MEI (MERLIN Extended Intelligence) ROM

SOFTWARE DOCUMENTATION

TABLE of CONTENTS

Copyright 1978 by MiniTerm Associates, inc.

MEI (MERLIN Extended Intelligence) ROM SOFTWARE DOCUMENTATION

INTRODUCTION

The MEI ROM is a 2K by 8 mask ROM that plugs into the MERLIN Video Interface board, adding many powerful Monitor and Editor commands, plus a graphic keyboard drawing mode. The MEI Software also contains a number of useful, general purpose graphic subroutines that significantly reduce graphic program development time. These subrouintes also ease interfacing graphics with BASIC and other high level languages.

NOTE: The MEI requires the MBI ROM.

The following is a summary of the MEI commands and subroutines.

Monitor Commands

Command Letter	Function
G	-Enter Graphic Keyboard input mode.
HE	-Extended Execute mode. Sets stack pointer so that programs can end with an 'RET' instruction.
HL	-Hex Locate. Locate and display address of hex string.
HX	-Display/Modify Z-80, IX & IY Registers.
V	-Verify two blocks of memory.
KR	-MCAS Read Block
KW	-MCAS Write Block
KV	-MCAS Verify Block
KE	-MCAS Read and Execute

Edit Commands

Command Letter	Function
B	-Block Insert. Move block (CU-A to CU-B) to before cursor.
K	-Move Cursor to next word.
J	-Delete word.
L	-String Locate/Change.
V	-Delete Block.

Graphic Input Commands

Command Letter	Function
&,',(,)	-Move Graphic Cursor to left, up, right, and down. (Shift 6 to 9)
6,7,8,9	-Move cursor to left, up, right or down and then set, clear or compliment bit.
0	-Compliment cursor bit.
M	-Mark graphic cursor position.
L#	-Draw full, dotted, or dashed line from 'mark' to cursor.
S,C,X	-Select 'Set' (OR), 'Clear', or 'XOR' (Compliment) draw mode.
Z	-Zero (Clear) Screen.
P#	-Select pattern.

D -Display (Set/Clear) pattern to screen.

The last two commands are available only in the Super Dense Graphic mode. All others operate in either the Dense or Super Dense Graphic modes.

Graphic Subroutines

Name	Function
XYTA	-Transform X,Y coordinates to memory address and bit position.
XYTO	-Transform X,Y coordinates to offset address and bit position.
DLINE	-Draw line from x1,y1 to x2,y2.
pATN	-Display selected pattern to memory.
MVCU	-Incremental graphic cursor movement.
DSPY	-Display at graphic cursor (Set/Clear/XOR).

General Extensions

-Accepts lower case characters as hex value input.
-Accepts lower case edit command characters.
-New keyboard input driver keeps cursor on screen without <CR>s, for improved text input and editing.

MONITOR COMMANDS

The following Monitor commands are available when the MEI ROM is installed on the MERLIN board. Most of the commands, like the MBI commands, are one letter Monitor commands. However, some of the commands are two (2) letters in length, plus arguments.

Command: HLocate HEX LOCATE

Format: HLstart,end :data1,...,data-n<CR>
start - memory address where search is to start
end - end address of search area
data1 to data-n - hex data (0<n<=10)

The 'end' address is terminated with a 'space' and the program displays the ":" to prompt for the data. Data ends with a carriage return.

Description: The hex locate command searches the specified memory area for all occurences of the search string: "data-1,...,data-n". The start addresses of all identical data strings are displayed. The hex value 81 can not be used as data! The address listing can be aborted by typing 'ESC'.

For example, to find all occurances of calls to the console input subroutine in the MBI, type:

```
HLc000,c800 :cd 7b c1<CR>      (Note LOWER CASE hex characters!)
C0F4
C191
C3EA     (Displayed)
C693
```

Command: HExecute EXTENDED EXECUTE

Format: HEaddr<CR>

Description: The 'HE' command is the same as the MBI 'E' command except the present stack pointer is used, instead of the 'User Program' stack pointer. The 'User Program' stack pointer is the one displayed with the 'X' command. The advantages of the 'HE' command are that programs can end with an 'RET' instruction. This means that they can be called ('HExecuted), and control returns to the calling program when finished. Also any subroutine call be executed from the Monitor with the 'HE' command. Register data can be initailized with the 'X' command before the call ('HE'), The second advantage is that the user does not have to initialize the stack pointer before execution ('X' command), or at the beginning of his program (LXI SP).

NOTE: Do NOT use the 'HE' command when DEBUGGING programs. Set the stack pointer and use the MBI 'E' command. Otherwise there will be a stack conflict between the MBI Monitor and the User Program.

Command: HX DISPLAY/MOD Z-80 IX,IY

Format: HX

Description: The HX command causes the contents of the Z-80 IX register to be displayed. Enter a hex value to change IX (terminate with 'space' or ','); otherwise enter just a 'space' or ','. The contents of the IY register are then displayed. Enter a hex value (terminate with a CR) to change IY, or just enter a <CR> to terminate the command.

