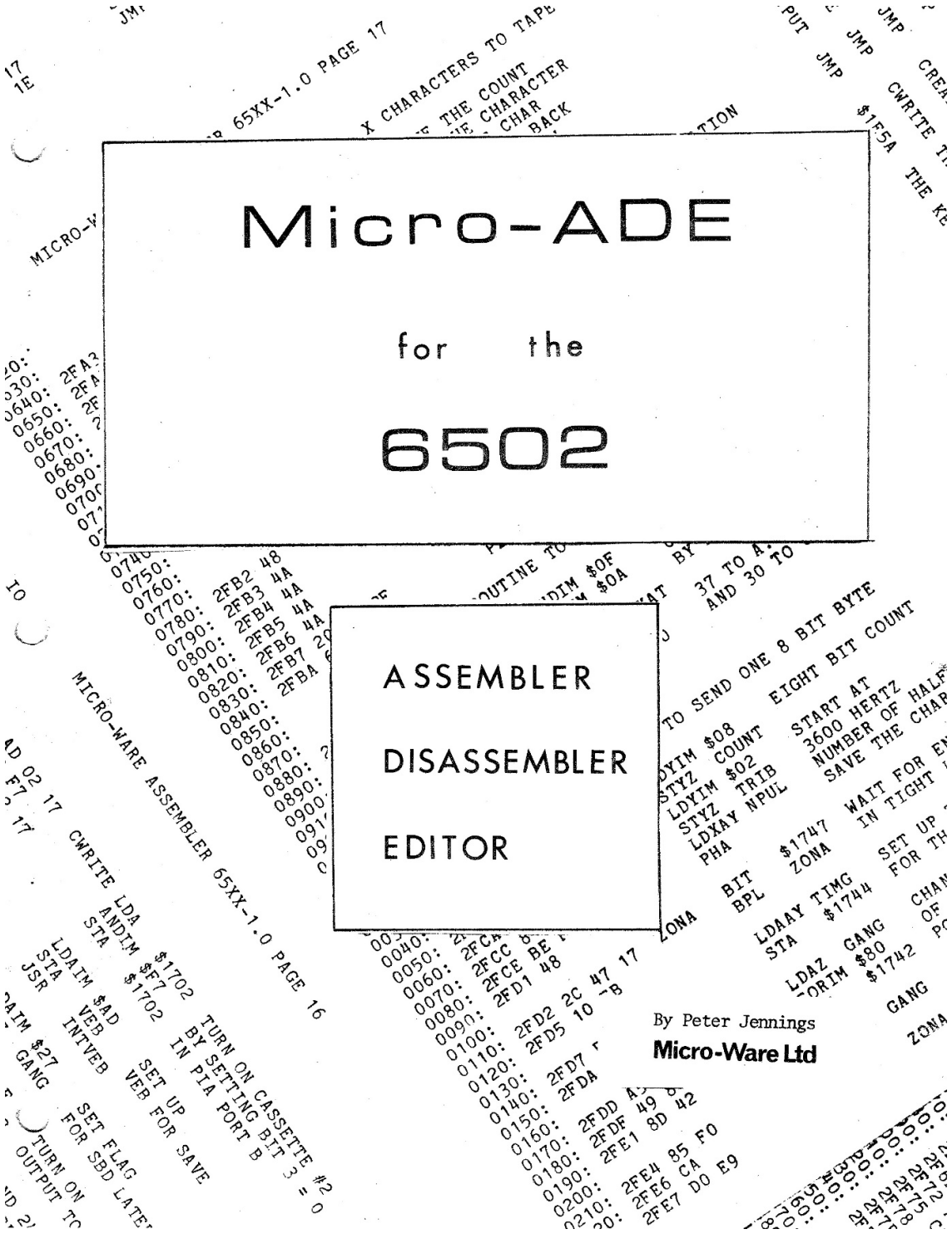


Micro-ADE

for the

6502

ASSEMBLER

DISASSEMBLER

EDITOR

By Peter Jennings

Micro-Ware Ltd

Micro-ADE

for the

6502

ASSEMBLER

DISASSEMBLER

EDITOR

By Peter R. Jennings

Micro-Ware Ltd 27 FIRSTBROOKE ROAD, TORONTO, ONTARIO, CANADA. M4E 2L2.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

SYSTEM DESCRIPTION

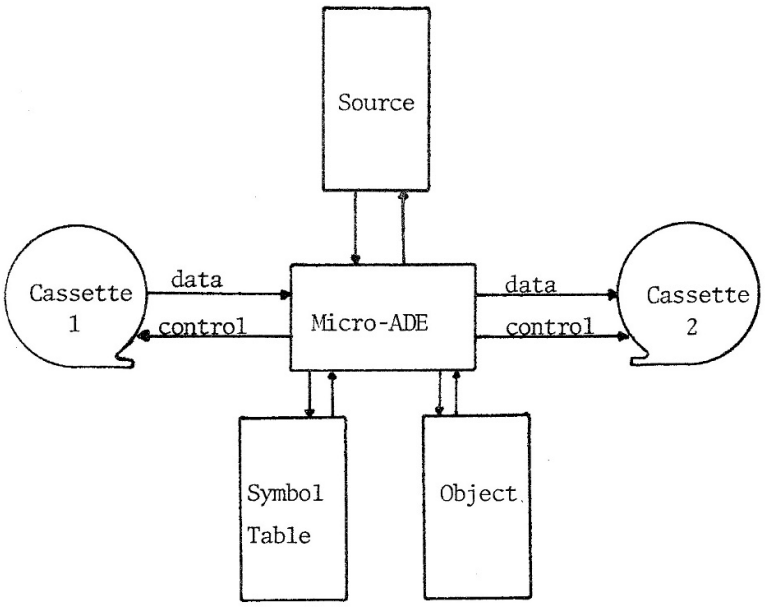
The Micro-ADE system is designed for use with any 6502 microcomputer and consists of three major programs as well as a number of utility programs. The major programs are an assembler, a disassembler, and a text editor.

The assembler is used to create machine executable code for the 6502 from a symbolic input source program. Small programs can be created and tested directly in memory. Larger programs may be written using cassette tapes for source input and object output.

The disassembler is used to list executable 6502 machine code in the symbolic assembler source format. Symbols are generated if they are defined in the symbol table.

The text editor is used to create source programs in the format required for the assembler. It contains the necessary routines for easy manipulation of text data in memory or from cassette files.

The minimum system configuration for full use of all Micro-ADE features consists of a 6502 CPU, 8K of random access memory, 2 cassette recorders with start/stop control, and an ASCII input/output device. It is possible to use all parts of the system in a restricted way with less memory and a single manually operated cassette recorder.



SYSTEM ENTRY

Before executing the program, the NMI vector ($17FA, $17FB on the KIM) may be initialized to return control to the Micro-ADE editor at the warm-start entry point ($2031 in version 1.0) so that a hardware interrupt such as the [ST] key on the KIM, may be used to break the program.

Initial entry into the Micro-ADE system is made via the cold-start entry point (Address $2000 in version 1.0). All hexadecimal values will be preceded by a dollar sign throughout this manual. The editor CLEAR command is automatically executed, and the system will prompt "NEW?". If you respond with Y or YES, the source workspace will be cleared and formatted for new data entry. Micro-ADE will indicate this condition by displaying "CLEAR", and will then issue the ready prompt (-).

```
KIM
0000 23 2000
2000 D8 G

NEW?YES (r)

CLEAR
-
```

(r) will be used to indicate the carriage return throughout this manual.

You are now in the editor command mode. Any valid command may be entered.

At this point, if you are using cassette files, the input tape should be loaded onto cassette 1, and it should be turned on in PLAY position. A blank tape should be loaded onto cassette 2, and it should be turned on in RECORD position. Always check your tape recorders for proper operation before continuing further.

THE EDITOR

EDITOR COMMAND MODE

The editor and command mode for the Micro-ADE system indicates that it is ready to accept commands by printing a hyphen (-). Commands must begin in the first column after this prompt. They may be abbreviated to a single letter, or a single word of any length may be used. The first argument may begin immediately at the end of the command unless it is a hexadecimal argument beginning with one of the letters A through F. One or more spaces must separate these arguments from the GET, SAVE, XEQ, or REPRODUCE command. The second and third arguments are delimited by commas. Finally, the command input string must be terminated with a carriage return. The following are valid commands:

```
-L10(r)
-LIST10(r)
-L      0010(r)
-L 10,30(r)
```

DEL and ctl-E

Command lines may be edited using the DEL (NUL or RUBOUT) key to delete the last character entered. The ctl-E character also operates in command mode to allow you to copy the previously entered command again. For example, if you have entered "-SAVE A3,2000,3000(r)", and the operation has been carried out, you may now type ctl-E to the input prompt, and the command will be returned to the input buffer. It is possible to delete parts of the command before typing RETURN to begin execution. This feature is particularly useful for making multiple copies of a file.

