THE COMMENT FIELD

The last field of a source statement is the comment field. It may be of any length provided that the I/O buffer does not overflow. The comment is separated from the argument by a single space. If the line is a comment only, it must begin in column four.

In general, comments may include any printable or non-printing character with the exception of the end of file character. Comments may not begin with the symbol modification characters +,-, or /.

ASSEMBLER OPERATING INSTRUCTIONS

Once you hace prepared a source program in the prescribed format shown above, you may execute the assembler to check for errors and prepare the object code for execution.

Enter the assembler from the editor command mode by typing X or XEQ. Micro-ADE will respond "PASS 1", and request an input file ID.

```
-X
PASS 1
ID=
```

If the source has been saved on cassette, and is not resident in memory, enter the ID of the cassette file. If several blocks are saved sequentially on cassette with sequential identification, they can be read as a group by entering the first ID, a comma, and the last ID. Micro-ADE will then read each block, assemble it, increment ID, read the next block, and so on until the last block of records has been assembled. If the source is resident in memory, enter the ID= 00. This will cause the assembler to skip the cassette read step and proceed directly to the first pass of the assembly.

Resident Source	Single Cassette File	Four Files with ID A1,A2,A3,A4
-X	-X	-X
PASS 1	PASS 1	PASS 1
ID= 00	ID= A1	ID= A1,A4

Note that since ID= 00 is used to indicate a resident file, a source file should never be saved with this ID.

PASS 1

As each block is assembled through pass one, errors detected by the assembler will be flagged, and the offending source line printed. When the assembler has completed the block, it will again prompt for an ID. If there are more blocks of source to be read, enter the ID of the next block. If this was the last file, respond with a RETURN to signify the end of the source program. The symbol table has now been compiled. Micro-ADE will proceed to pass two.

PASS 2

Immediately, the assembler will prompt "PRINT?". If you wish to have a listing of the program printed at your terminal, respond with a Y or YES. If not, respond with N or a RETURN.

The assembler will now ask for a "SAVE ID=". If you wish the object code generated to be saved on cassette enter a valid ID (01 to FF). After the code has been assembled, the object will be automatically written to the output cassette with the appropriate adresses for a direct load for execution. If you do not wish to save the object code at this time, respond with a carriage return.

If there are multiple input files, the ID of the output object block will be incremented each time a new input file is read. The resulting group of object blocks may then be loaded using the GET x,y command in the editor.

The assembler is now ready to execute pass two. It will prompt for the input ID once again. This should now be entered exactly as for pass one. Remember to rewind the input cassette first.

Examples continued from above.

```
ID=(r)              ID=(r)              ID=(r)
PASS 2              PASS 2              PASS 2
PRINT?YES(r)        PRINT?(r)           PRINT?NO(r)
SAVE ID=(r)         SAVE ID=23(r)       SAVE ID=A1(r)
ID= 00(r)           ID= 00(r)           ID= A1,A4(r)
```

(A listing will be printed)

Error flags will be printed with the offending source statement regardless of the response given to the PRINT query.

At the end of the assembly you will be returned to the editor command mode. If any errors were flagged, they should be corrected in the source file, and the program reassembled before attempting execution.

If no errors were detected during both passes of the assembler, rewind the output cassette, and place it on the input cassette player. Then, load the object code from cassette using the GET command. If the source was in a single block, you may move the object code to its execution address using the BLOCKMOVE command.

OBJECT FORMAT

The object code generated by the assembler is stored in an area of memory allocated to it. This allows you to write programs which are larger than the available memory when the source, and even the assembler are in the system. Each time a new source block is read, the object code pointers are reset and the new object code is written over the old object. For this reason, the object code must be saved in short blocks corresponding to each cassette load. This operation is carried out automatically by the assembler if you are using automatic cassette control.

The object saved on cassette is ready to be loaded using either the KIM cassette load program, or the Micro-ADE GET command.

If only a single source file was used, the entire object program will be resident in the object memory area. If it was ORGed for execution at that address, you may execute the program immediately. Otherwise, you can use the BLOCKMOVE command to move it to its execution address. This is also a convenient way to write short patches to existing programs using the assembler.

THE SYMBOL TABLE

The symbols defined by the assembler, and their two byte hexadecimal equivalents are stored in a reserved area of memory called the symbol table. The symbol table is also used by the disassembler to label addresses and symbolically define arguments.

The symbols are saved in a packed ASCII format which aloows three characters to be packed into two bytes. This is accomplished by stripping each character of the three most significant bits leaving only the five low order bits which

define the character itself. It is because of this packing operation that only the characters A through Z are allowed in symbols. Each six character symbol requires four bytes for the symbol, plus the two following bytes for the hexadecimal equivalent value. Using this scheme, more than 170 symbols can be packed into 1K of symbol area.

The symbol table may be listed at the terminal in either alphabetical or address order. The table in alphabetical order can be used to avoid duplication when defining new symbols, or as a reference when defining symbols external to another program. The symbol table in address order is useful when defining overlays or looking for unused areas of page zero for expansion of a program.

T The TABLE Command

The command T, or TØ will cause Micro-ADE to print the symbol table in alphabetical order. The starting and ending addresses of the table are also given for your information.

TABLE 1

The command T1 will cause the printing of the symbol table in address order.

TABLE 2

The command T2 is used to determine the starting and ending addresses of the symbol table. This is useful for determining how close the table is to overflowing, or for determining the exact table location for saving it on cassette.

TABLE 3

If you have saved the symbol table on cassette at the time of assembling a program, it is easy to reload it again if you wish to use the disassembler. Once the table has been loaded using the GET command, it is necessary to set the end of symbol table parameter so that the disassembler will search the table correctly. This may be accomplished with the T3, a command, where a is the new address of the end of the table. The previous address of the end of the table will be printed.

ASSEMBLER ENTRY ADDRESSES

It is possible to execute the assembler from addresses other than the normal start address in order to recover from a user error, or to use the assembler in a non-standard way. These are described below.

BAD CASSETTE READ

If a cassette will not read properly, return to the editor using the NMI (ST-key on KIM). Very often, there will be a single bad byte which has caused a checksum error. This may be corrected using the editor. Once done, you may save the clean copy, and resume the assembly from the point where you left off, by executing IDAS ($2608 in version 1.0). The assembler will prompt for an ID. Since the source is now resident, respond with 00, and continue the assembly as usual. This method may be used in pass one or pass two.

ADDITION TO SYMBOL TABLE

If you wish to add to an earlier symbol table, rather than creat a new one, you may execute OLDST ($2601) without resetting the symbol table parameter. The assembler will operate normally. This method is useful for assembling small patches or new programs which reference a large earlier program without having to define a large number of external symbols.

CONTINUE ENTRY

If an error occured during an assembly which caused a break in execution, you may wish to continue from the point where you left off in order ro check the source for syntaxs errors, etc. (The object code generated will not be executable). The assembler will continue from a BREAK with the next source statement if ERRTRY ($266C) is executed.

PASS 2 ONLY

If you have previously assembled a program, and the symbol table was saved, you may reassemble the second pass only in order to print a listing. Load the symbol table manually, remembering to reset the end of table address, and execute PASTWO ($26E6). This is only possible if no changes have been made to the source program.

PRINT ONE BLOCK ONLY

If you wish to list only one section of a long multi-file program, this can be accomplished as follows. When prompted for the ID, hit the BREAK key. Then, execute PASTWO ($26E6) and change your response to the "PRINT?" prompt. Respond to the ID prompt with the correct next file. This method may be used to set or reset the print flag.

