MicroWorks Inc.

DOCUMENTATION

MicroWorks Inc.
P. O. Box 44248
Cincinnati, OH 45244

MicroWorks Microware Documentation

Program: STEPPER v3.00
Program Author: James Shrider
Documentation author: James Shrider
Date of last revision: 07/10/80

MicroWorks Inc.
P. O. Box 44248
Cincinnati, OH 45244

Table of Contents

Table of Contents

CHAPTER 1

INTRODUCTION

STEPPER (Ver. 3.0) is a software machine code debugger which serves much the same purpose as a hardware front panel, but in a more comprehensive manner. At all times, the screen displays all CPU registers and flags, the top sixteen bytes of the stack, a sixty-four byte window of memory centered on the location indicated by the Program Counter register, the Z-80 mnemonic representation of the instruction indicated by the Program Counter, and several other non-Z-80 registers. In STEPPER, options exist for

*Examining or modifying any register, flag, or memory byte
*Block moving any sized block of memory to any location
*Block filling any sized block of memory with any single byte
*Zeroing any sized block of memory
*Searching memory for a specific block of data
*Single-step executing any instruction
*Setting and removing a breakpoint
*Setting and removing a trace stop condition
*Executing code from any point
*Setting input/output mode to ASCII, octal, or hexadecimal
*Relocating any section of code or table of addresses
*Relocating STEPPER anywhere in memory
*Disassembling any section of code at variable speeds
*Tracing program execution at variable speeds, with or without the display
*Dumping disassembly/trace information to a hardcopy device

1.1 Manual Conventions

In this documentation, the prefix "CTL" means that the following letter is to be entered while holding down the Control key. Thus, a "CTL A" would be entered by holding down the Control key and then typing an "A".

All constants given in the examples used in this manual are hexadecimal.

The term "cursor" refers to the asterisk on the screen; this indicates where a value will be deposited if typed.

All input prompts will appear below the memory window display (on the bottom line).

1.2 Making Corrections

If an error is made while typing in a numeric value, typing "Q" will cause the current process to be aborted.

CHAPTER 2

OPERATION

STEPPER commands are all initiated by typing a single character. If additional information is required, the program will ask for it. Commands are broken down into four basic groups: Cursor Movement, Input/Output, Macro, and Executive.

2.1 Cursor Movement Commands

Cursor movement commands are commands which cause the Program Counter to be altered.

2.1.1 Set High Order Byte (H)

This command will set the high order byte of the Program Counter to a specified value and then move the cursor to the resulting Program Counter location.

Example:	H	(Enter command)
	Address?	(Prompt)
	06	(Enter high address)

Will set the high order byte of the Program Counter equal to 06. The low order byte will not be affected.

2.1.2 Set Low Order Byte (L)

This command is used to set the low order byte of the Program Counter to a specified value and move the cursor to the resulting Program Counter location.

Example:	L	(Enter command)
	Address?	(Prompt)
	4E	(Enter low address)

Will set the low order byte of the Program Counter equal to 4E. The high order byte will not be affected.

2.2.3 Set Program Counter (J)

Used to set both high and low order bytes of the Program Counter to a specified value and move the cursor to the resulting location.

```
Example:  J              (Enter command)
          Address?       (Prompt)
          064E           (Enter address)
```

Will set the Program Counter equal to 064E.

2.1.4 Move Cursor Right (Space or CTL L)

This will move the cursor right and increment the Program Counter if the cursor points to the memory window; otherwise, if the cursor points to the registers or flags, this will move the cursor right to the next register or flag.

```
Example:  CTL L          (Enter command)
```

Will move the cursor right one byte.

2.1.5 Move Cursor Left (CTL H)

Will move the cursor left one byte and decrement the Program Counter if the cursor points to the memory window; otherwise, if the cursor points to the registers or flags, will move the cursor left to the next register or flag.

```
Example:  CTL H          (Enter command)
```

Will move the cursor left one byte.