Command: Verify VERIFY

Format: Vstart,stop,ver<C>
start - start address of memory block
stop - stop or ending address
ver - verification starting address

Description: The Verify command compares the contents of two blocks of memory and displays the data and addresses wherever there is a difference. The memory block to compare is designated with the 'start' and 'stop' addresses. The block to compare to is designated by the 'ver' address. Differences are displayed as follows:

```
data  data  address
 hh    HH    HHHH
```

All data and addresses are displyed in hex. The second byte of data corresponds to the data at the address displayed.

NOTE: The output can be ABORTED by typing 'ESC'.

The following four commands are used in conjunction with the MiniTerm Cassette Interface (MCAS) board. These commands should NOT be used if an MCAS unit is not connected to your MERLIN! More detail on these commands is provided in the MCAS Manual.

Command: KWrite Cassette WRITE

Format: KWaddr1,addr2[,rate]<CR>
addr1 - start address of the data block
addr2 - stop or ending address.
rate (optional) - baud rate byte.

Description: The 'KW' command writes a block of data to the MCAS in binary Tarbell format.

NOTE: The 'KW' Command does a CHECKSUM calculation and uses the LAST BYTE of the data block to store the checksum value. Normally data files are rounded up to the next 'XXFF' or 'XX00' address, depending on preference. Therefore the checksum will usually not overwrite valid data. However, always be sure to allow for the

extra checksum byte! The user must also be careful that the last byte is in RAM so it can hold the checksum value. For this reason writing to 'XXFF' is probably safer than to 'XX00' (e.g. KW400,1fff<CR>) .

Command: KRead Cassette READ

Format: KRaddr1,addr2[,rate]<CR>
addr1 - start address of the data block
addr2 - stop or ending address.
rate (optional) - baud rate byte.

Description: The 'KR' command reads a block of data from the MCAS. The data must be in Tarbell format.

NOTE: If a CHECKSUM ERROR is detected an inverted '?' is displayed and program control transfers to the MBI Monitor.

Command: KVerify

Format: KVaddr1,addr2[,rate]<CR>
addr1 - start address of the data block
addr2 - stop or ending address.
rate (optional) - baud rate byte.

Description: The 'KV' command reads a block of data from the MCAS and compares it to the data in memory. If the data does not compare, the read stops and the address at which the error occurred is displayed.

Command: KExecute

Format: KEaddr1,addr2[,rate]<CR>
addr1 - start address of the data block
addr2 - stop or ending address.
rate (optional) - baud rate byte.

Description: The 'KE' command reads a block of data from the MCAS and then jumps to the "start" address.

NOTE: If a CHECKSUM ERROR is detected, the read routine will abort to the Monitor and display an inverted '?'.

Command: Jump to 'Memory Reader' MEMORY READER

Format: Jaddr<CR>

The 'J' command selects a new input mode that gets input from

memory, starting at 'addr', rather than from the keyboard. Control is automatically returned to the keyboard when a 'RUBOUT' or any character with b7 set is encountered.

The power of this reader is increased by its detection of 'Ctrl-E char' as an Edit command. For example, E qE qE qE q in the file would cause the screen to flip back and forth four times.

There are many, many uses for this kind of reader. For example, a friend has a neat game running on his MITs BASIC and you have some other BASIC (such as NorthStar), the only way to get his program into your system is to type it in. However, if your friend can create a Tarbell tape of the program (in LISTING fomat), then you could read the BASIC program into memory. After editing the program with the MERLIN Editor, which is much more efficient that any BASIC line editor, the program can be read from memory by your BASIC using the 'J' command.

On many systems it is often necessary to type several commands to get a particular task done. The sequence to edit and assemble a file, even on a floppy based system is often: Boot in floppy, load in assembler/editor, read the file, select the edit mode, and then you are ready to edit. Those four commands could be stored in ROM and executed by just typing: 'Jaddr<CR>'.

The Memory Reader command ties in particularly well with the graphic, keyboard drawing program ('G' command, later Section). For example, enter the following:

```
I                       Go to INPUT mode.
Edit-A                  Cursor HOME.
U2000,4000,09<CR>       Define Dense drawing area.
E q                     Return (flip screen)
G                       Enter Graphic Drawing mode.
Z                       Clear screen.
Edit-0120<CR>           Move cursor to address 0120
666667777788888999999E aE 'ESC'J120<CR>
                        Draw box, Home cursor, Repeat.
```

The above sequence of commands is now in memory at addr 0100. To execute them, type: 'J100<CR>'. The drawing can be aborted by hitting any key. It is not necessary to end the file with a RUBOUT (ASCII 7fH) since the file ends with a 'jump' to repeat part of the command sequence indefinately (until user hits a key).

f
EDIT FUNCTIONS
****

The following Edit commands are available when the MEI ROM is installed on the MERLIN board. Note that either UPPER or LOWER case command letters can be used for these commands as well as the Edit functions in the MBI!

Text Editing

With the editing commands of the MBI and MEI ROMs, the MERLIN text editor is equivalent, and often better, than many word processing editors. For efficient use of MERLIN for text editing, the following procedure is recommended.

