1980            JMP ENTY
                1990 ;
                2000 ;SUBROUTINE LOAD BUFFER WITH DATA FROM TAPE
                2010 ;
02E6- A9 7A     2020 LOAD+BUFF  LDA #$7A ADDLO OF START OF HEADER
02E8- 8D 4C A6  2030            STA TSTART+$00
02EB- A9 7F     2040            LDA #$7F ADDLO OF END OF HEADER
02ED- 8D 4A A6  2050            STA TEND+$00
02F0- A9 01     2060            LDA #$01 HI ADDRS
02F2- 8D 4D A6  2070            STA TSTART+$01
02F5- 8D 4B A6  2080            STA TEND+$01
02F8- 8D 80 01  2090            STA LOAD/NO 01: INDICATE TO LOAD
02FB- 20 D5 03  2100            JSR USER/LOAD USER LDA+BD FROM TAPE ROUTINE
                2110 ;
                2120 ;THE ABOVE SETS UP AND LOADS HEADER INFORMATION
                2130 ;FROM TAPE.  THE HEADER CONTAINS THE MODULE FILE
                2140 ;NUMBER, AND STARTING AND ENDING ADDRESS OF FOLLOWING
                2150 ;DATA.
                2160 ;
                2170 ;
02FE- D0 4D     2180            BNE ERROR IF Z-BIT FALSE, THEN ERROR IN LOADING
0300- A2 00     2190            LDX #$00
                2200 ;
0302- AD 7D 01  2210            LDA HEND+$00
0305- 38        2220            SEC
0306- ED 7B 01  2230            SBC HSTART+$00
                2240 ;CALCULATE NUMBER OF BYTES IN FOLLOWING DATA
                2250 ;
0309- 8D 23 01  2260            STA BUFF.END INITIALIZE BUFFER END POINTER
030C- AD 7E 01  2270            LDA HEND+$01
030F- ED 7C 01  2280            SBC HSTART+$01
0312- D0 39     2290            BNE ERROR ONLY 256 BYTE BUFFER ALLOWED
```

```
0314- A5 C8     2300            LDA *BUFFER
0316- 8D 4C A6  2310            STA TSTART
0319- 18        2320            CLC
031A- 6D 23 01  2330            ADC BUFF.END # BYTES
031D- 8D 4A A6  2340            STA TEND
0320- A5 C9     2350            LDA *BUFFER+$01
0322- 8D 4D A6  2360            STA TSTART+$01
0325- 69 00     2370            ADC #$00
0327- 8D 4B A6  2380            STA TEND+$01
                2390 ;NOW THE START AND END ADDRESS PARMS HAVE BEEN
                2400 ;SET UP TO LOAD FROM TAPE INTO THE BUFFER.
                2410 ;
032A- AD 10 01  2420            LDA FILE/NO USER ENTERED FILE NUMBER
032D- F0 08     2430            BEQ STORE.DATA IF F# = 00, LOAD ANYWAY
032F- CD 7A 01  2440            CMP HFILE/NO CMP WITH USER VERSUS THAT ON TAPE
0332- F0 03     2450            BEQ STORE.DATA
0334- 8E 80 01  2460            STX LOAD/NO R(X)=0; NO STORE
0337- 20 D5 03  2470 STORE.DATA JSR USER/LOAD
                2480 ;
                2490 ;THE ABOVE LOADS IN DATA INTO BUFFER DEPENDING
                2500 ;ON THE STATE OF LOAD/NO
                2510 ;
033A- D0 11     2520            BNE ERROR Z-BIT = FALSE THEN ERROR
033C- A2 00     2530            LDX #$00
033E- AD 7A 01  2540            LDA HFILE/NO
0341- C9 EE     2550            CMP #$EE COMPARE IF END OF FILE
0343- D0 0C     2560            BNE BUFFLOADED
0345- A9 00     2570            LDA #$00 INDICATE GOOD LOAD
0347- 00        2580 B          BRK
0348- EA        2590            NOP
0349- EA        2600            NOP
034A- 4C 00 02  2610            JMP START
034D- A9 EE     2620 ERROR      LDA #$EE INDICATE ERROR IN LOAD
034F- D0 F6     2630            BNE B
                2640 ;
                2650 ;
                2660 ;NOW GET ADDRS. INFO. AND PUT IN ADDRS+$2, +$3
                2670 ;ADDRS INFO. IS IN FIRST TWO BYTES OF BUFFER
                2680 ;
0351- AD 80 01  2690 BUFFLOADED LDA LOAD/NO CKG. IF PROPER DATA
0354- F0 90     2700            BEQ LOAD+BUFF
0356- AE 21 01  2710            LDX SAVE RESTORE R(X)
0359- A0 00     2720            LDY #$00
035B- B1 C8     2730            LDA (BUFFER),Y
035D- 85 DE     2740            STA *ADDRS+$2
035F- C8        2750            INY
0360- B1 C8     2760            LDA (BUFFER),Y
0362- 85 DF     2770            STA *ADDRS+$3
0364- 8C 24 01  2780            STY BUFF.INDEX SET BUFFER DATA POINTER
                2790 ;
                2800 ;SET RELOCATION ADDRS. IN ADDRS+$0, +$1
0367- A5 DE     2810            LDA *ADDRS+$2
0369- 18        2820            CLC
036A- 65 E0     2830            ADC *OFFSET
036C- 85 DC     2840            STA *ADDRS
036E- A5 E1     2850            LDA *OFFSET+$1
0370- 65 DF     2860            ADC *ADDRS+$3
0372- 85 DD     2870            STA *ADDRS+$1
```