EDITOR COMMANDS

A The ADD Command

The ADD command is used to add new lines to the end of the source file. Upon typing ADD to the editor command prompt, Micro-ADE will respond with the line number of the next new line of source. You may now type data into workspace, terminating each line with a carriage return. After each line, Micro-ADE will prompt with the line number of the next line to be added. When you have completed your final line, and terminated it with a carriage return, respond to the next new line prompt with the Micro-ADE end of data character @ ($40), and a carriage return.

```
-ADD(r)
0110: THIS IS A NEW LINE(r)
0120: THIS IS THE NEXT NEW LINE(r)
0130: @(r)
-
```

C The CLEAR Command

The CLEAR command may be used at any time to delete all the data in the workspace and format it for new data. Upon typing CLEAR to the command prompt, Micro-ADE will respond with the question "NEW?". This prevents the accidental clearing of the workspace by a typing error. If you respond Y or YES to the prompt, the workspace will be cleared of all data and prepared for new data entry. It is usually a good idea to clear the workspace before loading a new file from cassette. When the Micro-ADE system is entered from the cold-start entry point, the CLEAR command is automatically executed.

```
-CLEAR(r)
NEW?YES(r)
CLEAR
-
```

D The DELETE Command

The DELETE command is used to delete one or more consecutive lines of source. Typing D i causes the editor to delete the line with number i. Typing D i,j causes the editor to delete the block of lines beginning with line i and ending with line

j. If there are a large number of lines to be deleted, this command may require several seconds to execute. When the deletion is complete, the editor ready prompt will be displayed.

```
-DELETE 20,40(r)
-
```

E The END command

The END command is used to determine how much memory of the allocated source workspace is remaining. Micro-ADE responds to the END command with the absolute address and line number of the last line of source.

```
-END(r)
2FCA 1990
-
```

F The FIX Command

The FIX command is used to fix or modify a single line and insert new lines immediately after it. After typing FIX i, Micro-ADE will print line i and prompt with the line number. You may now type in a new line, or you may edit the existing line with the ctl-E and DEL keys.

The ctl-E character causes the editor to copy the existing line from the current character to the end of the line. A RETURN may then be used to end the edit sequence. If there is nothing to be changed in the line you are FIXing, type ctl-E and RETURN to leave the line unchanged.

The DEL key causes a backspace of the input buffer over the previous character. Deleted characters may be returned again by use of the ctl-E.

For example, to replace the third character from the end of a line, one may type ctl-E,DEL,DEL,DEL, the new character, ctl-E, RETURN. The REPEAT key available on many terminals makes this a very fast method of line editing.

After you have typed RETURN, Micro-ADE will prompt with a new line number one higher than the previous one. You may continue to insert new lines at this point until you have completed your modification of the source. When you are completely finished

with your editing, type the end of data character (@) and a RETURN. The NUMBER command should be used as soon as possible after inserting new lines.

```
-FIX 2500(r)
2500: LINE 2410
2500: (ctl-E)LINE 2410(DEL)(DEL)(DEL)5(ctl-E)(r)
2501: @(r)
-L 2500(r)
2500: LINE 2500
-
```

I The INSERT Command

The INSERT command is used to insert one to nine new lines between two existing lines. Upon typing INSERT i, Micro-ADE will respond with a new line number equal to i-9. You may now enter new data in the space immediately before line i, terminating each new line with a carriage return. When you have inserted as many lines as you wish, enter the end of data character (@), followed by RETURN. The NUMBER command should be executed as soon as possible after new lines have been inserted.

If, due to a previous FIX or INSERT, there is not a space of nine lines at the point where you wish to insert a new line, it is necessary to renumber before executing the INSERT command.

```
-INSERT 100(r)
0091: AN INSERTED LINE(r)
0092: AND ONE MORE(r)
0093: @(r)
-NUMBER(r)
-
```

L The LIST Command

The LIST command is used to display the file at the terminal as it has been entered. LIST may have 0,1, or 2 parameters. LIST alone causes Micro-ADE to list the entire file. L i, causes the editor to list only line number i, and Li,j causes the editor to list line i and all subsequent lines up to and including line j. The BREAK key may be used at any time to interrupt the listing procedure and return you to the command prompt.

```
-LIST 300,310(r)
300: THIS IS LINE 300
310: THIS IS LINE 310
-
```

M The MOVE Command

The MOVE command is used to change the order of existing lines by moving one or more of them to another location. If used with two parameters, MOVE i,j, the single line j will be moved to a new position immediately before line i. If three parameters are used (M i,j,k), the block of lines beginning with line j and ending with line k will be moved to a new location immediately before line i.

If a large block of lines is being moved, this command may take a few seconds to execute. All of the inserted lines will be numbered 0000 after the move. It is necessary to use the NUMBER command as soon as possible after a move to renumber the lines in proper sequential order.

```
-L 10,40(r)
0010: TEN
0020: TWENTY
0030: THIRTY
0040: FORTY
-MOVE 20,30,40(r)
-LIST 10,40(r)
0010: TEN
0000: THIRTY
0000: FORTY
0020: TWENTY
-N(r)
-
```

N The NUMBER Command

The NUMBER command may be used at any time to renumber all lines in the workspace in a sequence of tens, starting at line number 0010. This command should always be used as soon as possible after executing the INSERT, FIX, or MOVE commands to prevent accidental errors which may occur from having two lines with the same number.

W The WHERE Command

The WHERE command is used to locate the absolute address of a particular line. This may be necessary to correct errors caused by a program bug, or a bad cassette read, if the editor cannot follow the non-ascii characters created, or if it is necessary to delete a line with the end of file character in it.

```
-WHERE 30(r)
210A 0030: THIS IS LINE 30
-
```

CASSETTE COMMANDS

G The GET Command

The GET command is used to load a file into memory from cassette tape. It must be followed by the hexadecimal identification of the file.

When Micro-ADE receives a GET command it switches on the input cassette recorder (cassette 1) using the remote input jack. The recorder should first be prepared in PLAY position with the appropriate cassette loaded and cued.

Read Status Indicator

As the read operation begins, the right hand digit of the KIM LED display will show the status of the read. When searching between data files, the random cassette noise will be displayed as a slowly oscillating set of random characters. If there is data present, but it is not being loaded, the display will be less bright and show an 8. When the cassette read software detects the stream of sync characters at the beginning of the data block, it will display the "sync locked" pattern ( ). Finally, as the data is being loaded into memory, it will display the "data loading" pattern ( ). If the display is motionless or blank when the GET command is first executed, the cassette recorder is not working properly. By watching the patterns on the LED it is usually possible to judge the status of the cassette read operation, and to detect the source of possible errors.

False ID

If an attempt is made to read a cassette file with an incorrect ID, the false ID read from the tape will be typed at the terminal for your information. Micro-ADE will then ignore the data, and continue to search for the correct block.

Multiple Files

Provision has been made to automatically read multiple files from the same cassette, provided that they were written with sequential identifiers. The command GET A1,A4(r) will cause Micro-ADE to search for file A1, load it, search for A2, load it, and so on until A4 has been loaded into memory. If a read error of any kind occurs during a cassette load, the read routine reverts to the search operation. This allows you to rewind the cassette and make a second read attempt. If you are unsure of the reliability of your cassette, it may be advisable to record two copies of each file. If an error occurs in

reading the first copy, the routine will automatically revert to the search operation and read the second copy when it comes to it.

```
Load 1 file          Attempt to load A1      Load files A1, A2
with ID  A1          but A2 is on tape       A3 and A4
-GET A1(r)          -GET A1(r)              -GET A1,A4(r)
-                    A2                     -
```

As soon as the data has been successfully loaded into memory, Micro-ADE will turn off the cassette and return you to the editor command mode.

Since the BREAK key is disabled during cassette read operations it is necessary to use either the [RS] or [ST] keys to interrupt the program. If the NMI has been set up to return to the editor, the [ST] key will return you directly to the editor command mode.

The SAVE Command

The SAVE command is used to write a file to the output cassette (cassette 2). Before executing the SAVE command, the recorder should be prepared with a blank cassette properly cued, and left in the RECORD position. Immediately after the SAVE command has been entered, the system will turn on the output cassette recorder and print the start and end addresses of the file at the terminal.