T H E D I S A S S E M B L E R

A useful program for debugging or modifying programs when the source listing is not available is a disassembler. The disassembler reads object code and interprets it into 6502 assembler instructions where possible. The symbol table is searched for addresses and arguments in order that lines may be labelled and arguments interpreted symbolically.

Z The DISASSEMBLE Command

The Z command is used to execute the disassembler. There are three modes of use. Z a,b will cause a disassembly of the data from address a to address b without a pause. If you are using a CRT, it is more convenient to disassemble a fixed number of lines at a time. Z a will disassemble from address a, until 16 lines heve been displayed. The system will then pause for a keyboard input. The space bar will cause the program to continue. Typing RETURN will cause the program to return to the command mode. If you now type the command Z, without parameters, the disassembler will resume disassembly from where if left off.

Symbols

If the program you are disassembling is the last one assembled, the symbol table will already be initialized, and the disassembly listing will have all symbols interpreted. If not, you will have to create a new symbol table. If the symbol table was saved from the assembly of a program, you can reload it with the GET command. The table end address must then be defined using the T3 command.

If you wish to create a new symbol table, use the assembler to do so. Symbols may be defined using the define symbol (*) pseudo instruction. Only one pass of the assembler is required. Use the BREAK key to exit from the assembler at pass two.

If the disassembler runs slowly and interprets symbols with unusual names, the end of the symbol table has not been initialized properly. Type X to the command prompt, then BREAK to return. The assembler will initialize the table, and the disassembler will now operate correctly.

Relocation

A disassembler can make the relocation of most programs very easy. Three byte instructions stand out clearly from the code. Change the high byte for each of these instructions and you

have done most of the work. Then, look carefully through the code for indirect operations. Find out where the page zero addresses used have been defined and make the necessary changes. Further changes are usually not necessary, but if they are, it may be necessary to single step through some of the code to detect unusual programming tricks that the author has used.

Patches

If you wish to change a single subroutine, address, or a special character, an easy way to locate most references to it is to define it as a symbol with a highly visible name, such as XXXXXX. Then, disassemble the entire program. The occurences will be easily seen.

EXAMPLE OF A DISASSEMBLY

```
-Z 2DF0, 2E29

2DF0 84 F4        OUTCH  STYZ   YTMP
2DF2 86 F5               STXZ   XTMP
2DF4 C9 0D               CMPIM  $000D
2DF6 D0 1E               BNE    NOCR
2DF8 A6 63               LDXZ   PMODE
2DFA D0 13               BNE    NOPG
2DFC E6 64               INCZ   COUNTL
2DFE D0 0F               BNE    NOPG
2E00 20 2B 2E            JSR    INCH
2E03 C9 1B               CMPIM  $001B
2E05 D0 04               BNE    ON
2E07 A9 FF               LDAIM  $00FF
2E09 85 63               STAZ   PMODE
2E0B A2 F0        ON     LDXIM  GANG
2E0D 86 64               STXZ   COUNTL
2E0F A9 0D        NOPG   LDAIM  $000D
2E11 20 16 2E            JSR    NOCR
2E14 A9 0A               LDAIM  $000A
2E16 20 A0 2E     NOCR   JSR    OUTPUT
2E19 2C 40 17     BRKTST BIT    $1740
2E1C 10 05               BPL    BREAK
2E1E A6 F5               LDXZ   XTMP
2E20 A4 F4               LDYZ   YTMP
2E22 60                  RTS
2E23 2C 40 17     BREAK  BIT    $1740
2E26 10 FB               BPL    BREAK
2E28 4C 31 20            JMP    RESTRT
```

EXAMPLE PROGRAM

```
KIM
29FF 20 F2
00F2 04 FF.              set up the
00F3 0D 17FA             NMI vector
17FA 00 31.
17FB 1C 20.
17FC 00 2000
2000 D8 G                execute from $2000

NEW?Y
                         Respond Y to clear workspace
CLEAR
-ADD
                         ADD new data
0000:   SHORT MESSAGE PROGRAM                Input for a
0010: EXAMPL ORG $0200                       program
0020: INDCT * $0066
0030: OUTCH * $2DF0 PRINT CHAR
0040:  LDAIN MESSG                           line 0010 -delete
0050:  STA INDCT                             was used to backspace
0060:  LDAIM MESSG /256                      over a typing error
0070:  STS INDCT +01
0080:  LDYIM $00                             note spacing of label,
0090: LOOP LDAIY INDCT                       instruction, and
0100:  JSR OUTCH                             argument
0110:  INY
0120:  BNE LOOP
0130:  CPYIM 02
0140:  JMP $2031
0150: MESSG = 'H
0160:   = 'I
0170: @                                      @ end of data input

-X                       execute the assembler to check for syntax
                         errors

PASS 1
ID=00
                         ID= 00  --resident source
**********XE2X0040               LDAIN MESSG          Adress mode!
**********X07X0070               STS   INDCT +01      Instruction!
ID=
                         a carriage return indicated the last tape

PASS 2
PRINT? NO                not worth printing yet

SAVE ID=                 a carriage return indicated-no save

ID=00                    00 - resident source

**********(E2)0040
0040: 0200 00 00 00            LDAIN MESSG
**********(07)0070
0070: 0207 00 00 00            STS   INDCT +01
**********(A8)0130
0130: 0214 00 00 00            CPYIM 02             Argument!
ID=
                         a carriage return indicated the end
```

```
-FIX 40
0040:  LDAIN MESSG
0040:  LDAIN MESSG
0041: @

-F70
0070:  STS INDCT +01
0070:  STA INDCT +01
0071: @

-F 130
0130:  CPYIM 02
0130:  CPYIM @@2
0131: @

-X

PASS 1
ID=00

ID=

PASS 2
PRINT? Y

SAVE ID=

ID=00
```

Fix line 40.
Use ctl-E and 7 deletes, type M, ctl-E.
No need to insert more lines - @ to end fix

Fix line 70.
Type to STA, then use ctl-E to the end.

Fix line 130.

Execute assembler again.

Print a listing this time.

```
EXAMPL    MICRO-WARE ASSEMBLER 65XX-1.0 PAGE 01


0000:                    SHORT MESSAGE PROGRAM
0010: 0200               EXAMPL ORG    $0200
0020: 0200               INDCT  *      $0066
0030: 0200               OUTCH  *      $2DF0  PRINT CHAR
0040: 0200 A9 17                LDAIM MESSG
0050: 0202 85 66                STA   INDCT
0060: 0204 A9 02                LDAIM MESSG  /256
0070: 0206 85 67                STA   INDCT  +01
0080: 0208 A0 00                LDYIM $00
0090: 020A B1 66         LOOP   LDAIY INDCT
0100: 020C 20 F0 2D             JSR   OUTCH
0110: 020F C8                   INY
0120: 0210 D0 F8                BNE   LOOP
0130: 0212 C0 02                CPYIM $02
0140: 0214 4C 31 20             JMP   $2031
0150: 0217 48            MESSG  =      'H
0160: 0218 49                   =      'I
ID=
```

Listing looks OK.