```
03CA- 85 DE     3460            STA *ADDRS+2
03CC- 98        3470            TYA  ;GET HI
03CD- 65 DF     3480            ADC *ADDRS+3
03CF- 85 DF     3490            STA *ADDRS+3
03D1- 68        3500 NO.PROC    PLA
03D2- 4C 11 02  3510            JMP LOOP1
                3520 ;
                3530 ;
                3540 ;
                3550 ;      ***SYM CASSETTE INTERFACE PATCH ***
                3560 ;
                3570 ;
                3580 ;SYM DEFINITIONS:
                3590 SAVER      .DE $8188
                3600 ACCESS     .DE $8B86
                3610 ID         .DE $A64E
                3620 MODE       .DE $FD
                3630 CONFIG     .DE $89A5
                3640 ZERCK      .DE $832E
                3650 P2SCR      .DE $829C
                3660 LOADT      .DE $8C78
                3670 NACCESS    .DE $8B9C
                3680 RESXAF     .DE $81B8
                3690 ;
                3700 ;
03D5- 20 88 81  3710 USER/LOAD  JSR SAVER     ;SAVE REGISTERS
03D8- A9 FF     3720            LDA #$FF      ;ID=FF FOR USER RANGE
03DA- 8D 4E A6  3730            STA ID
03DD- A0 80     3740            LDY #$80
03DF- 84 FD     3750            STY *MODE     ;BIT 7=1 FOR H.S.
03E1- A9 09     3760            LDA #$09
03E3- 20 A5 89  3770            JSR CONFIG
03E6- 20 2E 83  3780            JSR ZERCK
03E9- 20 9C 82  3790            JSR P2SCR
03EC- 20 7B 8C  3800            JSR LOADT+$3          ;ENTRY IN TAPE LOAD
03EF- D8        3810            CLD
03F0- A9 00     3820            LDA #$00      ;Z-BIT =T
03F2- 90 02     3830            BCC SKPERRU/L
03F4- A9 01     3840            LDA #$01      ;Z-BIT =F
                3850 SKPERRU/L
03F6- 4C B8 81  3860            JMP RESXAF    ;RESTORE REGS. EXCEPT A,PSR
                3870 ;
                3880 ;
                3890 END.PGM    .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/FILE/NO=0110           /OFFSET=00E0            /BUFFER=00C8
/LOAD/NO=0180           /TSTART=A64C            /TEND=A64A
/HFILE/NO=017A          /HSTART=017B            /HEND=017D
/SCRAT=011E             /TEMP1=011F             /TEMP2=0120
/SAVE=01c1              /ADDRS=00DC             /BUFF.END=0123
/BUFF.INDEX=0124        START=0200              LOOP1=0211
ENTY=0214               PRO.3F=021B             OP+CKG=022A
W:=0231                 CKNX=0239               NO+REL=0241
ONE+BYT+AD=0273         IMM+LO=0286             BACK+TO+L1=0293
```

```
IMM←HI=0296          TWO←BYT←AD=02AD      XX=02AF
XY=02C8              LOAD←BUFF=02E6       STORE.DATA=0337
B=0347               ERROR=034D           BUFFLOADED=0351
GET←DATA=0374        WX=0385              INC←ADDRS=0388
SKIP←INC1=038E       SKIP←INC2=0394       DEC←ADDRS=0395
SKIP←DEC1=039F       SKIP←DEC2=03A9       PRO.7F=03AA
PROC.DS=03B9         NO.PROC=03D1         /SAVER=8188
/ACCESS=8B86         /ID=A64E             /MODE=00FD
/CONFIG=89A5         /ZERCK=832E          /P2SCR=829C
/LOADT=8C78          /NACCESS=8B9C        /RESXAF=81B8
USER/LOAD=03D5       SKPERRU/L=03F6       END.PGM=03F9

//0000,03F9,03F9
>
```