MD000 01<CR>	This selects the extended keyboard driver which provides auto scrolling without <CR>s.
U400,xxFF,a8<CR>	Set up an alternate display area for the text buffer. The start address can be any address above 0400H. The buffer size should not be too much larger than the expected text since a large area slows down input in the INSERT (Edit-X) mode. The 'end' address should be xxff. The original display area (100 to 3ff) will be used as a scratch pad area for Monitor commands and the Edit-L command. The 'a8' mode turns on the <CR>s (white boxes). This makes it easy to determine when you are in the text buffer and when you are in the scratch pad area.
Edit-) (shift 9)	This clears the new text buffer.
I	Enter the text input mode (Monitor command processing is bypassed and is available only through Edit-ESC). Use 'Edit-Q' to flip to the scratch pad area for Monitor and DOS commands

Command: Edit-B BLOCK INSERT

Description: The Edit-B command copies the data block designated by CU-A (Edit-U) to CU-B (Edit-I) to the Cursor location, inserting the text before the cursor. CU-A and CU-B designate the same text after the copy. This is useful for multiple copies, and for block deletes (see next command). CU-A and CU-B do not have to be in the present display area.

The cursor is positioned after the inserted text, and is then

centerd on the display screen.

NOTE: When copying text, care must be taken that data at the end of the defined display area is not lost. If sufficient space is not left between the last text data and EOM (display end of memory), the block insert command can cause text to be pushed off the end and lost. Inserts in the last 255 bytes of the buffer are automatially supprssed.

Command: Edit-V DELETE BLOCK

Description: The Edit-V command deletes the text designated by the slave cursors: CU-A to CU-B. The delete is the same as Edit-W (character delete) and Edit-E (line delete) except the deleted text is from CU-A to CU-B. The command can be used after the Edit-B (Block Insert) command to delete the moved text.

Command: Edit-L LOCATE (SEARCH)/CHANGE

Description: The Edit-L command is a string locate and change command. Before using, the user must first define an alternate display area for his text editing. For example the user could type: "U400 1FFF A0<CR>" to define a text display area from 400 to 1FFF. Text can be entered from the keyboard after typing the 'I' command, or read from a storage media, such as cassette or floppy, into this area.

Then to locate a string, first position the cursor before the text to be searched. [This will often be "HOME" (Edit-A)]. Then type Edit-L. The screen will flip to the alternate (original) display area and wait for the string data to be input. TWO strings must ALWAYS be entered, separated by delimiters. The delimiter is always the FIRST character typed. (It is often convenient to use non-alphanumerics such as / or ^.) There is no limit to the length of either string! Any character (except the delimiter) can be in the string, including <CR> or other control characters. If you are doing just a search, the second string should be a null string, i.e., no characters. After the THIRD delimiter is typed the screen will flip back to the text area, move the cursor to the string, and, if possible, position the string as the center line on the screen. The cursor will now be winking at about twice its normal rate. This is to indicate that you are still in the Edit-L (locate/change) mode. The user must now type one of the following:

'space' - locate the next occurence of the string,
'.' - change 'string1' to 'string2' and end,
',' - change 'string1' to 'string2' and then locate the next occurence of the string,
'/' - change all occurences of 'string1' to 'string2'

ANY OTHER input (such as <CR>) terminates the command.

When string2 is longer than string1, care must be taken that data is not lost at the end of the defined display memory, and that the display does not do a reset/clear. To eliminate this problem (except for text inserts longer than 256 (100H) characters long), inserts can not be made in the last 256 bytes before EOM (end of

Display Memory). If an insert does not work, or if not all of the changes on an insert were made, check the display area size.

The insert inhibit in the last 256 (100H) bytes only occurs if the 'end' address ends with 'FF' (e.g. 1fff). The actual inhibit occurs when the MSB (most significant byte) of the cursor address is the same or greater than the MSB of 'EOM'. If the 'EOM' ends in '00' then there is no insert inhibit. The address of the last useful text data can be obtained by positioning the cursor to the end of text and then typing: Edit-U Edit-H. The address will be dsiplayed. There should generally be at least 256 bytes between the end of useful data and 'EOM'.

NOTE: The '/' (change all) mode can be ABORTED during the change by typing another '/'!

Example: To change all occurences of 'and' to '&', type:

Edit-L:and:&:/ (Colon is the delimiter & displays in reverse)

(NOTE: The ':and:&:' is displayed on the alternate screen. The '/' does not display.)

Always use caution when doing a mass change ('/' response). In the above example, all occurences of 'and' would be changed to '&', including 'sand' to 's&'!

WHEN ENTERNING THE STRING CHARACTERS, the use of ANY Edit character to change the string is valid UNTIL the third delimiter is typed. CARE MUST BE TAKEN NOT TO 'ERASE' AN ENTERED DELIMITER!

Command: Edit-J DELETE WORD

Description: The Edit-K command deletes from the cursor to the first non-alpha AFTER the cursor position.

Command: Edit-K MOVE TO NEXT WORD

Description: The Edit-J command moves the cursor to the first non-alpha following the next word.