2.1.6 Move Cursor Down (CTL J)

This command will move the cursor down one line (8 bytes) and change the Program Counter accordingly if the cursor points to the window of memory; otherwise, if the cursor points to the registers or flags, this command will be ignored.

```
Example:  CTL J          (Enter command)
```

Will move the cursor down one line.

2.1.7 Move Cursor Up (CTL K)

This will move the cursor up one line (8 bytes) and change the Program Counter accordingly if the cursor points to the window of memory; otherwise, if the cursor points to the registers or flags, this will be ignored.

```
Example:  CTL K          (Enter command)
```

Will move the cursor up one line.

2.1.8 Tab Cursor Down (CTL D)

This command tabs the cursor down four lines (32 bytes) and changes the Program Counter accordingly if the cursor points to the window of memory; otherwise, if the cursor points to the registers or flags, this command is ignored.

Example: CTL D (Enter command)

Will tab the cursor down four lines.

2.1.9 Tab Cursor Up (CTL U)

This will tab the cursor up four lines (32 bytes) and change the Program Counter accordingly if the cursor points to the window of memory; otherwise, if the cursor points to the registers or flags, this will be ignored.

Example: CTL U (Enter command)

Will tab the cursor up four lines.

2.1.10 Move Cursor to Register Bank (R)

Moves the cursor to the bank of registers and flags.

Example: R (Enter command)

Will cause the cursor to point to the Accumulator.

2.1.11 Move Cursor to Memory Window (P)

This command is used to move the cursor to the window of memory.

Example: P (Enter command)

Will cause the cursor to point to the byte indicated by the Program Counter.

2.1.12 Block Search Memory (CTL S)

This command is used to position the Program Counter at the first byte of the next occurrence of the search string following the current position of the Program Counter. The

search string is taken as the string which is pointed to by the "Search" register. The first byte of the search string must be the length (01-FF) of the search string. The second through last bytes are the search string itself. Note that the search string must be loaded into memory and the "Search" pointer set to it before this command can be used.

```
Example:  J              (Move PC to search string
          Address?       location and enter search string)
          2000
          04             (Length of search string)
          54             ("T", MSB low)
          45             ("E")
          53             ("S")
          54             ("T")
          J              (Move cursor somewhere else)
          Address?
          1000
          CTL S          (Search memory starting at 1000
                         for "TEST")
```

Will result in the Program Counter having a value of 2001.

Note that the Search String need only be entered once; CTL S may then be used as many times as needed.

2.2 Input/Output Commands

Input/output commands are commands which affect input or output modes in some way.

2.2.1 Shift Input/Output Mode (RUB or DEL)

This command is used to shift the current input/output mode to ASCII, octal, or hex.

```
Example:  RUB            (Enter command)
```

Will switch to ASCII if the current mode is hex, octal if the current mode is ASCII, or hex if the current mode is octal.

2.2.2 Turn Printer On/Off (CTL P)

This command may be used to turn the printer on or off. If the system has no printer driver, this command will have no effect.

```
Example:  CTL P          (Enter command)
```

Will turn the printer on if it is off, or off if it is on (see

chapter 4 for more information about Printer Routine Interfacing).

2.3 Macro Commands

Macro commands are commands which act on blocks of memory.

2.3.1 Block Move Memory (M)

This command may be used to copy any size block of memory from one location to another.

```
Example:  M              (Enter command)
          Old start?     (Prompt)
          1610           (Enter start of code to be moved)
          Old end?       (Prompt)
          161F           (Enter end of code to be moved)
          New start?     (Prompt)
          1700           (Enter new start of code)
```

Will move the block of memory starting with (1610) and ending with (161F) to (1700). Blocks may be moved up or down in memory, and may overlap without fear of data propagation.