Execute Program at $0200.

```
-X 200
HI HIt  g- g-%LI n-1 M-H@Ps g- %i%iEp M-% M-): p-L*L1  C'C&TP<) g-
-M 140,120

-N
```

Program failed.
Rearrange lines.
Number lines.

```
-L120,150
0120:   INY
0130:   CPYIM $02
0140:   BNE LOOP
0150:   JMP $2031

-X

PASS 1
ID=00

ID=

PASS 2
PRINT?

SAVE ID= B0

ID=00

ID=

-X 200
HI
-S E1
3600 3713
-Z 200

0200 A9 17        EXAMPL LDAIM $0017
0202 85 66               STAZ  INDCT
0204 A9 02               LDAIM $0002
0206 85 67               STAZ  $0067
0208 A0 00               LDYIM $0000
020A B1 66        LOOP   LDAIY INDCT
020C 20 F0 2D            JSR   OUTCH
020F C8                  INY
0210 C0 02               CPYIM $0002
0212 D0 F6               BNE   LOOP
0214 4C 31 20            JMP   $2031
0217 48           MESSG  PHA
0218 49 20               EORIM $0020
021A 48                  PHA
021B 49 F4               EORIM $00F4
021D A0 00               LDYIM $0000

-T

    SYMBOL TABLE 3000 301E
    EXAMPL 0200    INDCT  0066    LOOP   020A    MESSG  0217
    OUTCH  2DF0
-T1

    SYMBOL TABLE 3000 301E
    INDCT  0066    EXAMPL 0200    LOOP   020A    MESSG  0217
    OUTCH  2DF0
-
```

List corrected section of source.

Execute assembler again.

Save object with ID = B0

Execute program.
Success!

Save source as E1.

Disassemble from address 0200.

Even tables can look like program sometimes.

Print the symbol table.

SETTING UP THE MICRO-ADE SYSTEM

Once you have loaded Micro-ADE into your system, there are a number of parameters which may have to be initialized before you can use the program.

The JUMP Table

The following subroutines are external to Micro-ADE and must be defined for each system.

Address	Routine	KIM	TIM or JOLT	Other
2E94	PACKT	4C 00 1A	4C A9 2E	4C A9 2E
2E97	READ	4C AC 2E	JMP to your own cassette read	
2E9A	WRITE	4C 32 2F	JMP to your own cassette write	
2E9D	INPUT	4C 5A 1E	4C E9 72	ASCII input
2EA0	OUTPUT	4C A0 1E	4C C6 72	ASCII output

PACKT

PACKT is a KIM subroutine which is used to pack two ASCII characters into a hexadecimal byte. It is called twice, with the ASCII input in the accumulator each time. After the second call, the hex byte is returned in the accumulator and in location SAVX. If the ASCII character is a valid hexadecimal value, the Z-flag is set before returning. If not, the Z-flag is reset. The X register must be preserved. Many systems will already have such a routine in their operating system which may be used. If not, the routine below can be used. Since the CREAD and CWRITE routines cannot be used by systems other than KIMs, this area of memory is available for patches and expansion. Alternations must be made to the editor, because SAVX is accessed directly by some operations.

READ

This is a subroutine which is used to input the source and data files from cassette tape. The routine will read a file with a hexadecimal identification passed in ID ($0062). The address to which the data is written is part of the file itself. When a succesful read is completed, the subroutine returns. No registers need be saved.

WRITE

This is a subroutine which is used to output source or object files to cassette tape. The program saves a file with identification ID ($0062) as it exists in memory from address SAL, SAH ($17F5, $17F8) and writes the startaddress SALX, SAHX ($0061, $0062) onto the tape for disposition when loading.

The CREAD and CWRITE routines also turn on the cassette recorders using the PIA on the KIM. If these routines are not used, the initialization of the cassette control at address $2043 should be replaced with 8 NOPs.

READ and WRITE may be replaced with calls to any mass storage device capable of storing the data and reloading it in the required format. Paper tape, floppy disk or other media may easily be used. A disk oriented version of Micro-ADE is currently being developed.

INPUT

The INPUT subroutine polls a keyboard device and return with ASCII data in the accumulator. Mark, space, even, or odd parity may be used. No registers need be saved. A line feed is sent to the output routine each time a carriage return is entered. Otherwise, all echoing is assumed to be external to the Micro-ADE system.

OUTPUT

The OUTPUT subroutine prints the ASCII character passed in the accumulator on a display device. The data is passed with bit 7 equal to zero. No padding is provided for carriage returns. A line feed is automatically sent with each carriage return.

TERMINAL DEVICES

It seems that every terminal available today has one or two non-standard features. In order to allow each user to adapt the Micro-ADE package to his own hardware, we have provided the source listing for all of the key I/O functions. The comments will allow you to change the backspace character, remove printing control character, or unnecessary rub-outs of nonprinting control characters, change the delete function, use your own BREAK test, or completely modify the line input buffer to suit your own taste.

End of File Character

If you wish to change the end of file character from to something else, such as ctl-D, the locations to change are: $201F, $20E3, $2134, $215D, $23B0, $247C, $249D, and $24F0.

Page length

The assembler currently prints a form feed character ($0C) to start a new page. This character is located at address $29FE. It may be replaced with a return ($0D) or a null ($00).

The number of lines per assmbler page is specified as 58 by the $C8 at address $2A36. This byte may be changed to suit your printer.

The number of lines per disassembly for a CRT is specified as 16 by the $F0 at address $2308.

The number of lines per page in PAGE MODE is specified as 16 by the $F0 at $2E08.

Page 17 References

Since version 1.0 of Micro-ADE is set up to use KIM monitor routines, it was necessary to pass some parameters in page 17 locations. The cross reference table below will enable you to replace all of these addresses with the equivalent for your system.

SYMBOL	ADDRESS	REFERENCES	FUNCTION
SAVX	17E9	206F 2091 2096	used by PACKT
SAL	17F5	21C6 21EE 26C3	used by CWRITE
SAH	17F6	21D0 21F3 26C9	used by CWRITE
EAL	17F7	21CB 26D0	used by CWRITE
EAH	17F8	21D5 26D7	used by CWRITE
IRQ	17FE	203B 2678	change to your IRQ
	17FF	2040 267D	or FFFE, FFFF
PIA	1702	2045	cassette control
	1703	2048	PIA port

The PACKT Subroutine

```
0010: 2EA9            PACKT  ORG   $2EA9  77.06.29
0020:
0030: 2EA9            SAVX   *     $0065  TEMPORARY DATA
0031:
0040: 2EA9 C9 47             CMPIM $47    TOO HIGH?
0050: 2EAB B0 1B             BCS   RET
0060: 2EAD C9 30             CMPIM $30    TOO LOW?
0070: 2EAF 90 17             BCC   RET
0080: 2EB1 C9 40             CMPIM $40    LETTER?
0090: 2EB3 90 02             BCC   N
0100: 2EB5 69 08             ADCIM $08    MAKE IT HIGHER!
0110: 2EB7 29 0F      N      ANDIM $0F    REMOVE GARBAGE
0120: 2EB9 A8                TAY          HIDE HEX DIGIT
0130: 2EBA A5 65             LDA   SAVX   GET FIRST HALF
0140: 2EBC 0A                ASLA         SHIFT
0150: 2EBD 0A                ASLA         IT
0160: 2EBE 0A                ASLA         OVER
0170: 2EBF 0A                ASLA         TO LEFT
0180: 2EC0 84 65             STY   SAVX   SAVE IT
0190: 2EC2 05 65             ORA   SAVX   PUT THEM TOGETHER
0200: 2EC4 85 65             STA   SAVX   SAVE WHOLE BYTE
0210: 2EC6 A0 00             LDYIM $00    CLEAR Z
0220: 2EC8 60         RET    RTS          RETURN
0370:
0380:                 PATCHES TO EDITOR
0390: 206F                   ORG   $206F
0400: 206F 85 65             STA   SAVX
0410: 2071 EA                NOP
0420:
0430: 2091                   ORG   $2091
0440: 2091 05 65             ORA   SAVX
0450: 2093 EA                NOP
0460: 2094 85 18             STA   LO
0470: 2096 84 65             STY   SAVX
0480: 2098 EA                NOP
```