GRAPHIC INPUT COMMAND

The Graphic Keyboard Input Mode enables the user to draw figures and patterns on either the Dense or Super Dense Graphic screens. A graphic cursor, which can be a dot, small 3 x 3 box, or user defined pattern, is displayed winking on the screen. There are 17 command characters defined for moving, drawing, and setting the drawing mode. There is also a RAM linkage byte so that the user can add additional commands. The Graphic Input Mode is accessed through the Monitor 'G' command as described below.

Command: Graphics

Description: This command puts the MERLIN into a graphic keyboard input mode for drawing lines and patterns in either Dense or Super Dense graphic mode.

BEFORE typing the command letter 'G', the user must first define a graphic display area and mode with the 'Update' command and return to the text area/mode with the Edit-Q command. The screen will flip to the graphic area and the graphic cursor (small box) will be winking near the center of the screen. Type 'Z' to ZERO (Clear) the screen.

NOTE: The Dense Graphic mode should be selected as '09'. Any Super Dense mode, such as '11' can be used.

GRAPHIC INPUT COMMAND CHARACTERS

Cursor Movement

- & - Move the cursor to the left
- ' - Move the cursor up
- (- Move the cursor to the right
-) - Move the cursor down

Note that these characters are the 'move/set' command characters (below), shifted.

Set/Clear At Cursor

- 6 - Move left, set/clear
- 7 - Move up, set/clear
- 8 - Move right, set/clear
- 9 - Move down, set/clear
- 0 - Set/clear at cursor

Data at the cursor is usually complimented (default state). However, data can be SET (OR'd), CLEARED (compliment, AND'd) or COMPLIMENTED (XOR'd). The following command letters select the set/clear mode.

S,C,X - Select SET, CLEAR, COMPLIMENT Display Mode

The command letter 'S' selects the SET (OR) mode for both cursor and pattern operations. This means that the bit at the graphic cursor is set to '1' when the user enters one of the 'Move-Set/Clear' command characters (6,7,8,9,0). Also, the winking pattern, if the user has selected a pattern other than the 'dot', will be 'Set' into the display memory when he types the command letter 'D' (described below).

The command letter 'C' selects the CLEAR mode (load zeros) for both cursor and pattern operations.

The command letter 'X' selects the COMPLIMENT (XOR) mode for both cursor and pattern operations.

With the above simple command set it is possible to move and draw patterns and designs of any size and shape. Notice that the 'Move/Draw' command characters are easily located under the fingers of the right hand, and the shift key can be operated with the left.

Z - ZERO (CLEAR) Screen Command

Typing the letter 'Z' fills the screen area with zeros, clearing the screen. Only an UPPER CASE Z will be accepted. In all other commands, either upper or lower case command letters are accepted.

L - LINE COMMAND

Format: L#<CR>

To aid in drawing lines the 'L' command is available. The 'L' command is used in conjunction with the 'M' (Mark) command. Typing 'M' "marks" the present cursor position. The marked position will wink along with the cursor. Typing 'L<CR>' causes a line to be drawn from the "marked" position to the new cursor location. It is possible to draw dashed or dotted lines by following the 'L' with two digits and a CR as follows:

```
00 - full line (same as just CR
55 - dotted line (every other one)
77 - dotted line (1 every 4)
7f - dotted (1 every eight)
33 - dashed line (2 on, 2 off)
0f - dashed (4 on, 4 off)
11 - dashed (3 on,1 off)
```

The user should experiment with the above and with other combinations.

When in the 'XOR' mode a line can be erased by simply repeating the command. If you were in the 'Set' mode you will have to switch to the 'Clear' or 'XOR' mode before repeating the 'L' command in order to erase the line.

M - Mark Position for 'Start-of-Line'

Typing 'M' marks the present cursor location. The dot at the cursor winks along with the normal graphic cursor. The 'L' command can be used to draw a line from the 'marked' position to the cursor. After the 'L' command the marked position does not change, but will no longer wink.

PATTERN COMMANDS

The pattern placement routines are ONLY available in the SUPER DENSE Graphic mode, although a RAM linkage vector and Dense/Super Dense mode decoding allows Dense patten placement routines to be added by the user.

The user can define an UNLIMITED number of patterns and each pattern can be of ANY size. Also, the patterns can be placed anywhere in memory and need not be contiguous. A linked list structure for the pattern storage makes this possible.

P# - Select Pattern

Format: P#<CR>

The 'P' command selects the desired pattern. The selected pattern is displayed winking at the cursor location. Patterns '0' and '1' are predefined. Pattern 0 is the default and is a 3 x 3 box . Pattern 1 is just a single dot. Patterns 2 and up are USER DEFINED. Pattern 2 starts at RAM address 003E. The pattern table format is as follows:

```
Address of next Pattern  (2 bytes)
Horizontal (byte) width  (1 byte)
Vertical height          (1 byte)
...   data   ...         (H x V bytes)
```

When creating pattern tables, note that bit b0 is displayed first.

The following table displays a small heart as may be used in cards.

```
003E  4A 00    (Address of NEXT pattern)
0040  40 01 07 xx xx xx xx xx
0048  xx xx
```

NOTE: This is pattern #2 (starts a 003E). Pattern #3 starts at 004A.