2.3.2 Block Memory Fill (CTL F)

This command is used to fill any size block of memory with a single byte of data.

```
Example:  CTL F          (Enter command)
          Data?          (Prompt)
          04             (Enter data byte)
          Start?         (Prompt)
          1540           (Enter first loc to be filled)
          End?           (Prompt)
          16FF           (Enter last loc to be filled)
```

Will change all bytes in memory from (1540) through (16FF) inclusive to 04's.

2.3.3 Block Zero Memory (Z)

This command will fill any sized block of memory with 00's.

```
Example:  Z              (Enter command)
          Start?         (Prompt)
          1540           (Enter first loc to be filled)
          End?           (Prompt)
```

```
16FF            (Enter last loc to be filled)
```

Will change all bytes in memory from (1540) through (16FF) inclusive to 00's.

2.3.4 Relocate Code (N)

This command can be used to relocate a section of code so that any memory references are offset by a specified amount. This command will not move the code; the Block Move command (see section 2.3.1) should be used for that. The code to be relocated must not contain data areas; relocation will be aborted if an illegal instruction is encountered. The following instructions will be relocated if encountered (standrd Zilog mnemonics):

```
LD   A,(nn)
LD   (nn),A
LD   HL,(nn)
LD   dd,(nn)
LD   IX,(nn)
LD   IY,(nn)
LD   (nn),HL
LD   (nn),dd
LD   (nn),IX
LD   (nn),IY
JP   nn
JP   cc,nn
CALL nn
CALL cc,nn
```

Essentially, the command works like this: The instruction indicated by the Program Counter is is disassmbled. If one of the above instructions is found, "Offset" is added to "nn". An illegal instruction ("ERR" mnemonic) will abort the process; otherwise, it will continue until the Program Counter reaches a specified ending address.

Load immediate instructions, such as LD HL,nn, will NOT be relocated. If there are load immediate instructions which reference hard addresses, these should be changed to load from a table in memory. The Relocate Table command can then be used to relocate the table (see section 2.3.5). For instance, the following will not be correctly relocated:

```
1000        LD   HL,TSMSG    Point to message.
1010        CALL MESSAG      Display message.
  .
  .
  .
8000 TSMSG  DC   "This is a test.",0
```

It should be changed to:

```
1000          LD   HL,(VTSMSG)  Point to message.
1010          CALL MESSAG       Display message.
  .
  .
  .
8000 TSMSG    DC   "This is a test.",0
9000 VTSMSG   DW   TSMSG
```

The code could then be relocated in two steps: first the actual code, and then the address table.

```
Example:  J              (Place cursor at beginning of
          Address?       code to be relocated)
          1000
          N              (Enter command)
          Offset?        (Prompt)
          0100           (Enter offset value)
          End?           (Prompt)
          10FF           (Enter ending address)
```

Would relocate the code from (1000) to (10FF) inclusive to run at (1100) to (11FF). The address of the Program Counter would be displayed in the upper left corner of the screen.

2.3.5 Relocate Table (CTL N)

This command is used to relocate a table of sixteen bit addresses. The table may also contain data which is not to be relocated, as long as each table entry is the same number of bytes in length. Note that this command does not actually move the table; it merely offsets the addresses contained in it.

```
Example:  J              (Move cursor to first address in
          Address?       table to be relocated)
          2000
          CTL N          (Enter command)
          Step?          (Prompt)
          0001           (Enter # of bytes between addrs)
          Offset?        (Prompt)
          FF00           (Enter offset value)
          End?           (Prompt)
          2006           (Enter ending address)
```

Assume that the following table existed at (2000):

```
2000:  00 10             (First address in table)
2002:  D4                (Data byte - not an address)
2003:  34 12             (Second address)
```

```
2005:  C7                  (Data byte)
```

After the above example, the table would look like this:

```
2000:  00 0F               (First address relocated)
2002:  D4                  (Data byte - unaffected)
2003:  34 11               (Second address relocated)
2005:  C7                  (Data byte)
```

At the end of the relocation, the Program Counter points to the last byte which was relocated.