Source Files

Source files may be saved using the SAVE or S x commands. The S command without parameters will cause the system to save the resident source file with the same ID as the last file accessed (presumably the read operation of the same file before editing). The start address of the saved file will be the first address of the memory allocated to the source. The end address will be determined by the location of the end of file record at the end of the source program. If the SAVE x command is used, the ID of the saved file will be x, where x may be any two digit hexadecimal value.

Data Files

The general three parameter form of the SAVE command may be used to save files of data or source from anywhere in memory. S x,a,b causes the system to save a block of data from address a to address b-1 with ID = x. This data file may be loaded again using either the GET command or the usual KIM cassette load routine at $1873.

```
-S 77,2000,3000 will save the Micro-ADE program.
-

-S      will save the current source file with its old ID.
-

-S F7     will save the current source file with ID = F7.
-
```

The REPRODUCE Command

The REPRODUCE command is used to reproduce a source file from the input cassette on the output cassette. This is a very handy feature of Micro-ADE for editing a multiple file source.

Entering R x will cause the system to execute a GET x command followed immediately by a SAVE command. Thus, the file with ID = x will be loaded from the input cassette player and written to the output cassette player.

Multiple Files

The command R x,y will cause the set of files with the sequential identification x,x+1,...,y to be copied to the output cassette.

It is important to remember that this command can only be used to reproduce source files because the save parameters are generated from the data, not from the read operation.

```
-R A1,A9      will reproduce files A1, A2, ... A9
-
```

OTHER COMMANDS

B The BLOCKMOVE Command

The BLOCKMOVE command may be used to move a page or less of data from one memory location to another. The command B a,b will cause the relocation of the data from address a through a+FF to the new location b through b+FF. If less than a full page of data is to be moved, a third parameter, the number of bytes, can be added. B a,b,x will cause the movement of the block [a,a+x-1] to the new area [b,b+x-1].

Overlapping blocks

Because of the manner in which the BLOCKMOVE command operates, it is not possible to move a block to a lower address than its initial position if the end of the new block will overlap the start of the old block. To perform this move, it would be necessary to move the data to an unused page first, and then move it from there to the new location. It is possible to move overlapping blocks to a higher address. Remember, however, that if more than one page is to be moved, the highest page must be moved first or the overlap will write over some of the unmoved data.

```
-B 200,3E00      will cause the data from [200,2FF] to
                 be relocated to [3E00,3EFF]

-B 300,3F00,40   will cause the data from [300,33F] to
                 be relocated to [3F00,3F3F]
```

P The PAGE Command

The PAGE command may be utilized by users with CRT terminals in order to break up all output into 16 line blocks. By typing PAGE, the Page Mode is either set or reset depending upon its status immediately before the command was entered. When in Page Mode, the system counts the number of lines which have been displayed (including input lines). When this number reaches 16, the system will pause and wait for a key to be pressed. Usually a space or other non-printing character is entered, and the output continues. This feature is especially useful for long searches with the LIST command, or for examining the output from the assembler on a CRT.

When the system pauses for an input at the 16th line, it is possible to escape from Page Mode by entering the ESCAPE (ALT-MODE) key. The system will reset the Page Mode flag and continue the output without interruption.

```
-PAGE
-
```

X The XECUTE Command

The XECUTE command is used to execute programs directly from the editor command mode. If no address is entered after an X command, the system will execute the assembler.

If an address parameter is used with the X command, the system will JUMP to that address and begin executing the user program. The user program can return to the editor command mode by executing a JMP to the restart entry point, or a BRK instruction if the IRQ vector was not changed. The restart address is $2031 in version 1.0.

```
-X          will execute the assembler.

-X 200      will execute a program at $0200.
```

THE ASSEMBLER

The Micro-ADE assembler is designed to make programming the 6502 microcomputer as easy as possible. A source program must first be created using the text editor and following the format described below. If the program is short, it can reside in the memory space allocated for source and be executed in memory. If it is long, it must be broken into segments which are stored on cassette tape.

Upon execution, the assembler translates the source statements you have written into machine instructions which will execute on the 6502 microcomputer. This is a two step process. During pass one, the assembler reads the source statements from memory, or in blocks from the cassette, and generates a symbol table which consists of all the symbols defined by the user, and their hexadecimal equivalent addresses or data. This table is stored in memory. During the second pass, the assembler reads the source statements and references the symbol table to generate the object code which is machine executable. The object code is saved in memory or in short blocks on the output cassette.

Once the program has been assembled, if there were no errors flagged by the assembler, the user can load the object code to its execution address and test it for operation.

SOURCE FORMAT

The input data for the assembler is formatted in blocks of variable length records. Each record contains a two byte hex line number, followed by 0 to 64 bytes of data, and terminated by a carriage return ($0D).

The source data is located in a previously defined area of memory consisting of at least one 256 byte page. Each block of data consists of a variable number records and is terminated with an end of file record consisting of a line number and the end of data character (@ = $40). The @ character is reserved in the Micro-ADE assembler, and may not be used except as the end of file indicator.

An initial carriage return is located in the first location of the source block. This byte is defined by the editor when executing the CLEAR command.

The source data format is shown below:

$0D	n	n	0 to 64 data bytes	$0D	n	n	data	$0d	n	n	$40	$0D

DATA FORMAT

Each source statement for the assembler can be divided into five fields. These are the label, the instruction, the address mode, the argument, and the comment.

Each field is delimited by a single space ($20), except for the address mode. In many cases, a field may not be present. If so, its absence must be shown by the leaving of a single space. It is important to remember that since spaces are used as delimiters, the number of spaces left between each field is critical.

The format of each statement is:

LABEL	Ø	INSTRUCTION	ADDRESS MODE	Ø	ARGUMENT	Ø	COMMENT

THE LABEL FIELD

Any program statement may be identified with a symbolic label. A label can contain from one to six alphabetic characters. No special symbols or numerals may be included in a symbol in this assembler. The label must always begin in the first column of the record. It is important to remember that symbols must be unique. That is, any symbol must be defined only once in a given program. The assembler will flag a duplicate symbol error if an attempt is made to create two identical symbols.

If the symbol is used as a label on any line, other than one containing the define symbol pseudo instruction (*), the symbol will be equated to the current address as calculated by the assembler for that line. The define symbol instruction may be used anywhere in a program to define a symbol in terms of a special address or hexadecimal constant. If a reference is made to a symbol as an argument at any point in a program, the assembler will automatically substitute the equivalent address or hexadecimal constant for the symbol.

Although most symbols may be defined anywhere in a program, symbols referring to page zero addresses must normally be defined before they are used in order that the assembler can correctly calculate the number of bytes required for the instruction on the first pass. If it is necessary to define a

INSTRUCTIONS AND ADDRESS MODES

	IM	A	AX	ZX	AY	ZY	IX	IY	Z	(abs)	(rel)	(imp)
ADC	X		X	X	X		X	X	X	X		
AND	X		X	X	X		X	X	X	X		
ASL		X	X	X					X	X		
BCC											X	
BCS											X	
BEQ											X	
BIT									X	X		
BMI											X	
BNE											X	
BPL											X	
BRK												X
BVC											X	
BVS											X	
CLC												X
CLD												X
CLI												X
CLV												X
CMP	X		X	X	X		X	X	X	X		
CPX	X								X	X		
CPY	X								X	X		
DEC			X	X					X	X		
DEX												X
DEY												X
EOR	X		X	X	X		X	X	X	X		
INC			X	X					X	X		
INX												X
INY												X
JMI										X		
JMP										X		
JSR										X		
LDA	X		X	X	X		X	X	X	X		
LDX	X				X	X			X	X		
LDY	X		X	X					X	X		
LSR		X	X	X					X	X		
NOP												X
ORA	X		X	X	X		X	X	X	X		
PHA												X
PHP												X
PLA												X
PLP												X
ROL		X	X	X					X	X		
ROR		X	X	X					X	X		
RTI												X
RTS												X
SBC	X		X	X	X		X	X	X	X		
SEC												X
SED												X
SEI												X
STA			X	X	X		X	X	X	X		
STX						X			X	X		
STY				X					X	X		
TAX												X
TAY												X
TSX												X
TXA												X
TXS												X
TYA												X

page zero symbol after its first use, you can use the Z addressing mode instead of allowing the assembler to automatically update an absolute addressing mode. See the Address Mode section for further details of this topic.

It is generally considered good programming practice to define all data symbols at the beginning of the program. This keeps them together for easier editing or relocation and prevents the possibility of referencing a page zero symbol before it is defined.