D - Display Pattern

The 'D' command displays the selected pattern. The selected pattern is the one winking. The pattern is either 'Set', 'Cleared' or 'Complimented' to the screen memory depending on the selected

display mode. When in the compliment mode, two successive D's will first 'Set' and then 'clear' the pattern. This is important to remember because you can not tell that the pattern is permanently 'Set' into the memory until the cursor is moved, or changed to another pattern.

Large patterns can take a little time to wink and may therefore limit the cursor movement rate. (The selected pattern is also the cursor marker.) The user should select pattern '0' or '1' (Command 'P<CR>' or 'P1<CR>') before moving if the movement rate is too slow.

CREATION OF PATTERNS

Probably the easiset way to create patterns is to use graph paper. The finer the grid, the better. Patterns up to 24 by 24 usually work the best although any size is possible. For example, say you wanted the characters in cards so you could link these to your Blackjack program for a more exciting display. You would need the following patterns:A,2,3,4,5,6,7,8,9,10,J,Q,K,Diamond, Heart, Spade, and Club.

First decide on the required size. Usually the smallest block that will yeild sufficient resolution is the best. A pattern size of 8 by 7 was selected for the above card example. Then sketch the patterns or symbols needed by filling in the squares. The following shows the card patterns in the above example. 'Read' the pattern (remember that b0 is displayed first), and fill in the table as pairs of hex digits.

Below is listed the actual pattern tables generated for the above patterns. Pattern #2 is a null (not used) pattern and serves only as a link to our patterns at address 1000H.

pattern no.	Symbol	Address	Data (addr, H, V, 7 data bytes)
2		003e	00 10 01 01 00
3	A	1000	0a 10 01 07 08 14 22 22 3e 22 22
4	2	100a	14 10 01 07 1c 22 20 18 04 02 3e
5	3	1014	1e 10 01 07 3e 20 10 18 20 22 1c
6	4	101e	28 10 01 07 10 18 14 12 3e 10 10
7	5	1028	32 10 01 07 3e 02 1e 20 20 22 1c
8	6	1032	3c 10 01 07 38 04 02 1e 22 22 1c
9	7	103c	46 10 01 07 3e 20 10 08 04 04 04
10	8	1046	50 10 01 07 1c 22 22 1c 22 22 1c
11	9	105a	64 10 01 07 1c 22 22 3c 20 20 1e
12	10	1064	6e 10 01 07 00 72 8a 8a 8a 8a 77
13	J	1078	82 10 01 07 3e 08 08 08 08 0a 0e
14	Q	1082	8c 10 01 07 1c 22 41 41 49 32 3c
15	K	108c	96 10 01 07 22 12 0a 06 0a 12 22
16	Diamond	1096	a0 10 01 07 08 1c 3e 7f 3e 1c 08
17	Heart	10a0	aa 10 01 07 22 77 7f 7f 3e 1c 08
18	Spade	10aa	b4 10 01 07 08 1c 3e 7f 7f 6b 08
19	Club	10b4	be 10 01 07 1c 1c 08 6b 7f 6b 08

The 'G' command initializes the 'Graphic Start Address' to the Graphic Display area 'HOME'. In some cases the user may want the 'Start' address to be different, e.g. when using mixed mode. This can be accomplished by setting HL (with the 'X' command) to the desired address and then typing: 'Exxxx'<CR>. This enters the 'G' command after the address initializatio.

The keyboard drawing mode does NOT check for X,Y limits (boundaries). This is not a problem when moving left or right, although it wraps from the left edge (X=0) to somewhere in the middle (X=255) when moving off the left edge. When moving off the top or bottom the user must be careful not to draw into program code.

RAM Patches to IMSAI 8K BASIC

The I/O sections of the IMSAI 8K BASIC have to be changed to call the MERLIN MBI I/O subroutines for proper interface to MERLIN. The following changes are all that are required.

Address	Old Code	New Code
00A6	D8 03	CD E8
A8	E6 02	C1 00
18F5	DB 03	CD 7B
7	E6 02	C1 C3
9	CA F5 18	FF 18 00
1928	DB 03	C5 4F
2A	IF	CD
2B	D2 28 19	A6 C1 C1
2E	F1	F1
2F	03 02	00 00
1937	3E 0A	C3 44
39	CD 27 19	19 00 00
1A 0E	D8 03	CD E8
10	E6 02	C1 F8
12	C8	C8
13	DB 02	CD 7B
15	E6 7F	C1 00

INTERFACING MERLIN TO TDL 8K or 12K BASIC

TDL software is designed to access all I/O through a jump table to the ZAP or ZAPPLE Monitor. If you have one of these Monitors and are planning on using it, the I/O patches should be made in the Monitor. That way the Monitor, BASIC and all other TDL software will automatically be interfaced to the MERLIN I/O routines. Interfacing Notes IN-202 (ZAP, Console I/O), IN-204 (ZAP Commands), IN-206 (ZAPPLE, Console I/O) and IN-208 (TDL System Monitor Board) should be consulted for these patches.