2.3.6 Relocate STEPPER Code (U)

This will move STEPPER to a specified new starting page, relocate its code and tables, and jump to the relocated version. Because the code is relocated AFTER it is moved, the original version will remain intact in memory.

```
Example:  U                (Enter command)
          New starting page?  (Prompt)
          21               (Enter new starting page)
```

Will first clear the screen and display the message "Relocating STEPPER." Next, will move STEPPER to (2100) and relocate the code to run there. Finally, will jump to (2100).

Because of the method of relocation, STEPPER must not be moved within its own boundries or catastrophic error may occur. Thus, it must be moved at least as many pages as its own length. To move a shorter distance, two moves must be made.

```
Example:  U                (Enter command)
          New starting page?  (Prompt)
          30               (Move STEPPER clear of itself)
          U                (Enter command)
          New starting page?  (Prompt)
          04               (Enter actual desired location)
```

This will safely move STEPPER from page 01 to page 04.

2.3.7 Disassemble Code (CTL A)

This command may be used to disassemble any section of code. This will really only be useful if the output is sent to a printer.

```
Example:  CTL A            (Enter Command)
          End?             (Prompt)
          1000             (Enter ending address)
```

This will disassemble the code beginning at the location indicated by the Program Counter and ending at the specified location.

The following input keys are active while the disassembly is in progress:

0	Freeze
1-9	Speed (1 is slowest, 9 fastest)
Q	Abort process

2.4 Executive Commands

Executive commands are commands which are used to control the execution of machine code.

2.4.1 Trace Start (T)

This is used to execute the machine code beginning at the location indicated by the Program Counter. The execution is not free running; STEPPER retains control at all times. Normally, all registers and flags are displayed, so the user can see exactly what the processor is doing at all times. (The display may be turned off by typing "F" during trace. To turn the display back on, type "S" during trace. Turning off the display allows you to see information written to the display by the program being traced. It also causes the code to be executed at least 20 times as fast as when the display is turned on). Because STEPPER retains control, it is possible to set a Trace Stop Condition for any register, flag, or memory location (see section 2.4.3). Trace information will also be sent to the printer, if it is turned on.

```
Example:  T                    (Enter command)
          Hit any key to begin.  (Prompt)
          9
```

This cause the system to initiate a trace at speed nine. The trace will stop if a HALT (76) or Breakpoint is encountered, or if the Trace Stop Condition (if any) is met.

The Reason for hitting a key to begin execution is that it allows setting an initial speed (see below). Note that this will not work on some types of keyboards.

The following input keys are active while the trace is in progress:

0	Freeze

1-9	Speed (1 is slowest, 9 fastest)
Q	Abort process
F	Turn off display (Fast Trace)
S	Turn on display (Slow Trace)

2.4.2 Set Trace Stop Condition (O)

This command is used to define the Trace Stop Condition. To use it, move the cursor to the register, flag, or memory location that you wish to use for the condition test. Next, enter the value that you wish to test for (see chapter 3 for information on data entry). Then, back the cursor up so that it again points to the item. Enter the Set Trace Stop Condition command (O). This will cause the "Cond" register to show the condition value. Finally, replace the value with the one that was originally there.

Now, whenever the selected register, flag, or memory location contains the specified value, a trace will be aborted.

```
Example:  R          (Move cursor to Accumulator)
          04         (Enter condition value)
          CTL H      (Move cursor left)
          O          (Set Trace Stop Condition)
          00         (Restore original value in Acc)
```

Will set a Trace Stop Condition of 04 on the Accumulator. Now, assume that the following routine were in memory:

```
2000:  3C            (Increment Accumulator)
2001:  18 FD         (Jump Relative to 2000)
```

Now, furthur assume that the routine were executed using trace after the above condition had been set.

```
Example:  J          (Move Program Counter to 2000)
          Address?
          2000
          T          (Initiate a slow trace)
```

The Accumulator would be incremented until it contained a value of 04, at which point the trace would be aborted.