MEMORY ALLOCATION

The Micro-ADE system (version 1.0) uses the following areas of memory:

Page 0	0010 to 0064	data
	00F0 to 00FF	temporary data
Page 1	0100 to 0140	input buffer
	01E0 to 01FF	stack
Page 17	17E9 to 17FF	see above
Page 20-2F	2000 to 2FFF	Micro-ADE program

The program from $2000 to $2FFF is pure code. Once initialized, it may be executed in protected memory, or placed in ROM. The program will not change any data in this area during execution.

MEMORY ALLOCATION TABLE

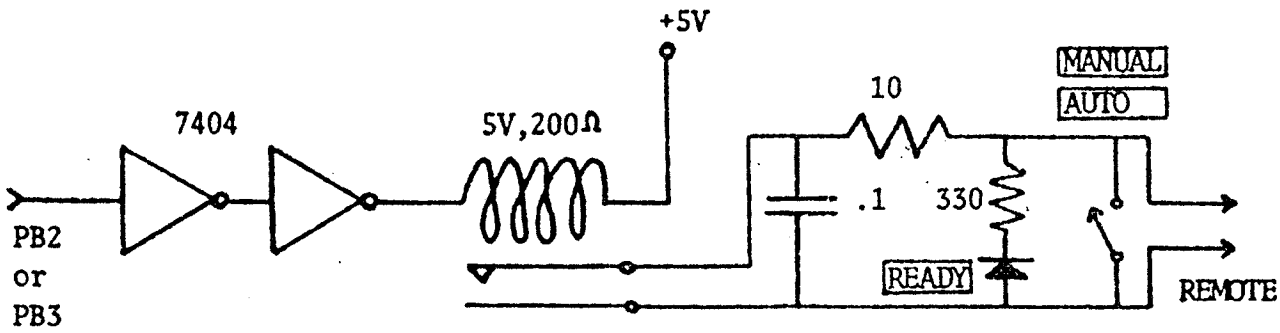
The areas of memory to be used for the various files associated with the Micro-ADE system are allocated by a table at address $2EA3. In this case, $3600 to $3FFF has been allocated as the source, $3000 to $35FF has been allocated as the symbol table, and $0200 upward has been allocated for object code.

Address	Definition	Allocation
2EA3	SOURCM = $35	SOURCE -1
2EA4	SOURCE = $36	First page of source code.
2EA5	SOURCF = $40	Last page of source +1.
2EA6	SYMBOL = $30	First page of symbol table.
2EA7	SYMF = $36	Last page of symbol table +1.
2EA8	OBJECT = $02	First page of object code.

The amount of memory allocated to each file will depend upon the memory available in your system as well as your personal programming style. The allocation shown above has proven to be ideal for writing programs of up to 400 bytes without the use of cassettes, and of up to 3K without overflowing the symbol table. The object allocation should always be approximately one fifth the size of the source area to prevent the possibility of overflow.

CASSETTE CONTROL

Micro-ADE is designed to be used with two computer controlled cassette recorders. Cassette 1 is used for input to the system, and cassette 2 is used for output from the system. These cassettes are turned on and off by the computer using the REMOTE input jack available on most recorders. The schematic for a simple interface between the KIM-1 PIA port and the cassette recorders is shown below.



Cassette Control Interface
(1 for each recorder)

PB2 controls Cassette 1

PB3 controls Cassette 2

Parts List

1 7404 IC
2 5V,200Ω spst reed relays
2 10Ω,1W resistors
* 2 330Ω,.5W resistors
2 .1uF,100V capacitors
* 2 spst switches
* 2 LEDs
* optional

ASSEMBLING WITH MANUAL CASSETTE CONTROL

Although the assembler is designed to operate most efficiently using two computer controlled cassette recorders, it is possible to use the system with as little as as one manually operated recorder.