If you do not have one of the TDL Monitor packages, the TDL BASIC can be patched directly. The patches required for TDL BASIC only, and an explanation of each, is given below. Load BASIC from whatever storage media you are using. We recommend that if you have a relocatable copy of BASIC, that you load it at 0600H. The MERLIN display area can then be assigned to 0200H (8K version) or 0300H (12K version) to 05DFH, leaving room for the BASIC patches at 05E0H to 05FFH. If your copy of BASIC is absolute and resides at 0200H or some other address, the MERLIN display area can be assigned to the top of your memory.

Console Input

The MBI Console Input routine (CDI) is the same as TDL's CI routine, except that data is obtained from the MERLIN keyboard port, and 'Edit' inputs are checked for and processed. Therefore all that needs to be changed for the CI routine is the CI address vector at BASIC relative location 0009'. Use the MERLIN Monitor 'M' command to make the following changes:

	from	to
0009'	C3 03 F0	C3 7B C1

Note that 0009' is TDL relocatable notation. Be sure and add your relocation offset to this address. Also note that the memory address is already in reverse byte format.

Console Output

The MBI Console Output routine (CDO) requires two small changes to be compatible with TDL. First, the MERLIN display does not use line feeds or rubouts, and second, BASIC wants the data that is passed in the C register, returned in the A register. A small interface routine (12 bytes) between BASIC and the MBI CDO subroutine is required. We recommend putting it at location 05E0 if BASIC is at 0600, or at 0040 if BASIC is at 0200. If you can put the patch routines in EROM that is even better. They can be put anywhere you have room.

The CO interface patch is listed below. Note TDL relocating address format.

```
0000'  79          MOV A,C
0001'  FE 0A       CPI 0AH   ; LINE FEED?
0003'  C8          RZ        ; SKIP
0004'  FE FF       CPI 0FF   ; RUBOUT
0006'  C8          RZ        ; SKIP
0007'  CD A6 C1    CALL CDO  ; MBI (reverse byte address)
000A'  79          MOV A,C
000B'  C9          RET
```

The BASIC vector jump address to the CO routine is located at 000F' and must be modified to jump to the above patch.

For example if the above routine was assembled at 05E0, then 000F' would be changed to: C3 E0 05 with the 'M' command.

List Output

If you do not have a TTY or other hard copy device the LO (List Output) should be patched to the CO (Console Output). The jump address for LO is at 0015'. Modify data at this address to the same as 000F' above. If you have a hard copy device, this address should be to your hard copy output driver.

Console Status

BASIC makes use of 'CSTS' (Console Status) to determine if a key has been pressed. An MBI equivalent is not available so a small patch routine is required. We recommend putting it

after the CDO patch.

```
000C'  3A 03 D8   LDA STAT ;READ STATUS
000F'  E6 01      ANI 01   ;MASK READY BIT
0011'  3E 00      MVI A,0
0013'  C0         RNZ
0014'  2F         CMA
0015'  C9         RET
```

The BASIC vector address is at 0018' and must be modified to the address used above (000C' + offset).

IOCHK and IOSET

These should return zero and can be patched as follows:

```
001B'  3E 00 C9
001E'  3E 00 C9
```

Note: 001B' is relative to BASIC, not the patch area used above!

MEMSIZE

This routine is used by BASIC to find the upper end of RAM space. A search routine similar to the one in ZAP or ZAPPLE can be loaded into RAM or EROM, or you can load the following simple routine that returns a set value. Since your system memory size will probably not often change, this is quite adequate. This patch can be assembled right after the status patch routine. Addresses below are relative to patch area, not BASIC.

```
0016'  3E zz      MVI A,addrH
0017'  06 yy      MVI B,addrL
```

where zzyy is the highest RAM address in your system. For example, if you have 16K of RAM memory starting at 0000, the last RAM address is 3FFF; zz would be 3F and yy would be FF. In fact yy will almost always be FF.

If the MERLIN display area was assigned to the top of your memory, this should be taken into account. For example, again assume 16K of RAM and you've assigned MERLIN 1K at the top (3C00 to 3FFF). Then zz becomes 3B.

The BASIC jump vector to MEMSIZ is at 0021' and must be patched to the above address plus its offset.

Trap

TDL BASIC has a trap or error address to go to if BASIC blows up. This should be patched to the MBI BKPT routine. To do this modify 0024' to: C3 2F C6.

Stack

TDL 8K BASIC may not assign enough initial stack area. To eliminate any initialization problems, the following patch should be made:

	from	to
1E80'	02	D1

Summary

This completes the BASIC patches for use with MERLIN. A summary of the patch routines and BASIC patches is listed below (it assumes BASIC loads at 0600 and the patch routines are at 05E0). Standard reverse byte address notation is used; not TDL format.

patch area:

```
05E0  79             CO Patch
  E1  FE 0A
  E3  C8
  E4  FE FF
  E6  C8
  E7  CD A6 C1
  EA  79
  EB  C9

05EC  3A 03 D8       CSTS
  EF  E6 01
  F1  3E 00
  F3  C0
  F4  2F
  F5  C9

05F6  3E addrH       Top of RAM
  F8  06 FF
  FA  C9
```

