Identification

Product Code:	MFE-8E-MON8-M
Product Name:	Memon-8 System Monitor with TD8-E Functions
Date Created:	October 3, 2020
Maintainer:	Martin Eberhard
Author:	Martin Eberhard

Revision History

Manual Version	Memon-8 Version	Date	Author	Comments
MFE-8E-MON8-A	1.00-1.03	12 Apr 2020	M. Eberhard	Rough manual created, rough code created
MFE-8E-MON8-B	1.00-1.03	12 Apr 2020	M. Eberhard	Improved manual, debug code
MFE-8E-MON8-C	1.04	19 Apr 2020	M. Eberhard	Add TB command to boot from DECtape. Rewind tape after XR, XS, and XV commands. (Thanks Volker for the suggestions!)
MFE-8E-MON8-D	1.05-1.07	23 Apr 2020	M. Eberhard	Take RIM & BIN loader snapshot, add RL command to restore them.
MFE-8E-MON8-E	1.08-1.10	3 May 2020	M. Eberhard	Add RR and RR9 commands
MFE-8E-MON8-F	1.11 & 1.12	12 May 2020	M. Eberhard	Add RRC TRC, TWC, TVC commands
MFE-8E-MON8-G	1.13	2 Sep 2020	M. Eberhard	Use field 6 as buffer if running in field 7. (Allows 7 full fields for OS/8, without overwriting Memon-8)
MFE-8E-MON8-H	1.14	9 Sep 2020	M. Eberhard	And TC and TN commands
MFE-8E-MON8-I	1.15	11 Sep 2020	M. Eberhard	Verify (in reverse) during XR command, for a big speedup of the tape write & verify (formerly XR then XV) process. Adjust parameters for RR commands so that the provided block number is the lowest block number read. (Start with block bbbb+nnn-1)
MFE-8E-MON8-J	1.15	11 Sep 2020	M. Eberhard	Manual correction for reverse-read commands
MFE-8E-MON8-K	1.16	18 Sep 2020	M. Eberhard	Add BF command to change the buffer field. Add marginal tape block counter. Small bug fix. Improve manual readability a lot.
MFE-8E-MON8-L	1.17	19 Sep 2020	M. Eberhard	Treat lost block as a non-fatal error in XS command.
MFE-8E-MON8-M	1.18	3 Oct 2020	M. Eberhard	Default TZ command to TZ R (to rewind the tape). Write full 12-bit values for forward and reverse checksums into the buffer for TRC command.

1. ABSTRACT

Memon-8 is a general purpose stand-alone system monitor intended to be used as a hardware and software debugging tool on a PDP-8e family computer. It includes functions for examining and modifying core, and for filling, copying, and verifying copies of core blocks. It also includes functions for reading, writing, and verifying portions of a DECtape on a TD8-E subsystem. Memon-8 can read, write, and verify PDP-8 "TU56" files that are loaded through a serial port using the Xmodem protocol. These files can be as long as an entire DECtape. Memon-8 has a command to restore the standard paper tape loaders to core, and can boot from DECtape, similar to DIGITAL's TD8-E boot loader.

2. REQUIREMENTS

2.1 Equipment

PDP-8e, a Teletype or other serial console device on the same IOT device as the RIM loader. A second serial device (KL8-E or equivalent) is helpful (but not required) for sending and receiving "TU56" files via Xmodem.

2.2 Storage

Memon-8 requires 8k words of core (3K if the DECtape functions are not used). Memon-8 occupies core from 0002 through A bit less than 6000 in any memory field. The entire subsequent memory field is used as the default buffer for the DECtape and Xmodem commands. However, If Memon-8 is running in field 7 then its default buffer will be in field 6. (You can change the buffer field with the BF command.)

The binary and RIM loaders should be in the same field as Memon-8 the first time Memon-8 is run. The RIM loader must be the standard DIGITAL RIM loader, though it may be modified for a different IOT code.