Valid symbol usage	Invalid symbol usage
DATA LDA X	DATA1 LDA X3
TEST = $03	TEST SBCIM $03
SUB * TEST +01	TEST+1 = DAT A

INSTRUCTION FIELD

The second field of each source data record is the instruction field. It must always be separated from the last character of the label by exactly one space. If no label is present, the instruction field will always begin in column two. Instructions consist of three character mnemonics for 6502 CPU operations. These mnemonics are exactly the same as the MOS Technology instructions found on the reference card, or in the Programming Manual with the single exception of the jump indirect instruction. This is represented for the Micro-ADE assembler as a separate instruction, JMI, instead of as a JMP with a special address mode. A complete table of instructions and the valid address modes for each is shown below.

PSEUDO INSTRUCTIONS

There are three pseudo instructions which may be used in the Micro-ADE assembler. These are: "ORG", which is used to define the origin address for the program; "*", define symbol, which is used to define a symbol directly; and "=", define byte, which is used to define a byte directly.

ORG

The ORG instruction is used to define the origin address for the program being assembled. It should always be placed at the beginning of any program. If a label is placed on the ORG statement, it will become part of the header line printed at the top of each page. Any valid argument may be used to define

the origin address, and comments may be placed on the line in the usual way.

Normally, the ORG instruction should only be used once in a program. If it is necessary to redefine the origin in the middle of a program, the new origin must be the first statement of a NEW cassette file. The addreses saved with a cassette object block, which allow it to be loaded to the correct location, are based upon the ORG statement, and therefore must be unique for each block generated. One object block is generated for each source block.

EDITOR ORG $2000 VERSION 1.0 (77.07.01)

* The DEFINE SYMBOL Instruction

The define symbol instruction, *, may be used at any point to define the label field as equivalent to the following argument field. Once defined, symbols may be used in any type of instruction as an argument. The assembler will substitute the hexadecimal value defined for the symbol. The program address is not altered by a define symbol instruction. This is the only type of statement (other than a comment) which may precede the ORG statement.

```
ZERO * $0000 defines the symbol ZERO as equivalent to $0000
THREE * ZERO +03 defines THREE as equivalent to $0003
QMARK * '?    defines the symbol QMARK as equivalent to $3F
```

= The DEFINE BYTE Instruction

The define byte instruction, =, is used to directly define a single byte of memory. It is usually used to construct a data table. The argument following may be symbolic, hexadecimal, or ASCII.

```
 = $33    defines the current byte as $33
 = '?     defines the current byte as $3F
```

ADDRESS MODE

The address mode consists of zero, one, or two characters immediately following the instruction field. No space is required before the address mode field. Since the address mode is often implied directly by the instruction, it may in some cases be omitted. If no mode is given, and the instruction is not a relative branch, an implied register operation, or a pseudo instruction, the absolute mode is assumed.

The valid address modes are:

A Accumulator addressing. The instruction operates on the accumulator.

IM Immediate. The operand of the instruction is the argument following. The argument may be any valid symbolic, hexadecimal, or ASCII constant.

AX Absolute indexed by X. The operand of the instruction is the address represented by the argument added to the value of the X index. If the argument represents a page zero location, and if a valid page zero instruction exists, the assembler will automatically substitute the ZX address mode.

ZX The operand of the instruction is the sum of the address represented by the argument and the value of the X register. The high byte of the address will be ignored and the effective address will always be in page zero.

AY Absolute indexed by Y. The operand is the address represented by the argument plus the value of the Y index. If the argument is in page zero, and a valid zero page instruction exists, the ZY mode will be automatically used by the asembler.

ZY Zero page indexed by Y. The address of the argument is added to the Y index to form the effective address in page zero.

IX Indexed Indirect. The argument address is added to the X index which points to a location in page zero. The memory location pointed to by the page zero address calculated and the subsequent location is used as the operand for the instruction.

IY Indirect Indexed. The argument points to an address in page zero. The contents of that memory location and the subsequent location are added to the Y register to form the effective address of the operand.

Absolute. Absolute indexing is the default mode. The effective address is given directly by the argument. If the argument is a page zero location, the assembler will automatically substitute the appropriate zero page address mode.

Z Zero Page. The argument is assumed to be an address in page zero. The contents of this memory location are the argument for the operation. If the argument is not a page zero address, the high byte will be ignored.

Relative. Relative instructions cause a branch to within 128 bytes of the current address. Since this type of instruction is easily distinguished from all others, the address mode need not be explicitly defined.

Implied. Implied addressing requires no specification because the operand of the instruction is an internal register and is defined by the instruction itself.

Indirect. There is no indirect mode in the Micro-ADE assembler. The JMP indirect instruction is replaced by the JMI instruction which has an absolute address mode. The JMI instruction sets the program counter to the contents of the memory location pointed to by the argument and the subsequent location.

The assembler will flag most common address mode errors. Although it will not detect illogical use of an address mode (e.g. ASLIM), it will always detect illegal but logical address mode misuse (e.g. LSRAY).

THE ARGUMENT FIELD

The argument field is used to define the operand for an instruction or a pseudo instruction. There are three basic types of arguments which may be used with the Micro-ADE assembler. These are symbolic, hexadecimal, or ASCII.

Symbolic Arguments

Symbolic arguments are symbols defined elsewhere in the program. The equivalent address or data is substituted for the symbol in the object code. If the symbol refers to a page zero address, it should be defined before it is used. If it is not a page zero address, it may be defined anywhere in the program.

Modified Symbolic Arguments

In order to conserve the memory required for the saving of the symbol table, or in order to access part of a data table, it is sometimes necessary to define an argument in terms of a symbol with an offset. Offsets may be defined by appending a positive or negative value to the symbol. A single space should be left between the symbol and the operator (+ or -). The offset itself is a two digit hexadecimal value between 00 and FF. It must be exactly two characters long. For example, if BUFFER has been defined by a define symbol statement as being equivalent to address $0100, then BUFFER +03 may be used to represent address $0103.

If a symbol is referred to by an immediate operation, the low byte of the symbol is used as the operand. It may be necesary in some cases to reference the high byte of a symbol in order to set up an indirect table reference. This may be accomplished by appending a "/" symbol to the symbol. A single space should be left between the symbol and the slash. An example of the use of this operation is shown below:

```
0020: KIM * $1C00          0200              KIM   *      $1C00
0030:   LDAIM KIM          0200 A9 00              LDAIM  KIM
0040:   STA NMI            0202 8D FA 17           STA    NMI
0050:   LDAIM KIM /256     0205 A9 1C              LDAIM  KIM     /256
0060:   STA NMI +01        0207 8D FB 17           STA    NMI     +01
```

(The 256 shown after the slash is actually a comment.)

Hexadecimal Arguments

Hexadecimal arguments are identified by a dollar sign as the first character of the argument field. The following hex constant may be one or two bytes in length. Offsets may not be used with hexadecimal arguments.

Sample arguments would be: $0100 $0D

Character Arguments

ASCII arguments are identified by a single quotation mark (') as the first character in the argument field. A single character may be defined, the hexadecimal value of which, will be used as the operand for the instruction.

For example: = 'A CMPIM 'Y

Note that the @ character may not be used as an argument in this way because of its special end of file significance. Use $40 to represent the @ character if necessary.

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 have 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 the ID, read the next block, and so on until the last block of records has been assembled. If the source is resident in memory already, 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 asembler 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 addresses 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 allows 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 ADDRESES

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.

ADDITIONS TO SYMBOL TABLE

If you wish to add to an earlier symbol table, rather than create a new one, you may execute OLDST ($2601) without resetting the symbol table parameters. 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 occurred during an asembly which caused a break in execution, you may wish to continue from the point where you left off in order to check the source for syntax 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.