BASIC

```
0609   C3 7B C1        CI
060c   C3               Reader
060F   C3 E0 05         CO
0612   C3               Punch
0615   C3 E0 05         LO (CO)
0618   C3 EC 05         CSTS
061B   3E 00 C9         I/O SET
061E   3E 00 C9         I/O SET
0621   C3 F6 05         MEMCK
0624   C3 2F C6         TRAP
```

8K Version only

```
2480   D1               STACK
```

Modification Procedure

The following steps summarize the modification procedure:

1) Initialize MERLIN by executing C000.
2) Update the MERLIN display area (U300 5DF A0 if BASIC is at 0600).
3) Load BASIC, preferably at 0600 if relocatable.
4) Make the patches listed above, plus any of your own.
5) Save a copy of the modified BASIC and the patches.
6) Execute BASIC (E600, if BASIC was loaded at 0600)

On subsequent usage, be sure and Update the MERLIN display before loading BASIC.

Usage

Using BASIC with the MERLIN display is the same as with any I/O device with a few exceptions. One is graphics, for which Interface Note IN-210 should be consulted. The other is editing and access to the MERLIN Monitor commands from BASIC. All edit commands, cursor movement, etc., are processed by the MBI software and are ignored by BASIC. MBI Monitor commands, such as the FILL (F) command are accessable by typing Edit-'ESC' followed by one Monitor command. Control returns to BASIC after the command is completed. Multiple Edit-'ESC' may be typed for multiple MBI Monitor commands.

INTERFACING MERLIN TO IMSAI 8K BASIC, Ver 1.4

****

The IMSAI 8K BASIC is set up for TTY Terminal input and output. To be used with other I/O devices, such as MERLIN, the BASIC I/O routines must be changed. The patches to interface IMSAI BASIC to MERLIN are given below. The Terminal input, Terminal output, and Terminal status routines are changed to calls to the MERLIN MBI routines. The MERLIN input patch also converts lower to upper case for those who have lower case keyboards. The initialization section has been changed to read the upper memory address from the keyboard. IMSAI BASIC assigns all contiguous memory, from zero on up, to itself. In many systems this would leave no room for the MERLIN display memory. By defining an upper limit fo BASIC below your actual end of RAM memory, the top few "K" can be assigned to MERLIN and to assembly subroutines.

Initialization

Listing Pages 3 & 4

```
0000  31 00 D1          LXI SP,0D100H ;Temporary stack
0003  C3 81 00          JMP INIT1

0081  21 90 00        INIT1:LXI H,MSG1
0084  CD 6F C1          CALL MSG ;MBI Message routine
0087  0E 01             MVI C,1
0089  CD 74 C2          CALL RHV ;MBI Hex Read sub
008C  E1                POP H
008D  C3 9F 00          JMP INIT3
                      MSG1:
0090  4D 45 4D 20 53    DS "MEM SIZE (H)? "
0095  49 5A 45 20 28
009A  48 29 3F 20 FF    DB 0FFH
```

Terminal (Console) Input

Listing Page 80

```
191D  CD 7B C1      TREAD:CALL CDI ;MBI read char sub
1920  FE 60            CPI 60H
1922  DA 2C 19          JC T2
1925  FE 7B            CPI 7BH
1927  D2 2C 19          JNC T2
192A  D6 20             SUI 20H
192C  77            T2:MOV M,A
```

Terminal (Console) Output — Listing page 80

```
194F  C5           TESTO:PUSH B ;Save
1950  4F             MOV C,A
1951  CD A6 C1       CALL CDO ;MBI Output sub
1954  C1             POP B
1955  C9             RET

Supress Line Feeds

195F  C3 70 19       JMP 1970H
```

Terminal (Console) Status — Listing Pages 83 & 84

```
1A3A  CD E8 C1     TSTCC:CALL KSTAT ;MBI Status sub
1A3D  F8             RM ;Edit
1A3E  C8             RZ ;No Data
w13F  C3 43 1A       JMP GETCH
```

Note: During output to the display, the Control-C may have to be typed several times before it is recognized. This is because the display output subroutine also polls the keyboard and discards any inputs except "Space" and "ESC".

Before loading your copy of BASIC, update the display area to the top of your RAM memory. For example if you have 16K of memory, type: 'U3C00,3FFF,A0■'. Then load BASIC, make the above patches, and save the new version of BASIC for later use.

Type: 'E0■' to execute BASIC. In response to "MEM SIZE (H)?", enter the upper limit of RAM for BASIC use. Assuming 16K of RAM, and MERLIN display area from 3C00 to 3FFF, enter: '3C00'. In response to ':', type: 'NEW' (the first command to BASIC must be: NEW).

When loading the revised version remember to update the display area to the top of RAM first.

INTERFACING MERLIN TO Altair 8800 BASIC, Version 4.0

All 4.0 versions of Altair BASIC have the same I/O structure, and require the same I/O patches to run with MERLIN. The addresses at which the patches are made may be different for the different BASICs, but the MERLIN I/O patches are the same. Location 0039H of 4.0 BASIC contains the address of the I/O vector table. This table tells you where to put the patches. The I/O patches for MERLIN are listed below with general address format. The 8K BASIC is used as an example later. Refer to the January 1977 Altair 88000 BASIC User Manual, pages 131 to 134 (Appendix L) for more detail on the BASIC I/O structure.