2.4.3 Erase Trase Stop Condition (X)

This command simply erases any existing Trace Stop Condition. It is used when you wish to trace without stopping for any condition.

```
Example:  X          (Enter command)
```

Would remove any existing Trace Stop Condition and place "none" in the "Cond" register.

2.4.4 Quit STEPPER

This command is used to exit STEPPER and enter the resident operating system.

Example: Q (Enter command)

Would cause a jump to location 0000.

2.4.5 Single-Step (S)

This command may be used to single-step execute one machine instruction.

Example: S (Enter command)

Would execute the instruction indicated by the Program Counter and update the display accordingly.

2.4.6 Execute Code (G)

This command is used to execute code beginning with the instruction indicated by the program counter. This is a free running execution; it will continue at full speed until the system encounters a Breakpoint (if any has been set).

Example: G (Enter command)

Would restore all registers and flags to the values indicated on the screen, clear the screen, and execute the code starting with the instruction indicated by the Program Counter.

2.4.7 Set Breakpoint

This command sets a Breakpoint at the location indicated by the Program Counter. Any existing Breakpoint will be cleared. Breakpoints may not be set in ROM or in non-existant memory; any such attempt will be ignored.

This command works by placing a Restart instruction at the specified location whenever a "G" command is executed; re-entering STEPPER causes the Restart instruction to be removed and the original data restored.

Because a Restart instruction is used, the Restart vector must jump to STEPPER. Starting with page zero, STEPPER checks to see if it can place a JP instruction which vectors back to itself. If not (because of ROM), it checks to see if there is a JP to another location; if so, it chases the chain until it finds RAM where it can place a JP instruction back to itself. If STEPPER can not complete this chain, the Breakpoint will not be set.

Example: CTL B (Enter command)

Will set the Breakpoint at the Program Counter and remove any existing Breakpoint. The location of the Breakpoint will be placed in the "Break" register. When the Breakpoint is encountered by the CPU, control will be vectored to STEPPER, which will then display the status of the registers, flags, and stack at the time execution was halted.

Note: The Restart instruction used as a Breakpoint may be changed if the vector is used by the system for some other purpose. To do this, change locations (0152) and (0154) so that they contain the desired Restart instruction. These addresses assume a starting STEPPER page of 01. Valid Restart instructions are 00, 08, 10, 18, 20, 28, 30, and 38.

2.4.8 Erase Breakpoint (CTL E)

This command will remove any existing breakpoint and place "none" in the "Break" register.

Example: CTL E (Enter command)

Will remove the Breakpoint if one exists.

Note: The Set Breakpoint and Erase Breakpoint commands are most useful when used in conjunction with the Execute Code command (see section 2.4.7).

CHAPTER 3

DATA ENTRY

To enter data into a register, flag, or memory location, simply move the cursor to the desired location and type in the data. The cursor will be moved right to the next item when entry of a data item is complete. Any character which is not valid for the item or the Input/Output mode will be ignored.

Octal values are split octal; thus, the value 777 will not be accepted, while the value 377 will.

When in the ASCII mode, all commands except "RUB", "CTL L", "CTL H", "CTL J", "CTL K", "CTL D", and "CTL U" are disabled. Any other characters will be entered directly into the memory location or register indicated by the cursor. Characters will be entered with the most significant bit set to the value indicated by the MSB status register.

Example: 2F (Enter data byte)

Would enter 2F into the location indicated by the cursor.

To alter data on the stack, the "J" command should be used to set the Program Counter equal to the stack area. Data on the stack can then be modified by changing memory with the method described above.

CHAPTER 4

SPECIAL CONSIDERATIONS

The user stack is displayed with the two bytes of each item reversed; this is to compensate for the fact that the CPU handles all sixteen bit data values with the two bytes reversed.

The stack pointer displayed must not point anywhere within STEPPER's code. If it does, executing any Executive Command will destroy STEPPER.