3. LOADING PROCEDURE

Load Memon-8 into any field using the standard binary loader.

4. OPERATION

4.1 Starting and Restarting Procedure

Make sure the correct field is selected with the EXT ADDR switch. Then set the switch register to 0200 and press LOAD ADDR. Now press CLEAR and then CONT.

Memon-8 can be restarted any time by this procedure.

The first time Memon-8 is run after being loaded, it will examine the RIM loader in core (in the same field) to determine the console port's IOT code. If it finds a RIM loader, it will use that RIM loader's IOT code for the console. Otherwise it will use the standard Teletype console IOT code, 603X and 604X.

When Memon-8 is restarted, it will not look for the RIM loader again, but will use the same console IOT code as the last time it was run.

4.2 Operating Procedure

Memon-8 prints its sign-on banner and prompt on the console, which is the same device the RIM loader uses for loading. (If you wish to use another IOT device as the console, you can use the CP command to change the console port.) The sign-on banner looks like this when running in field 0:

```
Memon-8   1.18
NY M.  EBERHARD

BUFFER AT 10000

>
```

Commands are typed at the “>” prompt, followed by the “return” key. You can edit your typing using the “backspace” or “delete” key.

All numerical values in Memon-8 are expressed in octal.

4.2.1 Console Commands

`?` Print a help screen

CB 0/1 **Console Backspace**: controls how deleted characters are displayed. If the console is a Teletype or other printing terminal, then type “CB 0”, which will cause deleted characters to be displayed between back-slashes. If the console is a video terminal, such as a vt52, then type “CB 1”, which will cause deleted characters to be erased on the screen. (CB 1 is the default when Memon-8 first runs.)

CP 0nn0 **Change Console Port**: allows you to change the console IOT code. The standard PDP-8 Teletype console uses IOT codes 603X and 604X. To select this device, you would type “CP 0030”. The standard IOT code for a second KL8-E serial interface is 630X and 631X. To select this device, you would type “CP 0300”. (You can always eliminate leading zeros, and type “CP 300”.)

RL [n] **Restore Loaders**: restores the BIN and RIM loaders to their correct locations in field N. if n is unspecified, then it defaults to field 0. If a standard RIM loader (which may have been modified for a different IOT code) was found the first time that Memon-8 was run then Memon-8 took a snapshot of the BIN and RIM loaders for restoration by this command. If no RIM loader was found, then this command will restore the unmodified standard DIGITAL BIN and RIM loaders.

4.2.2 Core Commands

For all core commands, aaaaa and ddddd are core addresses. The memory field number is the first digit of a 5-digit value. Leading zeros are assumed for values with fewer than 5 digits, implying field 0.

For all core commands, ccccc is a count value. The maximum value for ccccc is 10000. 0 is interpreted as 10000. If you do not enter a value for ccccc, then it defaults to 1.

CO aaaaa ddddd [ccccc] **Copy**: copies ccccc words of core from address aaaaa to address ddddd.

DU aaaaa [ccccc] **Dump**: displays core contents on screen in octal. You can pause and restart the display with any key on the keyboard. Control-C aborts the DU command and returns to the prompt.

EN aaaaa **Enter**: displays the contents of address aaaaa, and allows you to enter a new value. If you type an octal value and then hit return, the typed value will be written into core at the address. If you just hit return, the original contents will be unchanged. The next address and its contents will then be displayed, allowing you to enter a new value there. This will continue until you type Control-C.

EX aaaaa **Execute**: execution begins at address aaaaa. AC = 0 on arrival at the execution address.

FI aaaaa vvvv [ccccc] **Fill**: fill memory with octal value vvvv, which defaults to 0000.

VE aaaaa ddddd [ccccc] **Verify**: ccccc words of memory starting at address aaaaa are compared to memory starting at address ddddd. Differences are reported on the console.