THE DISASSEMBLER

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 have 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 it 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 necesary changes. Further changes are usually not necessary, but if they are, it may be neccessary 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 occurrences 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 0A       NOPG   LDAIM  $000A
2E11 20 16 2E           JSR    NOCR
2E14 A9 0D              LDAIM  $000D
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
-
```

```
KIM
29FF 20 F2
00F2 04 FF.
00F3 0D 17FA
17FA 00 31.
17FB 1C 20.
17FC 00 2000
2000 D8 G

NEW?Y

CLEAR
-ADD

0000:    SHORT MESSAGE PROGRAM
0010: EXAMPL ORG $0200
0020: INDCT * $0066
0030: OUTCH * $2DF0 PRINT CHAR
0040:  LDAIN MESSG
0050:  STA INDCT
0060:  LDAIM MESSG /256
0070:  STS INDCT +01
0080:  LDYIM $00
0090: LOOP LDAIY INDCT
0100:  JSR OUTCH
0110:  INY
0120:  BNE LOOP
0130:  CPYIM 02
0140:  JMP $2031
0150: MESSG = 'H
0160:  = 'I
0170: @

-X

PASS 1
ID=00

***********<E2>0040
***********<07>0070
ID=

PASS 2
PRINT? NO

SAVE ID=

ID=00

***********<E2>0040
0040: 0200 00 00 00
***********<07>0070
0070: 0207 00 00 00
***********<A8>0130
0130: 0214 00 00 00
ID=
```

set up the NMI vector

execute from $2000

Respond Y to clear workspace

ADD new data

Input for a program

line 0010 -delete was used to backspace over a typing error

Note spacing of label, instruction, and argument

@ end of data input

execute the assembler to check for syntax errors

ID= 00 --resident source

```
      LDAIN MESSG
      STS   INDCT  +01
```

Address mode!
Instruction!

a carriage return indicated the last tape

not worth printing yet

a carriage return indicated-no save

00 - resident source

```
      LDAIN MESSG
      STS   INDCT  +01
      CPYIM 02
```

Argument!

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 $02
0131: @

-X

PASS 1
ID=00

ID=

PASS 2
PRINT? Y

SAVE ID=

ID=00


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=

-X 200
HI HIt  g- g-%LI n-1 M-H@Ps g- %i%iEp M-% M-): p-L*L1  C'C&TP<) g-)¦T
-M 140,120

-N
```

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.

Listing looks OK.

Execute Program at $0200.

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 sytem, 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. Alterations 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 successful 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,$17F6) to EAL, EAH ($17F7, $17F8) and writes the start address 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 returns 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 adress $29FE. It may be replaced with a return ($0D) or a null ($00).

The number of lines per assembler 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

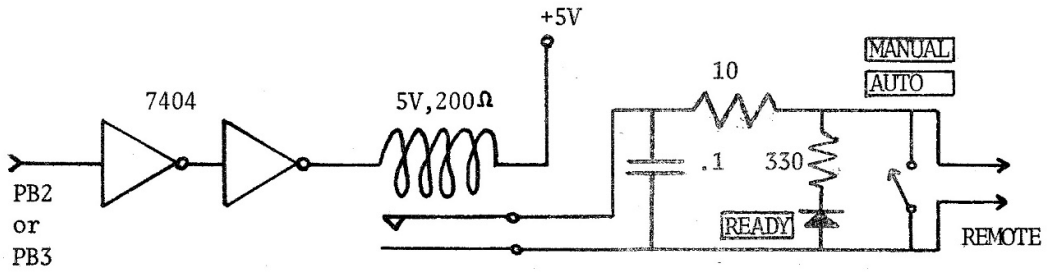
The areas of memory to be used for the various files associated with the Micro-ADE system are allocated by a table at address $2EA3. In this case, $3600 to $3FFF has been allocated as the source, $3000 to $35FF has been allocated as the symbol table, and $0200 upward has been allocated for object code.

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
0190: 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
0380: 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:
0760: 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:
0950:                   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!
0140:
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 SAVE
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:
0330: 2E70 20 E3 2D   OVR    JSR   BACKSP CANCEL THE CTL CHAR (THIS MAY NOT BE
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! BUFFER
0410: 2E80 D0 F1             BNE   OVX     KEEP GOING                    LIMIT
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 INTO
0620:                                                                    HEX
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
0720:
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 HERE
0850:
ID=04

0010:
0020:                   ************************************************
0030:                   ***** KIM CASSETTE READ AND WRITE ROUTINES *****
0040:                   ************************************************
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 HARD
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
0080: 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 # DEPEN
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
0270: 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=
```

Hex Dump of Micro-ADE

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

2000: D8 20 E7 2D A9 FF 85 63 A0 63 20 94 27 D0 22 20
2010: F4 23 A9 0D 20 2F 24 20 28 24 F0 06 A0 00 A9 40
2020: 91 10 20 0D 25 A5 11 CD A5 2E D0 F2 A0 69 20 A0
2030: 27 20 E7 2D A2 FF 9A D8 58 A9 04 8D FE 17 A9 24
2040: 8D FF 17 A9 0C 8D 03 17 8D 02 17 A9 2D 20 F0 2D
2050: 20 41 2E A9 00 A2 06 95 19 CA D0 FB 85 17 BD 00
2060: 01 C9 41 30 03 E8 D0 F6 CA A9 00 85 18 85 19 8D
2070: E9 17 E8 BD 00 01 C9 0D F0 23 C9 20 F0 F4 C9 2C
2080: F0 1B 20 94 2E D0 15 A5 18 A0 04 0A 26 19 88 D0
2090: FA 0D E9 17 85 18 8C E9 17 4C 72 20 00 A4 17 48
20A0: A5 18 99 1A 00 A5 19 99 1D 00 68 E6 17 C9 0D D0
20B0: B8 C6 17 D0 08 A5 1A 85 1B A5 1D 85 1E AD 00 01
20C0: C9 41 D0 2C 20 96 24 20 E6 23 20 E6 23 20 E6 23
20D0: A9 0D A0 00 91 10 20 E7 2D 20 88 24 20 28 24 20
20E0: 76 24 C9 40 F0 05 20 3F 24 90 EE 20 2F 24 F0 07
20F0: C9 4C D0 06 20 67 23 4C 31 20 C9 49 D0 52 20 4F
2100: 24 F8 A5 15 38 E9 10 85 15 A5 16 E9 00 85 16 A9
2110: 01 20 41 24 A5 15 C5 1A D0 01 00 20 88 24 A5 15
2120: 48 A5 16 48 20 B2 24 68 85 16 68 85 15 20 28 24
2130: 20 76 24 C9 40 D0 D8 A9 20 20 2F 24 A9 0D 20 2F
2140: 24 A5 15 85 1A 85 1B A5 16 85 1D 85 1E 4C 71 21
2150: C9 4E D0 19 20 F4 23 20 FB 24 30 18 C9 40 F0 14
2160: C9 0D D0 F3 20 3F 24 20 28 24 4C 57 21 C9 44 D0
2170: 06 20 A0 23 4C 31 20 C9 43 D0 03 4C 00 20 C9 46
2180: D0 09 20 67 23 20 A0 23 4C 1B 21 C9 57 D0 0D 20
2190: 4F 24 A5 11 A6 10 20 EB 2D 4C F4 20 C9 53 D0 5B
21A0: 20 A6 21 4C 31 20 A6 17 E0 02 F0 18 20 96 24 A9
21B0: 00 85 1B AD A4 2E 85 1E A5 10 69 06 85 1C A5 11
21C0: 69 00 85 1F A5 1B 8D F5 17 A5 1C 8D F7 17 A5 1E
21D0: 8D F6 17 A5 1F 8D F8 17 A5 1E A6 1B 20 EB 2D A5
21E0: 1F A6 1C 20 EB 2D C6 62 A5 1A F0 02 85 62 AD F5
21F0: 17 85 60 AD F6 17 85 61 4C 9A 2E C9 4D F0 03 4C
2200: A4 22 A5 1A 85 18 A5 1D 85 19 20 4F 24 D0 15 A5
2210: 1B 85 1A A5 1E 85 1D A5 10 85 12 A5 11 85 13 20
2220: 4F 24 F0 01 00 A5 15 C5 1A D0 28 A5 16 C5 1D D0
2230: 22 20 0D 25 20 0D 25 20 DE 24 8A A8 88 A5 12 85
2240: 10 A5 13 85 11 20 B2 24 A9 00 20 2F 24 20 2F 24
2250: 20 76 24 A6 17 E0 02 D0 19 A9 01 18 F8 65 1A 85
2260: 1A A5 1D 69 00 85 1D D8 A5 1C C5 1A A5 1F E5 1D
2270: B0 18 A2 FB B5 20 95 1F E8 D0 F9 C6 17 D0 08 A5
2280: 1A 85 1B A5 1D 85 1E 4C 71 21 A5 1A 48 A5 1D 48
2290: A5 18 85 1A A5 19 85 1D 20 4F 24 68 85 1D 68 85
22A0: 1A 4C 17 22 C9 58 D0 10 A5 1D 05 1A D0 03 4C F8
22B0: 25 A5 1D 85 1B 6C 1A 00 C9 50 D0 0C A5 63 49 FF
22C0: 85 63 A9 F0 85 64 D0 26 C9 47 D0 11 A5 1A 85 62
22D0: 20 97 2E E6 62 A5 1B C5 62 B0 F5 90 11 C9 45 D0
22E0: 10 20 96 24 A5 11 A6 10 20 EB 2D 20 C5 2D 4C 31
22F0: 20 C9 5A D0 36 A6 17 D0 06 A5 1A 05 1D F0 08 A5
2300: 1A 85 3D A5 1D 85 3E A9 F0 85 18 20 7A 2C A6 17
2310: D0 0D E6 18 30 F5 20 2B 2E C9 0D D0 EA F0 CF A5
2320: 3D C5 1B A5 3E E5 1E 90 E2 B0 C3 C9 54 D0 03 4C
2330: 14 25 C9 52 D0 18 A5 1B 85 27 A5 1A 85 62 20 97
2340: 2E 20 AC 21 E6 1A A5 27 C5 1A B0 EE 90 A0 C9 42
2350: D0 14 A4 1C A6 1B A5 1D 86 1D 85 1B 88 B1 1A 91
2360: 1D 98 D0 F8 F0 B7 00 20 4F 24 A5 1D 05 1A D0 06
2370: A9 99 85 1E 85 1B 20 FB 24 85 16 20 FB 24 85 15
2380: A5 1B C5 15 A5 1E E5 16 90 15 20 C5 2D A9 3A 20
2390: F0 2D 20 EE 2D 20 DE 24 48 20 84 2E 68 10 D7 60
23A0: 20 4F 24 10 33 60 A0 02 C8 B1 10 C9 0D F0 05 C9
23B0: 40 D0 F5 60 84 14 A5 10 85 12 A5 11 85 13 A4 14
23C0: B1 10 A0 00 91 10 20 0D 25 AD A5 2E C5 11 D0 EE
23D0: A5 12 85 10 A5 13 85 11 A0 02 A5 1B D1 10 88 A5
23E0: 1E F1 10 B0 C1 60 A5 10 38 E9 01 85 10 A5 11 E9
23F0: 00 85 11 60 A9 FF 85 10 AD A3 2E 85 11 A9 00 85
```