CHAPTER 5

PRINTER ROUTINE INTERFACING

STEPPER 3.0, while capable of sending output to a printer, does not contain any printer drivers. These must be installed by the user. They should be located immediately after STEPPER in memory. The following instructions assume that STEPPER is located to run at an address of 0100.

(013A) is a Jump instruction which vectors to the last instruction of STEPPER. This is currently a Return instruction. Your printer driver should begin at this location. The character to be printed will be passed in the Accumulator. No registers may be altered.

Once the driver is in place, you must set the sixteen bit value in (0102) to reflect the added length. Do this by taking the value from (0102), adding to it the length (in bytes) of your printer routine, and saving the new value.

The next step is to let the relocator know that your driver exists. At (0104), there begins a table which is used by the "Relocate STEPPER" command to determine what sections of code and address tables to relocate. Each entry in the table is five bytes in length, and has the following format:

Byte	Function
1	Control byte; 00 = End of table 01 = Relocate Code 02-FF = Relocate Table, where Step is n-2
2-3	Starting address (Offset from 0000)
4-5	Ending address (Offset from 0000)

The first three five-byte entries in the table are used to relocate STEPPER code and tables; the fourth is used to relocate the Input/Output subroutines. The next three five-byte entries are not used. These may be used to relocate your printer routine. The last entry need only be one byte long; it must be a 00.

Note that the starting and ending addresses are zero-relative; this is the address that the items would be at if STEPPER had a starting address of 0000.

APPENDIX A

INPUT/OUTPUT ROUTINES

STEPPER uses the following "external" routines to handle Input/Output. Each routine is called via a vector located in the first page of STEPPER. Vectors given below assume that STEPPER starts at 0100. Routines may only alter the Accumulator; no other register may return altered. Return to STEPPER is via a Return instruction (C9).

A.1 HOME Routine

The vector to this routine is at (0128). The routine must home the cursor. The screen may or may not be cleared in the process, but a cleaner display will result if it is not cleared.

A.2 CRT Routine

The vector to this routine is at (012B). The routine should display the character presented in the Accumulator. No control characters will be sent to this routine.

A.3 ERASE Routine

The vector to this routine is at (012E). This routine should erase the screen and home the cursor.

A.4 CRLF Routine

The vector to this routine is at (0131). This routine should place the cursor at the beginning of the next line of the display by writing spaces.

A.5 INP Routine

The vector to this routine is at (0134). This routine should simply poll the keyboard and place the result in the Accumulator. It should not wait for any special condition (such as strobe) before returning.

A.6 KEY Routine

The vector to this routine is at (0137). This routine should wait until a key is pressed. It should then save the data in the Accumulator and wait for the key to be released. When the key is released, the routine should return.

A.7 PTR Routine

The vector to this routine is at (013A). This routine should return immediately if there is no system printer; otherwise, it should send the character in the Accumulator to the printer and return.

APPENDIX B

COMMAND SUMMARY

Cursor Movement Commands:

H	Set High Order Byte
L	Set Low Order Byte
J	Program Counter
CTL L	Move Cursor Right
SPACE	Move Cursor Right
CTL H	Move Cursor Left
CTL J	Move Cursor Down
CTL K	Move Cursor Up
CTL D	Tab Cursor Down
CTL U	Tab Cursor Up
R	Move Cursor to Register Bank
P	Move Cursor to Memory Window
CTL S	Block Search Memory

Input/Output Commands:

RUB	Shift Input/Output Mode
CTL P	Turn Printer On/Off

Macro Commands:

M	Block Move Memory
CTL F	Block Memory Fill
Z	Block Zero Memory
N	Relocate Code
CTL N	Relocate Table
U	Relocate STEPPER Code
CTL A	Disassemble Code

Executive Commands:

T	Trace Start
O	Set Trace Stop Condition
X	Erase Trace Stop Condition
Q	Quit STEPPER
S	Single-Step
G	Execute Code
CTL B	Set Breakpoint
CTL E	Erase Breakpoint