The patches shown below will cause the system to print "R" when it is ready to read from a cassette, and "W", when it is ready to write to a cassette. It will then wait for a RETURN to indicate that the cassette recorder has been started. When the read or write operation is complete, Micro-ADE will type "X", and pause once again to allow you turn the recorder off. In addition to the patches below, the editor program should have the following code replaced with 8 NOP instructions: Address 2043: A9 0C 8D 03 17 8D 02 17.

```
2EAF A9 52      CREAD  LDAIM 'R
2EB1 20 F0 2D          JSR   OUTCH
2EB4 20 2B 2E          JSR   INCH


2F2A                   ORG   $2F2A


2F2A A9 58      OKRD   LDAIM 'X
2F2C 20 F0 2D          JSR   OUTCH
2F2F 20 2B 2E          JSR   INCH


2F35                   ORG   $2F35


2F35 A9 57      CWRITE LDAIM 'W
2F37 20 F0 2D          JSR   OUTCH
2F3A 20 2B 2E          JSR   INCH


2F98                   ORG   $2F98

2F98 20 8C 1E          JSR   INIT
2F9B A9 58             LDAIM 'X
2F9D 20 F0 2D          JSR   OUTCH
2FA0 4C 2B 2E          JMP   INCH
```

```
0010:
0020:
0030:
0040:
0050:
0060:                ************************************
0070:                ***** INPUT AND OUTPUT ROUTINES *****
0080:                ***** FOR THE  MICRO-ADE SYSTEM *****
0090:                ************************************
0100:
0110:
0120: 2DC5           IO      ORG   $2DC5  77.06.24
0130:
0140: 2DC5           BLO     *     $0010  POINTER TO WORKSPACE
0150: 2DC5           N       *     $0015  LINE NUMBER
0160: 2DC5           SALX    *     $0060  FILE EXECUTION ADDRESS
0170: 2DC5           SAHX    *     $0061
0180: 2DC5           ID      *     $0062  FILE ID
  90: 2DC5           PMODE   *     $0063  PAGE MODE FLAG
0200: 2DC5           COUNTL  *     $0064  LINE COUNT
0210:
0220: 2DC5           GANG    *     $00F0  CWRITE PULSER
0230: 2DC5           TIC     *     $00F1  CWRITE TIMER
0240: 2DC5           COUNT   *     $00F2  CWRITE COUNTER
0250: 2DC5           TMP     *     $00F3  TEMPORARY STORAGE
0260: 2DC5           YTMP    *     $00F4     "         "
0270: 2DC5           XTMP    *     $00F5     "         "
0280: 2DC5           TRIB    *     $00FE  CYCLE COUNTER
0290:
0300: 2DC5           BUFFER  *     $0100  INPUT/OUTPUT BUFFER
0310:
0320: 2DC5           RESTRT  *     $2031  EDITOR WARM ENTRY ADDRESS
0330:
0340:
0350:                KIM ROM AND PIA ADDRESSES
0360:
0370: 2DC5           SBD     *     $1742  PIA LOCATION
 380: 2DC5           CHKL    *     $17E7  CHECKSUM
0390: 2DC5           CHKH    *     $17E8
0400: 2DC5           VEB     *     $17EC  VOLATILE EXECUTION BLOCK
0410: 2DC5           SAL     *     $17F5  TAPE START ADDRESS
0420: 2DC5           SAH     *     $17F6
0430: 2DC5           EAL     *     $17F7  TAPE END ADDRESS
0440: 2DC5           EAH     *     $17F8
0450: 2DC5           INTVEB  *     $1932  INIT VEB SUBROUTINE
0460: 2DC5           CHKT    *     $194C  CHECK SUM SUBROUTINE
0470: 2DC5           INCVEB  *     $19EA  INCREMENT VEB SUB
0480: 2DC5           RDBYT   *     $19F3  READ BYTE SUBROUTINE
0490: 2DC5           RDCHT   *     $1A24  READ CHAR SUBROUTINE
0500: 2DC5           RDBIT   *     $1A41  READ BIT SUBROUTINE
0510: 2DC5           INIT    *     $1E8C  RESET ALL PIAS
0520:
0530:
0540:
0550:
0560:
```

```
0570:
0580:
0590:                   ******** INPUT AND OUTPUT ROUTINES ********
0600:
0610:                   SUBROUTINE TO PRINT THE CURRENT LINE NUMBER
0620:
0630: 2DC5 A5 16        NOUT    LDA    N        +01 GET HI N
0640: 2DC7 A6 15                LDX    N            GET LO N...PRINT THEM
0650:
0660:                   SUB TO PRINT 2 HEX BYTES
0670:                   FIRST BYTE IS IN A
0680:                   SECOND BYTE IS IN X
0690:
0700: 2DC9 20 CD 2D     HEXAX   JSR    HEXOUT   PRINT ACCUMULATOR
0710: 2DCC 8A                   TXA             GET BYTE IN X ... PRINT IT
0720:
0730:                   SUBROUTINE TO PRINT 1 HEX BYTE
0740:                   INPUT IS IN ACCUMULATOR
0750:
 760: 2DCD 48           HEXOUT  PHA             SAVE INPUT
0770: 2DCE 4A                   LSRA            GET
0780: 2DCF 4A                   LSRA            UPPER
0790: 2DD0 4A                   LSRA            NYBBLE
0800: 2DD1 4A                   LSRA
0810: 2DD2 20 D8 2D             JSR    HEX      PRINT UPPER NYBBLE
0820: 2DD5 68                   PLA             GET INPUT BACK
0830: 2DD6 29 0F                ANDIM $0F       GET LOWER NYBBLE
0840:
0850:                   SUBROUTINE TO PRINT 1 HEX CHARACTER
0860:                   INPUT CHAR IS IN ACCUMULATOR
0870:
0880: 2DD8 C9 0A        HEX     CMPIM $0A       LETTER OR NUMBER?
0890: 2DDA 18                   CLC             CALCULATE ASCII
0900: 2DDB 30 02                BMI    HEXA     IF IT IS A NUMBER!
0910: 2DDD 69 07                ADCIM $07       ADD 7 TO LETTER
0920: 2DDF 69 30        HEXA    ADCIM $30       AND 30 TO BOTH
0930: 2DE1 D0 0D                BNE    OUTCH    THEN PRINT IT
0940:
 950:                   SUBROUTINE TO PRINT A BACKSPACE
0960:                   IF YOUR TERMINAL CAN'T-CHANGE THE 5F TO
0970:                   ANOTHER CHARACTER TO INDICATE DELETES
0980:
0990: 2DE3 A9 5F        BACKSP LDAIM $5F        BACKSPACE CHARACTER
1000: 2DE5 D0 09                BNE    OUTCH    PRINT IT
1010:
1020:                   SUBROUTINE TO PRINT CARRIAGE RETURN
1030:                   AND LINE FEED
1040:
1050: 2DE7 A9 0D        CRLF    LDAIM $0D       GET CR CHARACTER
1060: 2DE9 D0 05                BNE    OUTCH    AND PRINT IT
ID=02

0010:
0020:                   SUBROUTINE TO PRINT  2 HEX BYTES
0030:                   FOLLOWED IMMEDIATEDLY BY A SPACE
0040:
0050: 2DEB 20 C9 2D     HEXSP   JSR    HEXAX    PRINT 2 HEX BYTES
0060:
```

```
0070:                   SUBROUTINE TO PRINT A SPACE