NOTE: before loading, patching and/or running BASIC, the MERLIN display area must be redefined since the default area of 100 to 3FF conflicts with BASIC. The easiest place to assign MERLIN is to the top of your RAM memory. For example, if you have 16K, update MERLIN to 3C00 to 3FFF mode A0 ("U3C00,3FFF,A0"). Refer to the MERLN manual if you are unfamiliar with the "Update" command.

Load, patch and save the new copy of BASIC. You are now ready to run BASIC. Type: E0<CR> to execute BASIC. When answering the first question ("MEMORY SIZE") don't forget to subtract space for the MERLIN display area, and any space needed for assembly language routines. For example, if you have 16K of RAM and MERLIN is assigned from 3C00 to 3FFF, memory size can be up to 15320.

```
0039    IOLST           I/O address table


IOLST   TRYIN           Console Input
        TRYOUT          Console Output
        ISCNTL          Console Status, CTRL-C?
        NEWSTT              "


TRYIN:  CD 7B C1        CALL CDI ;MERLIN Input
        FE 60           CPI 60H  ;Lower Case?
        DA __ __        JC S1
        D6 20           SUI 20H  ;Convert to Upper Case
        00          S1:NOP


TRYOUT: F1              POP PSW  ;Data
        C5              PUSH B
```


```
          F5              PUSH PSW ;Save B,C & SW
          E6 7F           ANI 7FH  ;Strip  off b7
          4F              MOV C,A
          CD A6 C1        CALL CDO ;MERLIN Output
          F1              POP PSW
          C1              POP B    ;Recover B,C & SW
          F5              PUSH PSW
          F1              POP PSW  ;Restore SW  on stack
          C9              RET

ISCNTC:   CD E8 C1        CALL KSTAT;Keyboard status
          00              NOP
          C8              RZ        ;No data
          00 00 00        NOP,NOP,NOP

NEWSTT:   CD E8 C1        CALL KSTAT;Keyboard status
          00              NOP
          C4              CNZ CNTCNN
```

8K BASIC EXAMPLE

```
0039  00 02           (0200H)

0200  45 05           TRYIN  (0545H)
0202  36 05           TRYOUT (0536H)
0204  57 06           ISCNTC (0536H)
0206  FB 05           NEWSTT (06FBH)

0536  F1 C5
0538  F5 E6 7F 4F CD A6 C1 F1
0540  C1 F5 F1 C9 00 CD 7B C1
0548  FE 60 DA 4F 05 D6 20 00

05FB  CD E8 C1 00 C4

0657  CD
0658  E8 C1 00 C8 00 00 00
```

The I/O address at 0039H is not used by BASIC, nor is RST 7. Therefore, the RST 7 at address 0038H should be patched to jump to the MERLIN breakpoint routine:

```
0038  C3 2F C6
```

MERLIN does not need Line Feeds. The Line Feed code appears as: 3E 0A DF and should be changed to three NOPs. In the 8K BASIC this code is at address 0885H. Also, the Line Feed output with "OK" can be eliminated by changing 0266 from 0A to 00 (8K BASIC).

```
         F5              PUSH PSW ;Save B,C & SW
         E6 7F           ANI 7FH  ;Strip  off b7
         4F              MOV C,A
         CD A6 C1        CALL CDO ;MERLIN Output
         F1              POP PSW
         C1              POP B    ;Recover B,C & SW
         F5              PUSH PSW
         F1              POP PSW  ;Restore SW  on stack
         C9              RET

ISCNTC:  CD E8 C1        CALL KSTAT;Keyboard status
         00              NOP
         C8              RZ        ;No data
         00 00 00        NOP,NOP,NOP

NEWSTT:  CD E8 C1        CALL KSTAT;Keyboard status
         00              NOP
         C4              CNZ CNTCNN
```

8K BASIC EXAMPLE

```
0039  00 02             (0200H)

0200  45 05             TRYIN  (0545H)
0202  36 05             TRYOUT (0536H)
0204  57 06             ISCNTC (0536H)
0206  FB 05             NEWSTT (06FBH)

0536  F1 C5
0538  F5 E6 7F 4F CD A6 C1 F1
0540  C1 F5 F1 C9 00 CD 7B C1
0548  FE 60 DA 4F 05 D6 20 00

05FB  CD E8 C1 00 C4

0657  CD
0658  E8 C1 00 C8 00 00 00
```

The I/O address at 0039H is not used by BASIC, nor is RST 7. Therefore, the RST 7 at address 0038H should be patched to jump to the MERLIN breakpoint routine:

```
0038  C3 2F C6
```

MERLIN does not need Line Feeds. The Line Feed code appears as: 3E 0A DF and should be changed to three NOPs. In the 8K BASIC this code is at address 0885H. Also, the Line Feed output with "OK" can be eliminated by changing 0266 from 0A to 00 (8K BASIC).