4.2.3 Transfer Port Setup Commands

The “transfer port” is a second KL8-E (or equivalent) serial port, used for transferring files to and from the PDP-8e.

PT **Transfer Port Test**: characters typed on the console keyboard are sent to the transfer port. Characters received by the transfer port are displayed on the console. Exit this mode by typing Control-C.

The parity bit is stripped from all characters.

Use this command to verify the connection, baud rate, and functionality of the transfer port.

XP [0nn0] **Change Transfer Port**: allows you to change the transfer port IOT code. The standard IOT code for a second KL8-E serial interface is 630X and 631x. To select this device, you would type “XP 300”.

If no value is specified, then the current transfer port's IOT code will displayed on the console.

You can set the transfer port to be the same as the console. If you do, then error messages will be suppressed and Control-C will not abort the command. (You can abort an Xmodem transfer by stopping the transfer on the PC side, and then waiting for Memon-8's Xmodem routine to time out.)

4.2.4 TD8-E DECtape Commands, No Buffer

TU [0/1] **Tape Unit**: set the tape unit for all future DECtape operations. The "SELECT" light on the specified TU56 tape unit will light when this command is issued, if that unit is set for "REMOTE" on the TU56.

If no unit is specified, then the currently selected unit will be displayed on the console.

TB **Tape Boot**: boot from tape, the same as DIGITAL's TD8-E boot loader. Tape block 0 is loaded at address 07360, tape block 1 is loaded at address 07600, and execution transfers to address 07400 with the tape still moving in the forward direction.

Note that if Memon-8 is running in field 0, then booting OS/8 will overwrite Memon-8. This works fine, but Memon-8 will be gone. (This is probably true for booting other programs too.)

TN **Report Next Tape Block Number**: If you repeat the TN command, there will typically be 5 or 6 block numbers between each TN, due to the overshoot as the tape stops moving, and the time to takes to get the tape back up to speed.

TS bbbb **Tape Seek**: seek the specified DECtape block.

TZ [F/R] **Tape End-Zone Seek**: seek forward or reverse to the DECtape end zone. If no parameter is given, TZ defaults to reverse, for rewinding as tape.

4.2.5 TD8-E DECtape Commands that Require the Buffer

The following DECtape commands use an entire memory field as a buffer. Memon-8's buffer field is the next-higher field above the field in which Memon-8 is running. The exception is if Memon-8 is running in field 7. In this case, the buffer will be in field 6. The buffer's memory field is displayed after the banner when Memon-8 is started.

Memon-8 assumes standard PDP-8 DECtape blocks, each with 129 words of data. Many of the DECtape commands allow you to specify 128-word blocks or 129-word blocks. The 128-word commands will write 0 to the 129th word, and will ignore the 129th data word when reading. (However, the 129th word is always included in checksum calculations.)

For all DECtape commands, bbbb specifies a block number in octal. The maximum block number is 2701.

For 128-word DECtape commands, the maximum value for the block count nnn is 100. For 129-word DECtape commands and the checksum commands,

the maximum value is 77. If no value is specified for nnn, then nnn defaults to one block. If bbbb+nnn-1 is greater than 2701, then the command will abort with the usual bad command error message.

All commands that use the buffer start with buffer address x0000.

BF [n] **Set Buffer Field**: Sets the buffer field to field n. If n is not specified, then display the current buffer field. Setting the buffer field to be the same as the field in which Memon-8 is running causes an error to be reported on the console.

4.2.5.1 Normal Transfer commands

TC [bbbb [nnnn]] **Tape Check**: read nnnn blocks from the tape starting at block bbbb, checking the format and the block checksums, reporting errors on the console. BBBB defaults to 0, and NNNN defaults to the rest of the tape. This command overwrites everything in the buffer.

When the tape check finishes, the total number of blocks with irrecoverable non-fatal errors (permanent lost block or checksum errors), as well as the number of marginal blocks (where a block was found or a correct checksum was achieved after more than one read), are reported on the console.