0080:
0090: 2DEE A9 20        OUTSP  LDAIM ' '       LOAD SPACE IN A
0100:
0110:                   SUBROUTINE TO PRINT AN ASCII CHARACTER
0120:                   INPUT CHARACTER IS IN THE ACCUMULATOR
0130:
0140: 2DF0 84 F4        OUTCH  STY   YTMP   HIDE Y
0150: 2DF2 86 F5               STX   XTMP   AND X
0160: 2DF4 C9 0D               CMPIM $0D    IS THIS A CARRIAGE RETURN?
0170: 2DF6 D0 1E               BNE   NOCR   SKIP LF IF NOT
0180:
0190: 2DF8 A6 63               LDX   PMODE  CHECK PAGE MODE FLAG
0200: 2DFA D0 13               BNE   NOPG   SKIP IF NOT ON
0210: 2DFC E6 64               INC   COUNTL ADD 1 TO LINES PRINTED
0220: 2DFE D0 0F               BNE   NOPG   SKIP IF NOT END OF SCREEN
0230:
0240: 2E00 20 2B 2E            JSR   INCH   PAUSE UNTIL INPUT OF ANY KEY
0250: 2E03 C9 1B               CMPIM $1B    WAS ESCAPE KEY ENTERED?
0260: 2E05 D0 04               BNE   ON     IF NOT CONTINUE IN PAGE MODE
0270: 2E07 A9 FF               LDAIM $FF    TURN OFF PAGE MODE
0280: 2E09 85 63               STA   PMODE
0290:
0300: 2E0B A2 F0        ON     LDXIM $F0    RESET LINE COUNTER
0310: 2E0D 86 64               STX   COUNTL TO -16
0320:
0330: 2E0F A9 0A        NOPG   LDAIM $0A    PRINT A LINE FEED
0340: 2E11 20 16 2E            JSR   NOCR   (REMOVE IF YOUR TERMINAL HAS AUTO
0350:
0360: 2E14 A9 0D               LDAIM $0D    THIS WAS A CR, REMEMBER
0370: 2E16 20 A0 2E     NOCR   JSR   OUTPUT SO PRINT IT
0380:
0390:                   ROUTINE TO TEST FOR BREAK DURING I/O
0400:
0410: 2E19 2C 40 17     BRKTST BIT   $1740  TEST INPUT PORT OF PIA
0420: 2E1C 10 05               BPL   BREAK  IF BIT 7=0
0430:
0440: 2E1E A6 F5               LDX   XTMP   SEEK HIDDEN X
0450: 2E20 A4 F4               LDY   YTMP   AND HIDDEN Y
0460: 2E22 60                  RTS          AND ITS ALL OVER
0470:
0480: 2E23 2C 40 17     BREAK  BIT   $1740  WAIT UNTIL KEY
0490: 2E26 10 FB               BPL   BREAK  IS RELEASED
0500: 2E28 4C 31 20            JMP   RESTRT THEN GO TO EDITOR
0510:
0520:                   ROUTINE TO INPUT AN ASCII CHARACTER
0530:                   RETURNS IT IN ACCUMULATOR
0540:
0550: 2E2B 86 F5        INCH   STX   XTMP   HIDE X
0560: 2E2D 84 F4               STY   YTMP   AND Y
0570: 2E2F 20 9D 2E            JSR   INPUT  CALL USER INPUT ROUTINE
0580: 2E32 29 7F               ANDIM $7F    STRIP PARITY BIT
0590: 2E34 C9 0D               CMPIM $0D    WAS INPUT A RETURN
0600: 2E36 D0 07               BNE   NOCRIN IF NOT ITS OK
0610:
0620: 2E38 A9 0A               LDAIM $0A    PRINT A LF WITH CR INPUT
```

```
0630: 2E3A 20 A0 2E          JSR   OUTPUT SKIP THIS IF AUTO LF ON TERMINAL
0640: 2E3D A9 0D             LDAIM $0D    REPLACE CR AGAIN FOR RTS
0650: 2E3F D0 D8     NOCRIN  BNE   BRKTST RETURN VIA BREAK TEST
0660:
ID=03

0010:
0020:                        SUBROUTINE TO FILL BUFFER FROM
0030:                        KEYBOARD INPUT
0040:
0050: 2E41 A0 00     BUFIN   LDYIM $00    RESET BUFFER COUNTER
0060: 2E43 20 2B 2E  INB     JSR   INCH    GET CHARACTER INPUT
0070: 2E46 C9 7F             CMPIM $7F    WAS IT A DELETE?
0080: 2E48 D0 07             BNE   ONIN    IF NOT, CARRY ON
0090:
0100: 2E4A 20 E3 2D          JSR   BACKSP PRINT A BACKSPACE
0110: 2E4D 88                DEY          BACK UP IN BUFFER
0120: 2E4E 10 F3             BPL   INB     AND GET IT RIGHT THIS TIME
0130: 2E50 00                BRK          ERROR--BACKED UP TOO FAR!
 140:
0150: 2E51 C9 5C     ONIN    CMPIM $5C    (BACKSLASH) WILL
0160: 2E53 D0 06             BNE   OKB     DELETE WHOLE LINE
0170: 2E55 20 E7 2D          JSR   CRLF    PRINT RETURN AND LF
0180: 2E58 4C 41 2E          JMP   BUFIN   AND START OVER
0190:
0200: 2E5B C9 05     OKB     CMPIM $05    WAS IT A CTL-E?
0210: 2E5D F0 11             BEQ   OVR     YES--GO TO OVR FUNCTION
0220: 2E5F 99 00 01          STAAY BUFFER JUST AN ORDINARY CHARACTER TO SAV
0230:
0240: 2E62 C9 0D             CMPIM $0D    WAS THIS THE END?
0250: 2E64 F0 09             BEQ   ENDBU   YES, SO GET OUT OF HERE
0260: 2E66 C8                INY          INCREMENT POINTER
0270: 2E67 C0 3A             CPYIM $3A    ALLOW ONLY 58 CHARS +6 PROMPT=64
0280: 2E69 30 D8             BMI   INB     STILL SOME ROOM FOR MORE
0290: 2E6B A9 0D             LDAIM $0D    FORCE CR TO END LINE
0300: 2E6D D0 EC             BNE   OKB     PRINT IT AND PUT IN BUFFER
0310: 2E6F 60        ENDBU   RTS          ALL DONE
0320:
 330: 2E70 20 E3 2D  OVR     JSR   BACKSP CANCEL THE CTL CHAR (THIS MAY NOT
0340:                                      NECESSARY ON SOME TERMINALS)
0350: 2E73 B9 00 01  OVX     LDAAY BUFFER GET CHARACTER IN BUFFER
0360: 2E76 C9 0D             CMPIM $0D    IS IT THE END?
0370: 2E78 F0 C9             BEQ   INB     IF SO--GO GET NEW ADDITION
0380: 2E7A 20 F0 2D          JSR   OUTCH   SHOW HIM WHAT IT IS
0390: 2E7D C8                INY          ON TO NEXT CHARACTER
0400: 2E7E C0 3A             CPYIM $3A    BUT DON'T GET CARRIED AWAY! BUFFE
0410: 2E80 D0 F1             BNE   OVX     KEEP GOING                   LIN
0420: 2E82 F0 BF             BEQ   INB     LET HIM FIX IT UP
0430:
0440:                        SUBROUTINE TO PRINT THE BUFFER
0450:
0460: 2E84 A0 00     PRBUF   LDYIM $00    RESET THE POINTER
0470: 2E86 B9 00 01  PRNTB   LDAAY BUFFER GET A CHARACTER
0480: 2E89 48                PHA          HIDE IT TEMPORARILY
0490: 2E8A 20 F0 2D          JSR   OUTCH   PRINT IT
0500: 2E8D 68                PLA          SEEK IT BACK
0510: 2E8E C8                INY          POINT TO NEXT CHARACTER
0520: 2E8F C9 0D             CMPIM $0D    WAS THAT THE END OF THE BUFFER?
```

```
0530: 2E91 D0 F3            BNE   PRNTB  NO, THERE MUST BE MORE
0540: 2E93 60               RTS          YES, SO RETURN
0550:
0560:                     **** ********* *********
0570:                  ***** USER SPECIFIED ADDRESSES *****