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

2400:  15 85 16 60 68 68 20 0C 24 4C 31 20 48 20 E7 2D
2410:  A0 0A A9 2A 20 F0 2D 88 10 F8 A9 3C 20 F0 2D 68
2420:  20 CD 2D A9 3E 4C F0 2D A5 16 20 2F 24 A5 15 20
2430:  0D 25 AC A5 2E C4 11 D0 01 00 A0 00 91 10 60 A9
2440:  10 18 F8 65 15 85 15 A5 16 69 00 85 16 D8 60 20
2450:  F4 23 20 FB 24 30 1E C9 0D D0 F7 20 FB 24 85 16
2460:  20 FB 24 85 15 C5 1A A5 16 E5 1D 90 E5 20 E6 23
2470:  20 E6 23 A9 00 60 A2 00 BD 00 01 C9 40 F0 08 20
2480:  2F 24 E8 C9 0D D0 F1 60 20 C5 2D A9 3A 20 F0 2D
2490:  20 EE 2D 4C 41 2E 20 F4 23 20 FB 24 C9 40 D0 01
24A0:  60 C9 0D D0 F4 20 FB 24 85 16 20 FB 24 85 15 4C
24B0:  99 24 C8 C8 C8 84 14 A5 10 85 12 A5 11 85 13 20
24C0:  96 24 A0 00 B1 10 A4 14 91 10 20 E6 23 CD A3 2E
24D0:  D0 01 00 C5 13 D0 EB A5 10 C5 12 D0 E5 60 A2 00
24E0:  20 FB 24 9D 00 01 E8 E0 40 10 08 C9 0D F0 0B C9
24F0:  40 D0 ED A9 0D 9D 00 01 A9 FF 60 20 0D 25 AC A5
2500:  2E C4 11 D0 03 A9 FF 60 A0 00 B1 10 60 E6 10 D0
2510:  02 E6 11 60 A0 4B 20 A0 27 AD A6 2E A2 00 20 EB
2520:  2D A5 42 A6 41 20 C9 2D A6 1A E0 02 F0 73 30 0A
2530:  A5 1B 85 41 A5 1E 85 42 D0 67 20 E6 25 A6 1A D0
2540:  11 A0 00 B1 55 D1 3F 90 16 D0 25 C8 C0 04 D0 F3
2550:  F0 1E A0 04 B1 55 D1 3F C8 B1 55 F1 3F B0 11 A0
2560:  05 B1 3F 48 B1 55 91 3F 68 91 55 88 10 F3 E6 54
2570:  20 C1 27 A2 16 20 C3 27 90 C3 A6 54 D0 BC A9 00
2580:  20 E6 25 E6 54 30 07 20 E7 2D A9 FC 85 54 20 A4
2590:  25 A0 04 B1 3F AA C8 B1 3F 20 C9 2D 20 C1 27 90
25A0:  E2 4C 31 20 A0 04 20 EE 2D 88 D0 FA A0 00 B1 3F
25B0:  99 20 00 C8 C0 04 D0 F6 A0 06 A2 06 A9 00 06 23
25C0:  26 22 26 21 26 20 2A 29 1F CA D0 F2 18 69 40 C9
25D0:  40 D0 02 A9 20 20 F0 2D A2 05 C0 04 D0 02 A2 06
25E0:  88 D0 D9 4C EE 2D A9 00 85 3F 85 54 A9 06 85 55
25F0:  AD A6 2E 85 40 85 56 60 90 00 84 41 AD A6 2E 85
2600:  42 20 A0 27 A9 FF 85 4C A0 5E 20 A0 27 E6 1C A5
2610:  1F C5 1C 90 0D A5 1C 85 62 20 CD 2D 20 87 27 4C
2620:  53 26 20 41 2E AD 00 01 C9 0D D0 03 4C E2 26 AE
2630:  01 01 20 FD 2A D0 18 85 1C 85 62 AD 02 01 C9 2C
2640:  D0 11 AD 03 01 AE 04 01 20 FD 2A 85 1F F0 04 A0
2650:  67 D0 B7 A5 62 F0 03 20 97 2E 20 F4 23 20 0D 25
2660:  A9 FF 85 4A AD A8 2E 38 E9 01 85 4B A2 FF 58 9A
2670:  A9 A4 8D FE 17 A9 28 8D FF 17 20 FB 24 85 16 20
2680:  FB 24 85 15 20 DE 24 30 26 20 D9 27 20 D4 28 85
2690:  47 20 56 29 20 20 27 A6 4C D0 0B 20 8B 29 20 CC
26A0:  2A 20 F9 29 F0 03 20 59 28 20 7B 29 4C 6C 26 A6
26B0:  4C D0 2C A6 51 D0 28 E6 5D A5 5D 85 62 8E F5 17
26C0:  AD A8 2E 8D F6 17 A5 4A 69 01 8D F7 17 A5 4B 69
26D0:  00 8D F8 17 20 9A 2E A5 3D 85 60 A5 3E 85 61 4C
26E0:  08 26 A5 4C F0 37 A9 00 85 4C A9 00 85 55 85 56
26F0:  A0 08 20 94 27 D0 02 A9 00 85 4D A0 18 20 A0 27
2700:  20 41 2E AD 00 01 C9 0D F0 0E AE 01 01 20 FD 2A
2710:  D0 E9 85 5D C6 5D A9 00 85 51 4C 08 26 4C 31 20
2720:  A5 47 C9 0C D0 1D A2 05 B5 20 95 57 CA 10 F9 20
2730:  8B 29 A5 49 85 61 85 3E A5 48 85 60 85 3D A9 FF
2740:  85 46 60 C9 80 F0 F7 C9 FA F0 F3 C9 4C F0 30 C9
2750:  6C F0 2C 29 0C C9 0C D0 26 A5 2B C9 24 D0 0A A5
2760:  2C 05 2D C9 30 F0 10 D0 16 A9 38 85 43 20 37 28
2770:  D0 0D C8 B1 3F D0 08 A5 47 29 F7 85 47 C6 46 60
2780:  A6 4D D0 FB 4C CD 2D A9 0D D0 02 A9 20 A6 4D D0
2790:  EE 4C F0 2D 20 A0 27 20 41 2E AD 00 01 C9 59 60
27A0:  B9 0A 2C C8 C9 40 F0 05 20 91 27 D0 F3 60 B9 00
27B0:  01 C9 0D D0 03 88 A9 20 95 20 E8 C8 C9 20 D0 EE
27C0:  60 A2 00 A9 06 18 75 3F 95 3F B5 40 69 00 95 40
27D0:  B5 3F C5 41 B5 40 E5 42 60 A2 1C A9 20 95 20 CA
27E0:  10 FB A0 00 E8 20 AE 27 A2 06 20 AE 27 E0 0A D0
27F0:  02 95 20 A2 0B 20 AE 27 84 31 A0 00 A2 00 20 1A
```

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