TR bbbb [nnn] **Tape Read, 128-word blocks**: read nnn blocks from DECtape, starting with block bbbb. Write the data into the buffer.

TR9 bbbb [nn] **Tape Read, 129-word blocks**: read nn blocks from DECtape into the buffer, starting with block bbbb.

TV bbbb [nnn] **Tape Verify, 128-word blocks**: compare nnn blocks from DECtape to buffer data. Display mismatches and tape errors on the console.

TV9 bbbb [nn] **Tape Verify, 129-word blocks**: compare nnn blocks from DECtape to buffer data. Display mismatches and tape errors on the console.

TW bbbb [nnn] **Tape Write, 128-word blocks**: write nnn blocks from the buffer to DECtape, starting with block bbbb.

TW9 bbbb [nn] **Tape Write, 129-word blocks**: write nnn blocks from the buffer to DECtape, starting with block bbbb.

4.2.5.2 Reverse Transfer Commands

Sometimes on a marginal tape, a block that has errors when read forward is readable without errors when read backward.

The reverse read commands read the tape backwards and then unscramble the data in the buffer, correcting both the word order and the data within each word. (When read backwards, every 12-bit data word is

scrambled: ABCD becomes !(DCBA), where A, B, C, and D are 3-bit nibbles, and "!" means 1's complement.)

The tape blocks are read starting with block bbbb+nnn-1, and continue downward from there to block bbbb. When the command completes, the data in the buffer is in the same order as with a forward read: the lowest numbered block is first in the buffer. For example, RR 17 3 would read blocks 19, 18, and then 17 in that order. Block 17 would be in the buffer starting at address X0000, block 18 would be in the buffer at X0200, and block 19 would be in the buffer at X0400.

RR bbbb [nnn] **Reverse Tape Read, 128-word blocks**: reverse-read nnn blocks from DECtape, starting with block bbbb+nnn-1. Write the data into the buffer starting at its beginning.

RR9 bbbb [nn] **Reverse Tape Read, 129-word blocks**: reverse-read nn blocks from DECtape, starting with block bbbb+nn-1. Write the data into the buffer starting at its beginning.

4.2.5.3 Data & Checksum Transfer Commands

In addition to the 129 block data words, these commands also write the forward and reverse checksums into the buffer, or read them from the buffer and write them to tape. For only the last of the nn blocks for the two read commands, the forward checksum is followed by the checksum that was computed from the data. For example, if you issued the command:

```
TRC 5 3
```

The buffer would contain the following:

```
10000  block 5 reverse checksum, should be 77XX
10001  block 5 1st data word
 ...
10201  block 5 129th data word
10202  block 5 forward checksum, should be 00XX
10203  block 6 reverse checksum, should be 77XX
10204  block 6 1st data word
 ...
10404  block 6 129th data word
10405  block 6 forward checksum, should be 00XX
10406  block 7 reverse checksum, should be 77XX
10407  block 7 1st data word
 ...
10607  block 7 129th data word
10608  block 7 forward checksum, should be 00XX
10609  block 7 computed checksum, 00XX
```

Note that the reverse checksum word is normally 7777. Regardless, bits 0:5 are not included in the computation of the checksum calculation.

Bits 6:11 of the forward checksum are the computed checksum (exclusive-or) of all of the 6-bit data nibbles of the block, including bits 6:11 of the reverse checksum word. Bits 0:5 of the forward checksum word are usually 00, and are not included in the checksum calculation.

TRC bbbb [nn] **Tape Read, 129-word blocks with Checksums**: read nn blocks from DECtape, starting with block bbbb. Write the data and checksums into the buffer.

RRC bbbb [nn] **Reverse-tape read, 129-word blocks with Checksums**: reverse-read nn blocks from DECtape, starting with block bbbb. Write the data and checksums into the buffer. Unscramble the data as with the RR9 command.