0580:                     **** ********* *********
0590:
0600:
0610: 2E94 4C 00 1A  PACKT  JMP   $1A00  A KIM SUBROUTINE TO PACK ASCII IN
0620:                                                                    HE
0630: 2E97 4C AF 2E  READ   JMP   CREAD  THE CASSETTE READ SUBROUTINE
0640:
0650: 2E9A 4C 35 2F  WRITE  JMP   CWRITE THE CASSETTE WRITE SUBROUTINE
0660:
0670: 2E9D 4C 5A 1E  INPUT  JMP   $1E5A  THE KEYBOARD INPUT ROUTINE
0680:
0690: 2EA0 4C A0 1E  OUTPUT JMP   $1EA0  THE PRINTER OUTPUT ROUTINE
0700:
0710:                DEFINITION OF SOURCE LOCATION
 720:
0730: 2EA3 35        SOURCM =     $35    SOURCE - 1
0740: 2EA4 36        SOURCE =     $36    SOURCE AREA OF MEMORY STARTS HERE
0750: 2EA5 40        SOURCF =     $40    AND ENDS JUST BELOW HERE
0760:
0770:                DEFINITION OF SYMBOL TABLE LOCATION
0780:
0790: 2EA6 30        SYMBOL =     $30    SYMBOL TABLE STARTS HERE
0800: 2EA7 36        SYMF   =     $36    AND JUST BELOW HERE
0810:
0820:                DEFINITION OF OBJECT LOCATION
0830:
0840: 2EA8 02        OBJECT =     $02    THE OBJECT WILL BE ASSEMBLED TO H
0850:
ID=04


0010:
0020:                ***************************************************
0030:                ***** KIM CASSETTE READ AND WRITE ROUTINES *****
  40:                ***************************************************
0050:
0060:                CASSETTE READ ERROR (INCORRECT ID)
0070:
0080: 2EA9 20 CD 2D  ERID   JSR   HEXOUT PRINT THE WRONG ID,
0090: 2EAC 20 EE 2D         JSR   OUTSP  SPACE AND THEN START OVER
0100:
0110:                ***** CASSETTE READ SUBROUTINE *****
0120:
0130:                VERY SIMILAR TO THE READ ROUTINE
0140:                IN THE KIM ROM.
0150:                THE LED DISPLAY OF INCOMING DATA ADAPTED
0160:                FROM VUTAPE, A PROGRAM BY JIM BUTTERFIELD
0170:                FROM THE KIM-1 USER NOTES.
0180:
0190:
0200: 2EAF AD 02 17  CREAD  LDA   $1702  TURN ON CASSETTE #1 BY
0210: 2EB2 29 FB            ANDIM $FB    CHANGING BIT 2 TO
0220: 2EB4 8D 02 17         STA   $1702  ZERO IN PIA PORT B
0230:
```

```
0240: 2EB7 A9 7F          LDAIM $7F    TURN ON THE KIM LED DISPLAY
0250: 2EB9 8D 41 17       STA   $1741  BY SETTING THE DD REG
0260:
0270: 2EBC D8             CLD          JUST TO MAKE SURE
0280:
0290: 2EBD A9 8D          LDAIM $8D    SET UP VEB
0300: 2EBF 8D EC 17       STA   VEB    TO SAVE DATA
0310: 2EC2 20 32 19       JSR   INTVEB (IN KIM ROM)
0320:
0330: 2EC5 A9 13          LDAIM $13    TURN ON INPUT PORT FROM CASSETTE HA
0340: 2EC7 8D 42 17       STA   SBD
0350:
0360: 2ECA 20 41 1A SYNC  JSR   RDBIT  START READING A BIT AT A TIME
0370:
0380: 2ECD 46 F3          LSRZ  TMP    SHIFT IT INTO TMP
0390: 2ECF 05 F3          ORAZ  TMP
0400: 2ED1 85 F3          STAZ  TMP    AND SAVE IT
0410: 2ED3 8D 40 17       STA   $1740  PLACE IT ON THE LED
0420:
0430: 2ED6 C9 16      TST CMPIM $16    IS IT A SYNC CHARACTER?
0440: 2ED8 D0 F0          BNE   SYNC   IF NOT, KEEP TRYING
0441:
0450: 2EDA 20 24 1A       JSR   RDCHT  IN SYNC, READ A CHARACTER
0460: 2EDD 8D 40 17       STA   $1740  DISPLAY IT ON LED
0470: 2EE0 C9 2A          CMPIM $2A    IS IT THE START OF DATA?
0480: 2EE2 D0 F2          BNE   TST    IF NOT, LOOP AGAIN
0481:
0490: 2EE4 20 F3 19       JSR   RDBYT  READ THE TAPE ID
0500: 2EE7 C5 62          CMP   ID     IS THIS THE RIGHT TAPE?
0510: 2EE9 D0 BE          BNE   ERID   PRINT IT IF WRONG
0511:
0520: 2EEB 20 F3 19       JSR   RDBYT  READ THE START ADDRESS
0530: 2EEE 20 4C 19       JSR   CHKT   INCLUDE IT IN CHECKSUM
0540: 2EF1 8D ED 17       STA   VEB    +01 AND SAVE IT IN VEB
0550: 2EF4 20 F3 19       JSR   RDBYT  READ THE HI PART OF ADDRESS
0560: 2EF7 20 4C 19       JSR   CHKT   INCLUDE IN SUM
0570: 2EFA 8D EE 17       STA   VEB    +02 SET IT UP IN VEB
0571:
0572:
0580: 2EFD A2 02      LOADIT LDXIM $02    START TO LOAD DATA AS
0590: 2EFF 20 24 1A   READIT JSR   RDCHT  ASCII CHARACTERS
0600: 2F02 C9 2F          CMPIM '/     END OF DATA SYMBOL
0610: 2F04 F0 14          BEQ   ENDRD  SO WIND IT UP
0611:
0620: 2F06 20 94 2E       JSR   PACKT  PACK THE ASCII INTO HEX
0630: 2F09 D0 BF          BNE   SYNC   ERROR IN CHARACTER READ NOT = HEX
0640: 2F0B CA             DEX          COUNT TO TWO
0650: 2F0C D0 F1          BNE   READIT READ SECOND HALF
0651:
0660: 2F0E 20 4C 19       JSR   CHKT   ADD TO CHECKSUM
0670: 2F11 20 EC 17       JSR   VEB    STORE VIA VEB
0680: 2F14 20 EA 19       JSR   INCVEB INCREMENT STORE ADDRESS
0690: 2F17 4C FD 2E       JMP   LOADIT AND READ NEXT BYTE
0691:
0700: 2F1A 20 F3 19   ENDRD JSR   RDBYT  READ CHECKSUM FROM TAPE
0710: 2F1D CD E7 17       CMP   CHKL   COMPARE TO CALCULATED
```

```
0720: 2F20 D0 A8          BNE   SYNC    AND START OVER IF WRONG
0730: 2F22 20 F3 19       JSR   RDBYT   GET SECOND HALF OF SUM
0740: 2F25 CD E8 17       CMP   CHKH    AND DO THE SAME
0750: 2F28 D0 A0          BNE   SYNC    WITH IT
0751:
0760: 2F2A AD 02 17 OKRD  LDA   $1702   TURN OFF CASSETTE
0770: 2F2D 09 04          ORAIM $04     BY SETTING BIT 2
0780: 2F2F 8D 02 17       STA   $1702   OF THE PORT
0790: 2F32 4C 8C 1E       JMP   INIT    RETURN VIA INIT (RESET ALL PORTS)
0791:
ID=05

0010:                     ***** KIM CASSETTE WRITE SUBROUTINE *****
0020:
0030:                     ADAPTED FROM SUPERTAPE BY JIM BUTTERFIELD
0040:                     AS PUBLISHED IN KIM-1 USER NOTES (V I,N 2)
0050:
0060:
0070: 2F35 AD 02 17 CWRITE LDA   $1702   TURN ON CASSETTE #2
  80: 2F38 29 F7          ANDIM $F7     BY SETTING BIT 3 = 0
0090: 2F3A 8D 02 17       STA   $1702   IN PIA PORT B
0100:
0110: 2F3D A9 AD          LDAIM $AD     SET UP
0120: 2F3F 8D EC 17       STA   VEB     VEB FOR SAVE
0130: 2F42 20 32 19       JSR   INTVEB
0140:
0150: 2F45 A9 27          LDAIM $27     SET FLAG
0160: 2F47 85 F0          STAZ  GANG    FOR SBD LATER