2800:   28 A0 03 A2 02 20 1A 28 A0 06 A2 04 20 1A 28 A0
2810:   0B A2 06 20 1A 28 A0 0E A2 08 B9 20 00 95 32 B9
2820:   22 00 29 1F 85 3C B9 21 00 0A 0A 0A 0A 36 32 2A
2830:   36 32 05 3C 95 33 60 A9 FA 85 3F AD A6 2E 38 E9
2840:   01 85 40 A0 00 B1 3F D1 43 D0 06 C8 C0 04 D0 F5
2850:   60 20 C1 27 90 ED A9 FF 60 A5 20 C9 20 D0 01 60
2860:   A9 32 85 43 A9 00 85 44 20 37 28 D0 01 00 A0 00
2870:   B1 43 91 41 C8 C0 04 D0 F7 A6 47 E0 FA D0 0E 20
2880:   8B 29 A5 48 A0 04 91 41 A5 49 C8 D0 07 A5 3D 91
2890:   41 C8 A5 3E 91 41 A2 02 20 C3 27 A5 42 CD A7 2E
28A0:   D0 01 00 60 68 68 20 0C 24 20 C5 2D A9 00 85 47
28B0:   85 48 85 49 A9 02 85 46 A5 4D 48 A9 00 85 4D A6
28C0:   4C D0 05 20 F9 29 F0 03 20 83 2A 68 85 4D 20 7B
28D0:   29 4C 6C 26 A2 09 A5 29 DD 9B 2B F0 04 CA 10 F6
28E0:   00 A5 2A DD A5 2B D0 F5 86 45 E0 00 D0 04 A0 37
28F0:   D0 02 A0 3F A5 36 D9 07 2B D0 07 A5 37 D9 47 2B
2900:   F0 04 88 10 EF 00 98 C9 35 D0 04 E0 06 F0 13 C9
2910:   3D D0 04 E0 07 F0 0B DD AF 2B 90 05 DD B9 2B 90
2920:   01 00 C0 30 10 05 BD 87 2B D0 03 BD 91 2B 85 3C
2930:   29 07 85 46 98 A2 03 26 3C 2A CA D0 FA 90 0F 26
2940:   3C 2A B0 02 10 0B 26 3C 2A 90 02 29 FD 60 29 F8
2950:   60 29 70 09 8A 60 A6 46 E0 03 30 1E A5 47 C9 20
2960:   F0 14 29 1F C9 10 D0 04 A2 01 D0 0C 29 0F C9 0B
2970:   10 04 A2 00 F0 02 A2 02 86 46 60 A5 46 30 0B 38
2980:   65 3D 85 3D A5 3E 69 00 85 3E 60 A5 2B C9 20 F0
2990:   F9 C9 24 F0 4D C9 27 F0 43 A9 38 85 43 A9 00 85
29A0:   44 20 37 28 F0 01 00 B1 3F 85 48 C8 B1 3F 85 49
29B0:   A4 31 B9 00 01 C9 2F F0 3B C9 2B F0 06 C9 2D D0
29C0:   C9 C6 49 29 02 48 C8 B9 00 01 C8 BE 00 01 20 FD
29D0:   2A 28 F0 03 49 FF 38 A2 09 4C C6 27 A5 2C 85 49
29E0:   D0 12 A5 2C A6 2D 20 FD 2A 85 49 A5 2E A6 2F 20
29F0:   FD 2A F0 02 A5 49 85 48 60 E6 56 30 3C A9 0C 20
2A00:   8D 27 20 87 27 A0 00 B9 57 00 20 8D 27 C8 C0 06
2A10:   D0 F5 A0 23 B9 0A 2C C9 40 F0 06 20 8D 27 C8 D0
2A20:   F3 A5 55 F8 18 69 01 85 55 D8 20 80 27 A0 03 20
2A30:   87 27 88 D0 FA A9 C8 85 56 20 87 27 A5 16 20 80
2A40:   27 A5 15 20 80 27 A9 3A 20 8D 27 20 8B 27 A5 26
2A50:   C9 20 D0 0A A0 0E 20 8B 27 88 10 FA 30 43 A5 3E
2A60:   20 80 27 A5 3D 20 80 27 20 8B 27 A5 47 A6 46 20
2A70:   B1 2A A5 48 A6 46 CA 20 B1 2A A5 49 A6 46 CA CA
2A80:   20 B1 2A 20 8B 27 A0 00 C0 06 F0 04 C0 0B D0 03
2A90:   20 8B 27 B9 20 00 20 8D 27 C8 C0 11 D0 EA 20 8B
2AA0:   27 A4 31 B9 00 01 C9 0D F0 06 20 8D 27 C8 D0 F3
2AB0:   60 30 10 A0 00 E6 4A D0 02 E6 4B 91 4A 20 80 27
2AC0:   4C 8B 27 20 8B 27 20 8B 27 4C 8B 27 A5 47 C9 DA
2AD0:   F0 26 29 1F C9 10 D0 1F A5 48 38 E9 02 48 A5 49
2AE0:   E9 00 85 49 68 38 E5 3D 85 48 A5 49 E5 3E A5 48
2AF0:   B0 03 30 03 00 30 FD 60 A5 49 85 47 60 20 94 2E
2B00:   D0 04 8A 20 94 2E 60 0A 41 0A 0D 2A 41 09 4C 4A
2B10:   41 0A 0D 4A 41 0A 4C 80 10 08 53 30 50 08 0D 0E
2B20:   25 09 0D 0E 25 08 4C 53 53 50 52 10 F4 39 A8 3E
2B30:   05 15 04 4E 30 0D 4C 06 49 32 49 4E 30 10 25 3E
2B40:   09 29 29 4E 30 0E 0E 4B 10 0C 83 72 90 A9 A3 89
2B50:   01 C3 89 93 81 D3 A9 00 B9 63 21 99 39 73 96 19
2B60:   D9 C5 84 18 D8 B1 A4 01 13 38 78 B8 00 F0 00 41
2B70:   C4 F2 83 81 81 B0 43 6C EC 72 F2 98 98 A3 C3 47
2B80:   34 B0 A9 99 99 19 18 49 29 48 6E 89 A9 CA EA A9
2B90:   09 11 31 50 76 91 B1 F2 F2 B1 11 49 5A 41 20 49
2BA0:   5A 41 41 5A 49 4D 20 20 20 59 58 59 58 59 58 14
2BB0:   28 30 00 28 28 28 28 34 28 36 40 34 40 30 3E 30
2BC0:   39 36 30 38 30 28 40 02 43 04 40 06 40 08 40 22
2BD0:   43 44 40 06 40 08 40 02 43 44 40 06 40 08 40 02
2BE0:   43 44 40 06 40 08 00 22 44 44 40 69 44 00 11 22
2BF0:   44 44 40 69 44 87 10 22 44 44 40 06 40 08 10 22
```