TWC BBBB [NN] **Write Data and Checksums**: for each block, the reverse checksum, 129 data words, and the forward checksum are written to tape from the buffer. Note that you can write incorrect checksums with this command.

4.2.6 Xmodem TD8-E DECtape Commands

These commands send and receive files via the transfer port, using the Xmodem protocol (with checksums) for error checking and flow control.

You can use any Xmodem-capable program on a PC (e.g. TeraTerm) for sending and receiving TU56 files. Be sure to specify checksum error checking (not CRC error checking).

Using the Xmodem protocol, these commands send and receive “PDP-8 TU56” files, which encode 12-bit PDP-8 words as two 8-bit bytes. For each pair of bytes, the first byte <7:0> has PDP-8 bits 4-11 and the second byte has PDP-8 bits 0-3 in its bit positions <3:0>. (Note the reversed bit order naming convention DIGITAL and typical 8-bit bytes.) All 129 words of the DECtape blocks are included in a TU56 file.

Xmodem blocks are fixed at 128 8-bit data bytes per block, with no indication of the actual file length. To accommodate this restriction, the last Xmodem block sent (with the XS command) is padded as needed at the end with zeros. When receiving an Xmodem file (with the XR or XV command), any partial tape block at the end will be discarded quietly.

The tape is rewound at the end of these commands.

When an XS or XV command finishes, the total number of blocks with irrecoverable non-fatal errors (lost block, permanent checksum, or verify errors), as well as the number of marginal blocks (where a block was found or a correct checksum was achieved after more than one read), will be reported after the Xmodem transfer completes.

Irrecoverable Xmodem errors are always reported on the console. (These will not be visible if the console is also the transfer port, and will cause the Xmodem transfer to hang on the PC side. You can cancel the PC’s Xmodem transfer to abort, but the error message will be lost.)

You can stop these commands while they running by typing Control-C, unless the transfer port is also the console.

XR [bbbb] **Xmodem Receive**: receive a file from the transfer port and write it to DECtape starting with DECtape block bbbb. Verify all data as it is written, reporting errors on the console. If bbbb is not specified, then start with DECtape block 0.

XS [bbbb [nnnn]] **Xmodem Send**: send nnnn DECtape blocks as a file from DECtape to the transfer port, starting at DECtape block bbbb. If nnnn is not specified, then send until the end of the DECtape. If bbbb is not specified, then start with DECtape block 0. Memon-8 will display unrecoverable lost block and DECtape checksum errors on the console.

XV [bbbb] **Xmodem Verify**: receive a file from the transfer port and compare it to DECtape starting with DECtape block bbbb. If bbbb is not specified, then start with DECtape block 0. Memon-8 will display mismatches and unrecoverable DECtape errors on the console.

4.3 Errors

4.3.1 Command Line Errors

Memon-8 will print “HUH?” and return to the prompt for any command line error - either an unrecognized commands or an illegal value.

4.3.2 DECtape Errors

Memon-8 will retry DECtape failures during seeking or reading several times before giving up, either with a fatal error or a non-fatal error.

Memon-8 will print an error message on the console and return to the prompt for any fatal DECtape error:

LOST BLOCK ERROR: the seek routine could not find a block. This usually implies a problem with the tape’s format.

SELECT?: the specified unit is not selected. This usually means the selected unit’s switch on the TU56 is not set to “REMOTE”.

TIMING ERROR: the TD8-E reported a timing error. This usually implies a problem with the tape’s format or the hardware. In some cases, Memon-8 can’t tell a timing error from a select error - both are reported as SELECT? errors.

WRITE LOCK?: a write operation was attempted while the TU56 “WRITE ENABLE”/”WRITE LOCK” switch was in the “WRITE LOCK” position.

Memon-8 will print an error message on the console and continue with the next block of the operation for any non-fatal error. These error messages are suppressed if the transfer port is also the console:

LOST BLOCK ERROR: Memon-8 could not find a block after several tries. (This is a fatal error for the TB command, but non-fatal for the XS command.)