0170:
0180: 2F49 A9 BF          LDAIM $BF     TURN ON
0190: 2F4B 8D 43 17       STA   $1743   OUTPUT TO CASSETTE
0200:
0210: 2F4E A2 F0          LDXIM $F0     SEND 240 SYNC PULSES (OPTIMUM # D
0220:                                    ON RECORDER START/STOP TIME)
0230: 2F50 A9 16          LDAIM $16     SYNC CHARACTER
0240: 2F52 20 A3 2F       JSR   HIC     OUTPUT X TIMES
0250:
0260: 2F55 A9 2A          LDAIM $2A     SEND START OF DATA CHAR
  70: 2F57 20 C6 2F       JSR   OUTCHT
0280:
0290: 2F5A A5 62          LDA   ID      GET ID
0300: 2F5C 20 B2 2F       JSR   OUTBT   AND SEND AS A BYTE
0310:
0320: 2F5F A5 60          LDAZ  SALX    SEND EXECUTION ADDRESS
0330: 2F61 20 AF 2F       JSR   OUTBTC  WITH CHECKSUM CALCULATION
0340: 2F64 A5 61          LDAZ  SAHX    HI PART TOO
0350: 2F66 20 AF 2F       JSR   OUTBTC
0360:
0370: 2F69 20 EC 17 DUMPTA JSR  VEB     GET A BYTE OF MEMORY
0380: 2F6C 20 AF 2F       JSR   OUTBTC  SEND AND CHECKSUM IT
0390: 2F6F 20 EA 19       JSR   INCVEB  POINT TO NEXT BYTE
0400: 2F72 AD ED 17       LDA   VEB     +01 CHECK FOR END
0410: 2F75 CD F7 17       CMP   EAL     AGAINST EAL
0420: 2F78 AD EE 17       LDA   VEB     +02 AND
0430: 2F7B ED F8 17       SBC   EAH     EAH
0440: 2F7E 90 E9          BCC   DUMPTA  AGAIN IF NOT END
0450:
0460: 2F80 A9 2F          LDAIM $2F     SEND END OF DATA CHAR
```

```
                        *
0470: 2F82 20 C6 2F           JSR   OUTCHT AS CHAR
0480:
0490: 2F85 AD E7 17           LDA   CHKL   SEND
0500: 2F88 20 B2 2F           JSR   OUTBT  CHECKSUM
0510: 2F8B AD E8 17           LDA   CHKH   LO AND
0520: 2F8E 20 B2 2F           JSR   OUTBT  HI
0530: 2F91 A2 02              LDXIM $02    AND SEND 2
0540: 2F93 A9 04              LDAIM $04    EOT CHARS
0550: 2F95 20 A3 2F           JSR   HIC
0560:
0570: 2F98 AD 02 17           LDA   $1702  TURN OFF CASSETTE
0580: 2F9B 09 08              ORAIM $08    BY SETTING BIT 3
0590: 2F9D 8D 02 17           STA   $1702  OF THE CONTROL PORT
0600: 2FA0 4C 8C 1E           JMP   INIT   RESET ALL PORTS
0610:
0620:                 SUBROUTINE TO SEND X CHARACTERS TO TAPE
0630:
0640: 2FA3 86 F1      HIC     STXZ  TIC    SAVE THE COUNT
0650: 2FA5 48         HICA    PHA          AND THE CHARACTER
0660: 2FA6 20 C6 2F           JSR   OUTCHT SEND THE CHAR
0670: 2FA9 68                 PLA          AND GET IT BACK
0680: 2FAA C6 F1              DECZ  TIC    TO SEND AGAIN
0690: 2FAC D0 F7              BNE   HICA   UNTIL COUNT = 0
0700: 2FAE 60                 RTS
0710:
0720:                 SUB TO SEND CHARACTER WITH CHECKSUM CALCULATION
0730:
0740: 2FAF 20 4C 19   OUTBTC  JSR   CHKT   ADD CHAR TO SUM
0750:
0760:                 SUB TO SEND BYTE AS TWO ASCII CHARS
0770:
0780: 2FB2 48         OUTBT   PHA          SAVE BYTE
0790: 2FB3 4A                 LSRA         GET
0800: 2FB4 4A                 LSRA         UPPER
0810: 2FB5 4A                 LSRA         NYBBLE
0820: 2FB6 4A                 LSRA
0830: 2FB7 20 BB 2F           JSR   HEXT   AND SEND IT
0840: 2FBA 68                 PLA          RETURN BYTE
0850:
0860:                 SUBROUTINE TO SEND ONE HEX CHAR AS ASCII
0870:
0880: 2FBB 29 0F      HEXT    ANDIM $0F    CLEAN UP DATA
0890: 2FBD C9 0A              CMPIM $0A    CHANGE TO ASCII
0900: 2FBF 18                 CLC          BY ADDING
0910: 2FC0 30 02              BMI   HEXAT
0920: 2FC2 69 07              ADCIM $07    37 TO A...F
0930: 2FC4 69 30      HEXAT   ADCIM $30    AND 30 TO 0...9
ID=06

0010:
0020:                 SUBROUTINE TO SEND ONE 8 BIT BYTE
0030:
0040: 2FC6 A0 08      OUTCHT  LDYIM $08    EIGHT BIT COUNT
0050: 2FC8 84 F2              STYZ  COUNT
0060: 2FCA A0 02      TRY     LDYIM $02    START AT
0070: 2FCC 84 FE              STYZ  TRIB   3600 HERTZ
0080: 2FCE BE FC 2F   ZON     LDXAY NPUL   NUMBER OF HALF CYCLES
0090: 2FD1 48                 PHA          SAVE THE CHAR
```

```
0100:
0110: 2FD2 2C 47 17  ZONA    BIT    $1747   WAIT FOR END OF CYCLE
0120: 2FD5 10 FB             BPL    ZONA    IN TIGHT LOOP
0130:
0140: 2FD7 B9 FD 2F          LDAAY  TIMG    SET UP TIMER
0150: 2FDA 8D 44 17          STA    $1744   FOR THIS PULSE
0160:
0170: 2FDD A5 F0             LDAZ   GANG    CHANGE STATE
0180: 2FDF 49 80             EORIM  $80     OF OUTPUT
0190: 2FE1 8D 42 17          STA    $1742   PORT
0200:
0210: 2FE4 85 F0             STAZ   GANG    AND SAVE STATE
0220: 2FE6 CA                DEX            DONE ALL CYCLES?
0230: 2FE7 D0 E9             BNE    ZONA    NO-THEN SEND ANOTHER
0240:
0250: 2FE9 68                PLA            RETRIEVE BYTE
0260: 2FEA C6 FE             DECZ   TRIB    ONE MORE GONE
0270: 2FEC F0 05             BEQ    SETZ    THE LAST ONE, TOO
0280: 2FEE 30 07             BMI    ROUT    EVEN THE LAST ONE WENT
0290:
0300: 2FF0 4A                LSRA           ANOTHER BIT TO THE CARRY
0310: 2FF1 90 DB             BCC    ZON     IF IT IS NOT SET
0320: 2FF3 A0 00       SETZ  LDYIM  $00     SWITCH TO 2400 HZ
0330: 2FF5 F0 D7             BEQ    ZON     ALWAYS
0340:
0350: 2FF7 C6 F2       ROUT  DECZ   COUNT   ONE BIT SENT
0360: 2FF9 D0 CF             BNE    TRY     BUT MORE TO GO
0370: 2FFB 60                RTS            ALL OVER, GO HOME
0380:
0390:                  TIMING TABLE
0400:
0410: 2FFC 02          NPUL  =      $02     TWO PULSES
0420: 2FFD C3          TIMG  =      $C3     THE RIGHT TIME
0430: 2FFE 03                =      $03     3 PULSES
0440: 2FFF 7E                =      $7E     AND ENOUGH TIME
0450:
0460:                  IF YOUR RECORDER CANNOT HANDLE THIS SPEED,
0470:                  YOU CAN SLOW DOWN BY CHANGING NPUL AND NPUL+02
0480:                  TO ONE OF THE FOLLOWING:      03       04
0490:                                                06       09
0500:                  (THIS IS THE KIM ROM SPEED)   0C       12
ID=
```