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

2C00: 44 44 40 06 40 08 A2 14 56 78 0D 50 41 53 53 20
2C10: 31 40 0D 50 41 53 53 20 32 0D 50 52 49 4E 54 3F
2C20: 20 40 0D 53 41 56 45 20 49 44 3D 20 40 20 20 20
2C30: 20 4D 49 43 52 4F 2D 57 41 52 45 20 41 53 53 45
2C40: 4D 42 4C 45 52 20 36 35 58 58 2D 31 2E 30 20 50
2C50: 41 47 45 20 40 0D 20 20 20 20 53 59 4D 42 4F 4C
2C60: 20 54 41 42 4C 45 20 40 0D 49 44 3D 40 0D 4E 45
2C70: 57 3F 40 0D 43 4C 45 41 52 40 20 E7 2D A5 3E A6
2C80: 3D 20 C9 2D A2 00 A1 3D A8 4A 90 0B 4A B0 17 C9
2C90: 22 F0 13 29 07 09 80 4A AA BD C6 2B B0 04 4A 4A
2CA0: 4A 4A 29 0F D0 04 A9 04 A0 80 AA 85 27 84 47 CA
2CB0: 20 2B 29 20 56 29 98 29 8F AA 98 A0 00 84 2B 4A
2CC0: 26 2B 4A 26 2B 4A 26 2B F0 11 4A A0 20 E0 8A F0
2CD0: 08 4A 46 2B A6 2B BC C3 2B 29 07 84 2B 05 2B 85
2CE0: 2B A0 00 20 EE 2D B1 3D 20 CD 2D A2 01 20 7A 2D
2CF0: C4 46 C8 90 F1 A2 03 C0 04 90 F2 A6 3D A4 3E A9
2D00: 06 85 28 20 88 2D A4 2B B9 07 2B 85 2B 4A 4A 20
2D10: 81 2D B9 47 2B 2A 26 2B 2A 26 2B 2A 26 2B A5 2B
2D20: 20 81 2D B9 47 2B 20 81 2D A6 27 BD 9A 2B 20 F0
2D30: 2D BD A4 2B 20 F0 2D 20 EE 2D A6 46 CA 30 37 D0
2D40: 25 A0 01 A5 47 29 1F C9 10 D0 15 B1 3D 48 38 69
2D50: 01 65 3D AA A5 3E 69 00 A8 68 10 13 88 4C 6F 2D
2D60: B1 3D AA 88 F0 09 A0 01 B1 3D AA C8 B1 3D A8 A9
2D70: 00 85 28 20 88 2D 20 7B 29 60 20 EE 2D CA D0 FA
2D80: 60 29 1F 09 40 4C F0 2D 86 29 84 2A 20 E6 25 A0
2D90: 04 B1 3F C5 29 D0 0A C8 B1 3F C5 2A D0 03 4C AC
2DA0: 25 20 C1 27 90 E9 A4 28 D0 0F A9 24 20 F0 2D A5
2DB0: 2A A6 29 20 C9 2D 20 EE 2D 20 EE 2D 88 10 FA 60
2DC0: 26 FF E3 FF EB A5 16 A6 15 20 CD 2D 8A 48 4A 4A
2DD0: 4A 4A 20 D8 2D 68 29 0F C9 0A 18 30 02 69 07 69
2DE0: 30 D0 0D A9 5F D0 09 A9 0D D0 05 20 C9 2D A9 20
2DF0: 84 F4 86 F5 C9 0D D0 1E A6 63 D0 13 E6 64 D0 0F
2E00: 20 2B 2E C9 1B D0 04 A9 FF 85 63 A2 F0 86 64 A9
2E10: 0A 20 16 2E A9 0D 20 A0 2E 2C 40 17 10 05 A6 F5
2E20: A4 F4 60 2C 40 17 10 FB 4C 31 20 86 F5 84 F4 20
2E30: 9D 2E 29 7F C9 0D D0 07 A9 0A 20 A0 2E A9 0D D0
2E40: D8 A0 00 20 2B 2E C9 7F D0 07 20 E3 2D 88 10 F3
2E50: 00 C9 5C D0 06 20 E7 2D 4C 41 2E C9 05 F0 11 99
2E60: 00 01 C9 0D F0 09 C8 C0 3A 30 D8 A9 0D D0 EC 60
2E70: 20 E3 2D B9 00 01 C9 0D F0 C9 20 F0 2D C8 C0 3A
2E80: D0 F1 F0 BF A0 00 B9 00 01 48 20 F0 2D 68 C8 C9
2E90: 0D D0 F3 60 4C 00 1A 4C AF 2E 4C 35 2F 4C 5A 1E
2EA0: 4C A0 1E 35 36 40 30 36 02 20 CD 2D 20 EE 2D AD
2EB0: 02 17 29 FB 8D 02 17 A9 7F 8D 41 17 D8 A9 8D 8D
2EC0: EC 17 20 32 19 A9 13 8D 42 17 20 41 1A 46 F3 05
2ED0: F3 85 F3 8D 40 17 C9 16 D0 F0 20 24 1A 8D 40 17
2EE0: C9 2A D0 F2 20 F3 19 C5 62 D0 BE 20 F3 19 20 4C
2EF0: 19 8D ED 17 20 F3 19 20 4C 19 8D EE 17 A2 02 20
2F00: 24 1A C9 2F F0 14 20 94 2E D0 BF CA D0 F1 20 4C
2F10: 19 20 EC 17 20 EA 19 4C FD 2E 20 F3 19 CD E7 17
2F20: D0 A8 20 F3 19 CD E8 17 D0 A0 AD 02 17 09 04 8D
2F30: 02 17 4C 8C 1E AD 02 17 29 F7 8D 02 17 A9 AD 8D
2F40: EC 17 20 32 19 A9 27 85 F0 A9 BF 8D 43 17 A2 F0
2F50: A9 16 20 A3 2F A9 2A 20 C6 2F A5 62 20 B2 2F A5
2F60: 60 20 AF 2F A5 61 20 AF 2F 20 EC 17 20 AF 2F 20
2F70: EA 19 AD ED 17 CD F7 17 AD EE 17 ED F8 17 90 E9
2F80: A9 2F 20 C6 2F AD E7 17 20 B2 2F AD E8 17 20 B2
2F90: 2F A2 02 A9 04 20 A3 2F AD 02 17 09 08 8D 02 17
2FA0: 4C 8C 1E 86 F1 48 20 C6 2F 68 C6 F1 D0 F7 60 20
2FB0: 4C 19 48 4A 4A 4A 4A 20 BB 2F 68 29 0F C9 0A 18
2FC0: 30 02 69 07 69 30 A0 08 84 F2 A0 02 84 FE BE FC
2FD0: 2F 48 2C 47 17 10 FB B9 FD 2F 8D 44 17 A5 F0 49
2FE0: 80 8D 42 17 85 F0 CA D0 E9 68 C6 FE F0 05 30 07
2FF0: 4A 90 DB A0 00 F0 D7 C6 F2 D0 CF 60 02 C3 03 7E
```

EDITOR ERROR MESSAGES

1C INSERTION OVERFLOW. An attempt has been made to insert 10 lines in a 9 line space.

26 ATTEMPT TO MOVE BEYOND THE END OF FILE. An illegal line number has been used in the MOVE command.

3B SOURCE FILE LIMIT EXCEEDED. An attempt has been made to store data beyond the allocated source file.

68 COMMAND SYNTAX ERROR. The command entered cannot be recognized.

9E COMMAND PARAMETER SYNTAX ERROR. An illegal character has been used in a command parameter.

D4 ATTEMPT TO MOVE BEGINNING OF FILE. The command executed did not operate properly because of a syntax error in the file. Check the first one or two lines of the file for duplication after this error is flagged.

ASSEMBLER ERROR MESSAGES

07 INSTRUCTION SYNTAX ERROR. The instruction field does not contain a valid instruction or pseudo instruction.

23 ILLEGAL ADDRESS MODE. The address mode used is not valid with this instruction.

6F DUPLICATE SYMBOL. An attempt has been made to redefine a symbol.

A4 SYMBOL TABLE OVERFLOW. Too many symbols have been defined.

A8 UNDEFINED SYMBOL. A symbol which has not been defined has been used as an argument.

E2 ADDRESS MODE SYNTAX ERROR. The address mode field does not contain a valid address mode.

F6 BRANCH OUT OF RANGE. A relative branch has been attempted beyond the legal range.

If Micro-ADE is relocated, the error numbers may change.

Micro-ADE COMMANDS

EDITOR COMMANDS

A	ADD new lines to current source file.
C	CLEAR and format the workspace.
L	LIST all lines at the terminal.
Li	LIST line i at the terminal.
Li,j	LIST lines i through j at the terminal.
Ii	INSERT new lines before line i.
Di	DELETE line i.
Di,j	DELETE lines i through j.
Fi	FIX line i. Print it and prompt for edit.
Mi,j	MOVE line j to immediately before line i.
Mi,j,k	MOVE lines j through k to immediately before line i.
N	NUMBER all lines in increments of 10.
Wi	WHERE. Print the absolute address of line i.
E	END. Print the absolute address and number of the last line

CASSETTE COMMANDS

G x	GET file with ID = x from Cassette 1.
G x,y	GET a group of files with ID = x, x+1, ... ,y.
S	SAVE a source file with the last used ID.
S x	SAVE a source file with ID = x.
S x,a,b	SAVE a data block from address a to b-1 with ID = x.
R x	REPRODUCE a source file with ID = x.
R x,y	REPRODUCE a group of source files with ID = x,x+1,...y.

OPERATING COMMANDS

P	Set or reset PAGE MODE.
X	EXECUTE the assembler.
X	EXECUTE address a.
T	Print the symbol TABLE in alphabetical order.
T1	Print the symbol TABLE in address order.
T2	Print the symbol TABLE start and end addresses.
T3,a	Set the symbol TABLE end address to a.
B a,b	BLOCKMOVE 256 bytes from address a to address b.
B a,b,x	BLOCKMOVE x bytes from address a to address b.
Z a,b	DISASSEMBLE continuously from address a to address b.
Z a	DISASSEMBLE 16 lines from address a.
Z	DISASSEMBLE 16 lines from last address disassembled.

Where a and b are hexadecimal addresses, i,j, and k are decimal line numbers, and x and y are 1 byte hexadecimal constants.