CHECKSUM ERROR: the checksum byte for the DECtape block did not match the computed checksum for the block’s data, after several reads of the block. (This is a fatal error for the TB command.)

VERIFY ERROR: the DECtape block's data did not match the data in the buffer. (This error only occurs for the verify commands and the XR command during its verification phases.)

4.3.2.1 Retries

For DECtape read and write operations, Memon-8 will retry several times for lost-block errors.

During DECtape read operations, Memon-8 will try several times to re-read any block that failed due to a checksum error.

No retries are attempted for checksum or verify errors during DECtape verify operations.

No retries are attempted for timing errors, as these errors indicate a more serious problem.

4.3.3 Xmodem Errors

Memon-8 recognizes several types of Xmodem errors. For some, it will retry the block, while others cause the transfer to abort, as defined in the Xmodem protocol. Memon-8 prints an error message if it aborts a transfer due to an irrecoverable Xmodem error. (These error messages will be eaten by the PC-side Xmodem program if the transfer port is the same as the console.)

XMODEM BABBLING: (XR and XV) at least 4096 characters were received without a pause. One cause of this error is sending the file directly, without using the Xmodem protocol.

XMODEM SYNC ERROR: (XR and XV) a complete, error-free block was received with the wrong block number, implying that a block was lost.

TOO MANY XMODEM ERRORS: (All Xmodem commands) a block was retried more than 10 times without success, or the program timed out waiting for the other side to respond.

TAPE OVERRUN: (XR and XV) the received data would write beyond the last block on the tape (block 2701).

4.4 Relocating Memon-8

You can relocate Memon-8 to any memory field using Memon-8's commands. As an example, to relocate Memon-8 from field 0 to field 7 and then execute in that field, use the following procedure. (The user types the underlined text.)

```
>CO 0 70000 5400    {Copies 5400 words from field o to field 5}
ok
>EX 70200           {Executes at address 0200 in field 5}

MEMON-8   1.18      {Memon-8 is now running in field 5}
BY M.  EBERHARD
```

```
    BUFFER AT 60000     {Memon-8 put its buffer in field 6 when running
                         in field 7.}
    >
```

4.5 Loading Binary Files with Memon-8

If necessary, use the RL command to restore the BIN and RIM loaders to their correct locations in core. Then type “EX 7777” to execute the BIN loader. Start the paper tape reader, and wait for the BIN loader to halt. If the loaded program did not overwrite Memon-8, you can use the front panel to restart Memon-8 at address 0200 in whichever field it is installed.

In the following example, Memon-8 is running in field 7, and a binary tape is to be loaded into field 0. Once the paper tape is loaded, Memon-8 is restarted:

```
    MEMON-8   1.18      {Memon-8 is now running in field 7}
    BY M.  EBERHARD

    BUFFER AT 60000     {Memon-8's buffer is still in field 6}

    >RL 0               {Restore the loaders in field 0}
    OK
    >EX 7777            {Execute the BIN loader in field 0}

  <Start the paper tape reader>
  <Wait for the "RUN" light to go off>  {The load is done}

  <Enter 0077 on the switch register>
  <Toggle the "EXTD ADDR LOAD" switch>  {Field 7 is selected}

  <Enter 0200 on the switch register>
  <Toggle the "ADDR LOAD" switch>
  <Toggle the "CONT" switch>            {Execute Memon-8}

    MEMON-8   1.18      {Memon-8 is again running in field 7}
    BY M.  EBERHARD

    BUFFER AT 60000

    >
```

4.6 Loading RIM Files with Memon-8

If necessary, use the RL command to restore the BIN and RIM loaders to their correct locations in core. Then type “EX 7756” to execute the RIM loader. Start the paper tape reader, and wait for tape to finish being read. Then toggle the “HALT” switch to stop the RIM loader. If you want to restart Memon-8 at this point, follow the procedure in section 4.5.